Implicit View-Time Interpolation of Stereo Videos using Multi-Plane Disparities and Non-Uniform Coordinates

Avinash Paliwal[1] Andrii Tsarov[2] Nima Khademi Kalantari[1]
[1]Texas A&M University [2]Leia Inc.
{avinashpaliwal,nimak}@tamu.edu, andrii.tsarov@leiainc.com

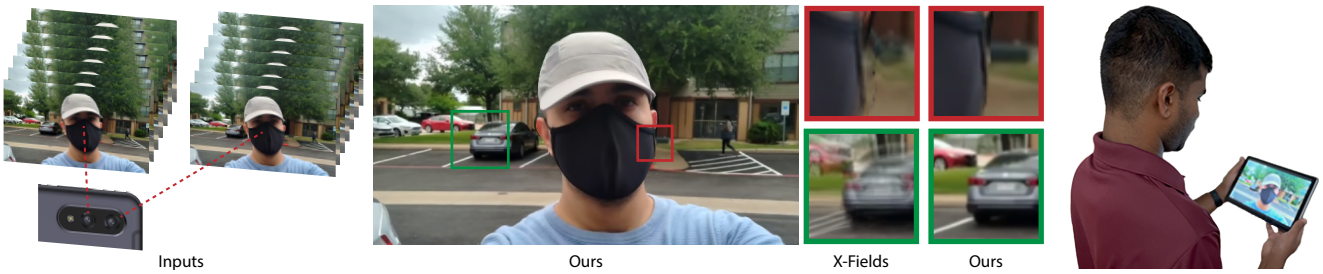

Figure 1. We propose a lightweight approach to reconstruct images at novel views and times from a stereo video, captured with standard cameras (e.g., cellphones). We build upon X-Fields and propose several key ideas including regularization losses and non-linear coordinates to significantly improve the results. Our method runs in near real-time rates (23 fps) and has low memory and storage costs. Our system can be deployed on VR and light field displays to provide an immersive experience for the users (Lume Pad with a light field display shown on the right).

In this supplementary document, we provide additional implementation details, ablation results, and qualitative comparisons.

1. Blending Network

In this section, we provide further details about the blending network training. As discussed in the paper, we train a UNet on the Vimeo90K [8] dataset to generate weight maps to blend the warped frames. Each scene in Vimeo90K consists of 7 frames. We use the corner frames (1 and 7) as reference frames and randomly select an intermediate frame from 2-6. We apply data augmentation techniques like random horizontal and vertical flip, and cropping to the sequence. We use Zhang et al.'s optical flow method [9] to estimate the flow between intermediate and reference frames. The reference frames are warped using these flows and the weight maps from the UNet are used to smoothly blend these warped frames. We use the following loss to train our blending network:

$$\mathcal{L} = \lambda_r \mathcal{L}_r + \lambda_p \mathcal{L}_p \tag{1}$$

where λ_r and λ_p are set to 100 and 0.05 in our implementation. The two terms are described below.

Reconstruction Loss $\mathcal{L}_r$ We use the commonly used pixel-wise $\mathcal{L}_1$ loss [2,5] between the blended image, $\tilde{I}_t$, and the ground truth intermediate frame, I_t.

$$\mathcal{L}_r = \|\tilde{I}_t - I_t\|_1. \tag{2}$$

Perceptual Loss $\mathcal{L}_p$ In addition, we utilize a VGG based perceptual loss [2, 5] to improve the details in the blended image [10]. Specifically, we define the loss function as:

$$\mathcal{L}_p = \|\phi(\tilde{I}_t) - \phi(I_t)\|_2^2, \tag{3}$$

where ϕ is the response of `conv4_3` layer of the pre-trained VGG-16 network.

Architecture The UNet architecture is described in Fig. 2. This network takes a 16 channel input (two input images, warped images and the corresponding flows) and estimates a one channel weight map. We use a LeakyReLU activation function for all the layers except the output for which we use sigmoid.

2. Multi-plane Disparity

We provide another illustrative figure (Fig. 4) for multi-plane disparity using a real example. The network outputs multi-planes with objects close to each other in both views u_j. These planes are shifted and merged using the `max` operator to obtain the final disparity, $J(u_j, t_i)$. Although the

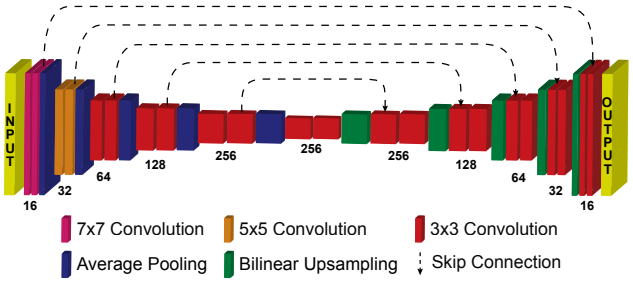

Figure 2. This figure shows the UNet architecture used for our blending network. We use a LeakyReLU activation function for all the layers except the output for which we use sigmoid.

Table 1. Effect of core components on view synthesis.

	PSNR	SSIM	LPIPS
Appearance Loss	27.11	0.843	0.0519
+ Mask + Jacobian Loss	28.34	0.870	0.0460
+ Positional Encoding	28.49	0.868	0.0456
+ Multi-plane disparities	**33.39**	**0.935**	**0.0242**

reference view Jacobian has large disparity, the encoded objects are spatially close which enables high quality interpolation.

3. Quantitative Results Dataset

In this section, we describe the steps to prepare the quantiative evaluation dataset used in our paper. We numerically compare our approach against the other methods on two lightfield video datasets, Sintel [3] and LFVID [7]. Sintel dataset contains 23 9x9 lightfield videos of 7 distinct scenes each with 20 to 50 frames. Among these we selected 5 sequences from distinct scenes that had larger number of frames (2 scenes contained only short sequences): `ambushfight_5`, `bamboo_1`, `chickenrun_3`, `foggyrocks_1`, as well as `shaman_2`. We use every other frame of the sequences `05_00` and `05_08` as the input left and right videos and the sequence `05_04` as our ground truth. We center crop the images to 360p. The LFVID dataset contains 4×4 light field videos, from which, we select 6 scenes: `Birthday`, `Hands`, `Painter`, `Rugby`, `Theater` and `Train`. We use every other frame of the videos from views 0 and 2 (from the second row) as our input and the video for view 1 as the ground truth. We downsample all the images by a factor of 2 and center crop them to 360p.

4. Ablation Experiments

Here, we evaluate the effectiveness of various components in our model. We use the `Painter` scene from LFVID [7] dataset for numerical ablations of view synthesis. In Table 1, we show the effectiveness of core components of view synthesis algorithm. The Jacobian loss with masked appearance loss significantly improves the output quality by applying occlusion aware training. Positional encoding improves the fine details of encoded Jacobians. Our multi-plane approach drastically improves the output quality as it is able to handle scenes with large disparity.

Table 2. Effect of components on multi-plane disparity training.

	PSNR	SSIM	LPIPS
w/o per plane regularization	28.41	0.892	0.0428
w/o disparity mask	31.48	0.924	0.0322
w/ pixel-wise sum	31.51	0.907	0.0316
w/ pixel-wise max (Ours)	**33.39**	**0.935**	**0.0242**

Table 3. Effect of number of planes on view synthesis.

#planes	PSNR	SSIM	LPIPS
2	27.66	0.860	0.0437
3	30.72	0.927	0.0297
4	32.13	0.930	0.0266
6	**33.39**	0.935	0.0242
9	33.23	**0.937**	**0.0235**

Table 4. Effect of components on time interpolation.

	PSNR	SSIM	LPIPS
Single Jacobian	32.88	0.954	0.0261
Dual Jacobian	33.12	0.954	0.0259
Non-Uniform coordinates	**33.59**	**0.959**	**0.0235**

We further explore the components of multi-plane disparity approach in Table 2. Without the per plane regularization, we see a drastic drop in performance highlighting its importance. Training without the disparity mask negatively affects the performance and creates sharp artifacts in the output. Using the pixel-wise `max` operator over `sum` to merge the planes gives a significant performance boost to our model. In addition, we demonstrate the effect of these components visually in Fig. 5. The first two rows show the guidance Jacobians and disparity masks for per plane regularization. The bottom four rows show the effect of multi-plane disparity components. While all the configurations are able to encode the guidance Jacobians ($J(u_1, t_i)$), the intermediate Jacobian ($J(u_{1.5}, t_i)$) contains artifacts for all the other configurations. Without per plane regularization, the network tends to put most of the details in one plane ($J_{d_5}(u_1, t_i)$). Note that while a couple of other planes contain content, they will be masked by the the fifth plane as it contains maximum disparity in most pixels. This leads to missing details in the intermediate Jacobian. Without dis-

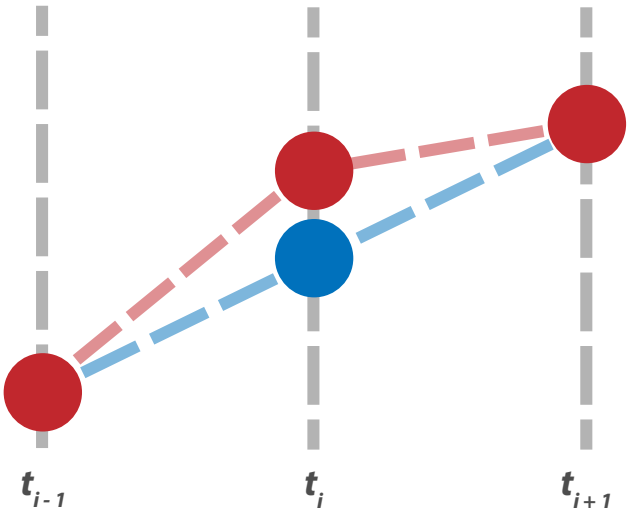

Figure 3. Here, we demonstrate the non-linear motion that frequently occurs in videos. The red dots show actual pixel motion at different time steps. Using a single Jacobian value to encode pixel motion to both left and right time steps results in averaging (blue dot) which leads to ghosting artifacts.

parity mask, the network has no flexibility in encoding the objects at different disparity. This ends up creating artifacts around plane boundaries (halo around the person) in the intermediate Jacobian. Using the pixel-wise `sum` operator creates fuzzy intermediate Jacobian. Our configuration with pixel-wise `max` operator creates detailed intermediate Jacobian without any significant boundary artifacts.

In Table 3, we show the effect of number of planes on the output quality. While the performance keeps improving until 6 planes, we do not observe significant improvement in performance beyond that. So, we choose $\#planes = 6$ to maximize the view synthesis quality while limiting the computation cost.

We use the CARS scene from our GoPro rig for numerical ablations of time interpolation. Since we captured the GoPro scenes at 240fps, we have ground truth frames for this experiment. In Table 4, we look at the time interpolation components of our model. Single Jacobian is unable to handle non-linear motion, as illustrated in Fig. 3. Using dual Jacobian is able to improve the time interpolation quality by handling small non-linear motion. Using non-uniform coordinates improves the performance by a good margin as it allows the network to encode highly non-linear motion and motion spikes. The effectiveness of non-uniform coordinates becomes apparent in visual comparisons. E.g., every time there is sudden change in camera or object motion in the supplementary video, X-Fields will generate glaring artifacts while our approach is able to smoothly handle them.

5. Results

5.1. Qualitative Results

We capture a set of stereo videos with a variety of motions using a stereo GoPro camera rig and Lume Pad [4] (stereo rear cameras as shown in Fig. 1). The Lume Pad videos are automatically time synchronized and rectified. For the GoPro stereo videos, we time synchronize the left and right videos using a high precision clock and synchronizing the clock transitions as shown in Fig. 9. We then perform uncalibrated stereo video rectification on the synchronized stereo videos.

We show comparisons against several state-of-the art approaches. Fig. 7 shows videos captured using the GoPro rig. The Lume Pad videos are shown in Fig. 8. For all the scenes, we show view-time interpolation at the middle of four observed view-time frames. As seen, other approaches produce results with noticeable ghosting and other artifacts, while our results are sharp and have clear boundaries.

We can also use video interpolation approaches like FILM [6] and RIFE [1] for view synthesis. However, these approaches tend to generate artifacts, as shown in Fig 6, which leads to temporal inconsistency.

References

[1] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 3, 6

[2] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[3] Takahiro Kinoshita and Satoshi Ono. Depth estimation from 4d light field videos. In *International Workshop on Advanced Imaging Technology (IWAIT) 2021*, volume 11766. International Society for Optics and Photonics, 2021. 2

[4] Leia. Leia inc. `https://www.leiainc.com/`, 2022. 3, 7

[5] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1710, 2018. 1

[6] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *European Conference on Computer Vision (ECCV)*, 2022. 3, 6

[7] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Tristan Langlois, Remy Gendrot, Olivier Bureller, Arno Schubert, and Valerie Allie. Dataset and pipeline for multi-view light-field video. In *CVPR Workshops*, 2017. 2

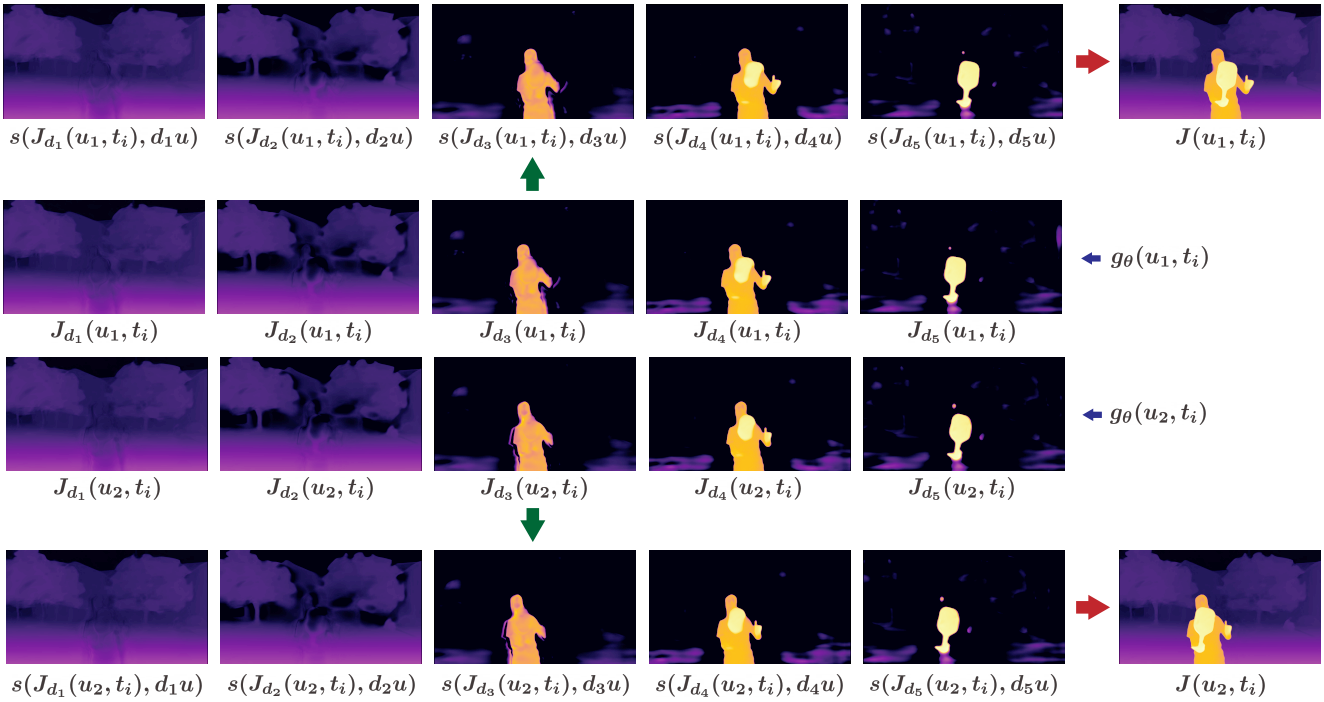

Figure 4. This figure demonstrates the multi-plane disparity approach with a real example. The network uses 5 planes to encode the scene disparity. You can see that the network learns to separate objects ($g_\theta(u_j,t_i)$) into separate planes ($J_{d_k}(u_j,t_i)$) with similar texture. For both views u_j, the objects in each plane are close to each other which enables high quality interpolation. These planes are then shifted ($s(J_{d_k}(u_j,t_i),d_ku)$) and merged using pixel-wise `max` operation to obtain the final view Jacobian ($J(u_j,t_i)$).

[8] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019. 1

[9] Feihu Zhang, Oliver J. Woodford, Victor Adrian Prisacariu, and Philip H.S. Torr. Separable flow: Learning motion cost volumes for optical flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10807–10817, 2021. 1

[10] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 1

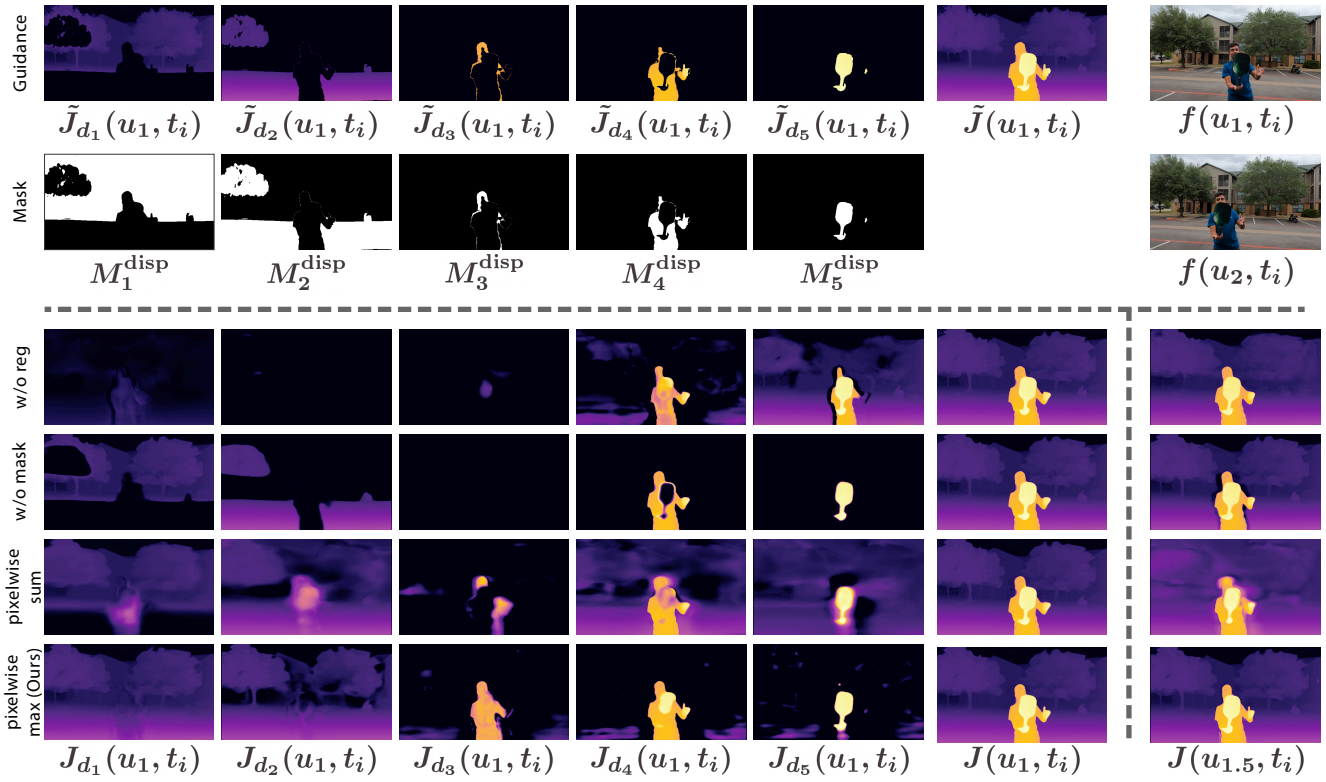

Figure 5. The top two rows consist of per plane guidance Jacobians ($\tilde{J}_{d_k}(u_1, t_i)$), disparity masks ($M_k^{disp}$), guidance Jacobian ($\tilde{J}(u_1, t_i)$) and the input reference frames ($f(u_j, t_i)$). The bottom four rows qualitatively demonstrate the effect of different components on multi-plane disparity training. Note that all configurations are able to encode guidance Jacobians correctly ($J(u_1, t_i)$). However, the quality of intermediate Jacobian used for view synthesis ($J(u_{1.5}, t_i)$) changes depending on the setting. Without per plane regularization (first row), the network tends to encode all information in a single plane ($J_{d_5}(u_1, t_i)$) which leads to loss of details in the intermediate Jacobian. Without disparity mask (second row), the network has a hard constraint to exactly encode the per plane guidance Jacobians which leads to artifacts around plane boundaries during view synthesis. Using `sum` operator to merge planes generates severe artifacts (third row) at intermediate coordinates. Our approach with `max` operator (fourth row) is able to generate high quality intermediate Jacbians.

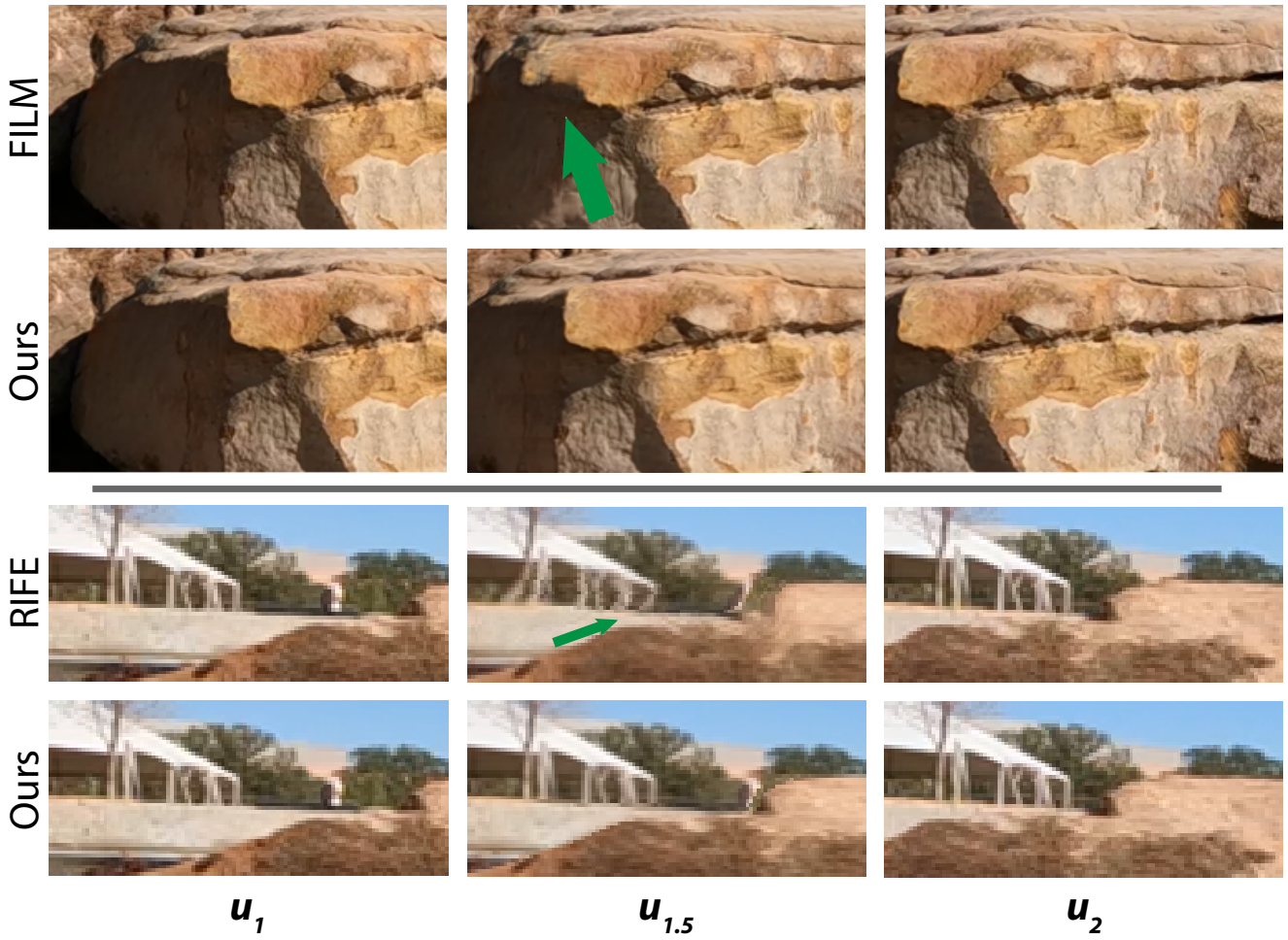

Figure 6. Another approach to view-time synthesis would be to use a video interpolation model to perform both view and time synthesis. However, without the constraint of static motion, these models tend to introduce artifacts that result in temporal inconsistency. E.g., while FILM [6] is able to handle large motion, it sometimes deforms the texture to smoothly animate between the two views. Our approach is able to correctly warp the texture. RIFE [1] on the other hand, while able to synthesize frames in real-time, struggles at motion boundaries.

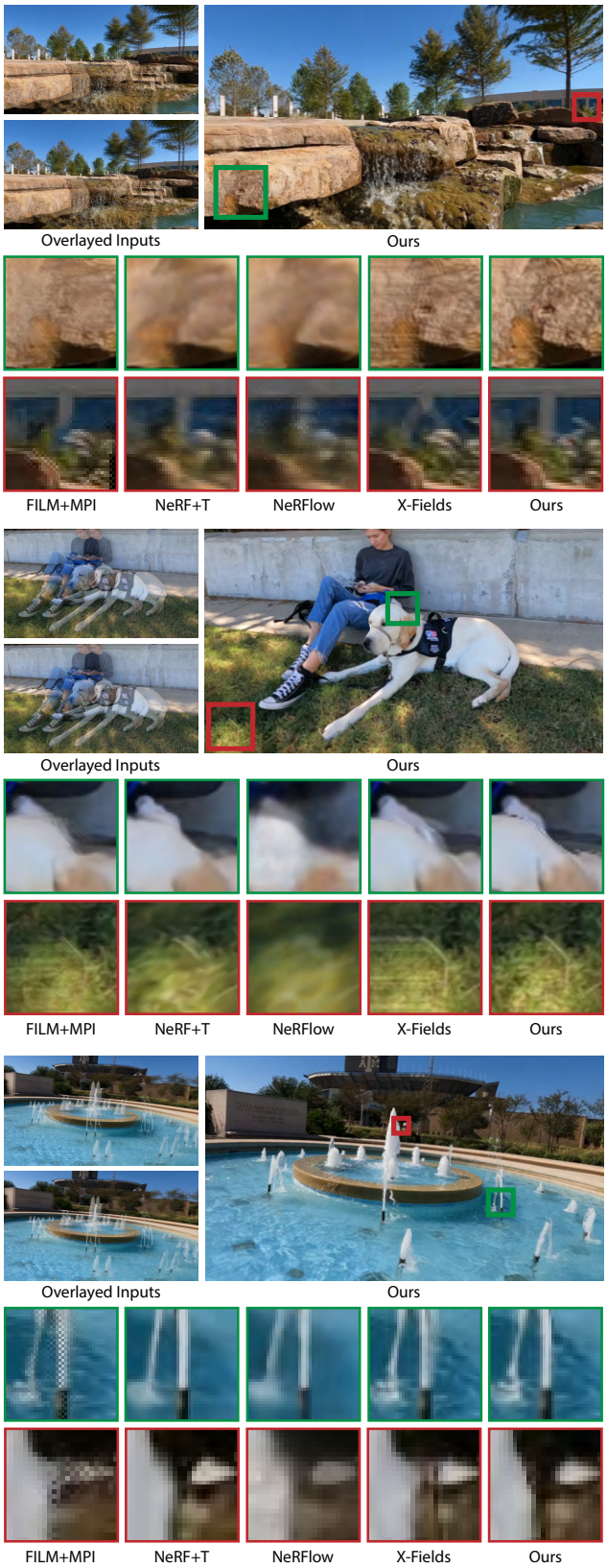

Figure 7. Comparison against several state-of-the-art methods on view-time interpolation. On the left we show the overlayed left and right views for two consecutive frames neighboring the coordinate of interest.

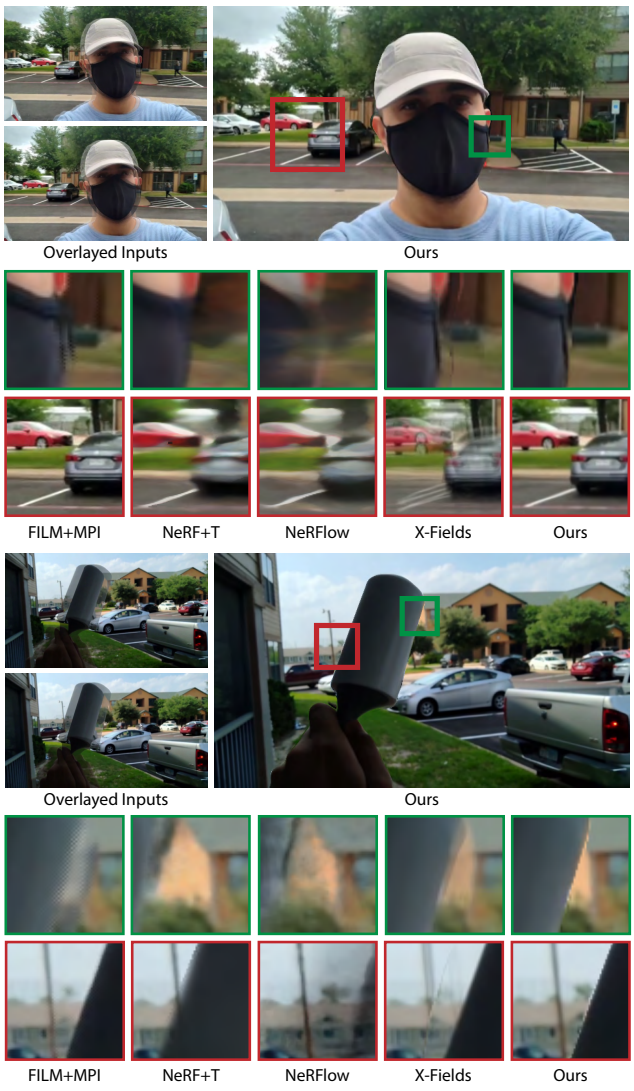

Figure 8. Comparison against several state-of-the-art view-time interpolation methods on Lume Pad [4] scenes. On the left we show the overlayed left and right views for two consecutive frames neighboring the coordinate of interest.

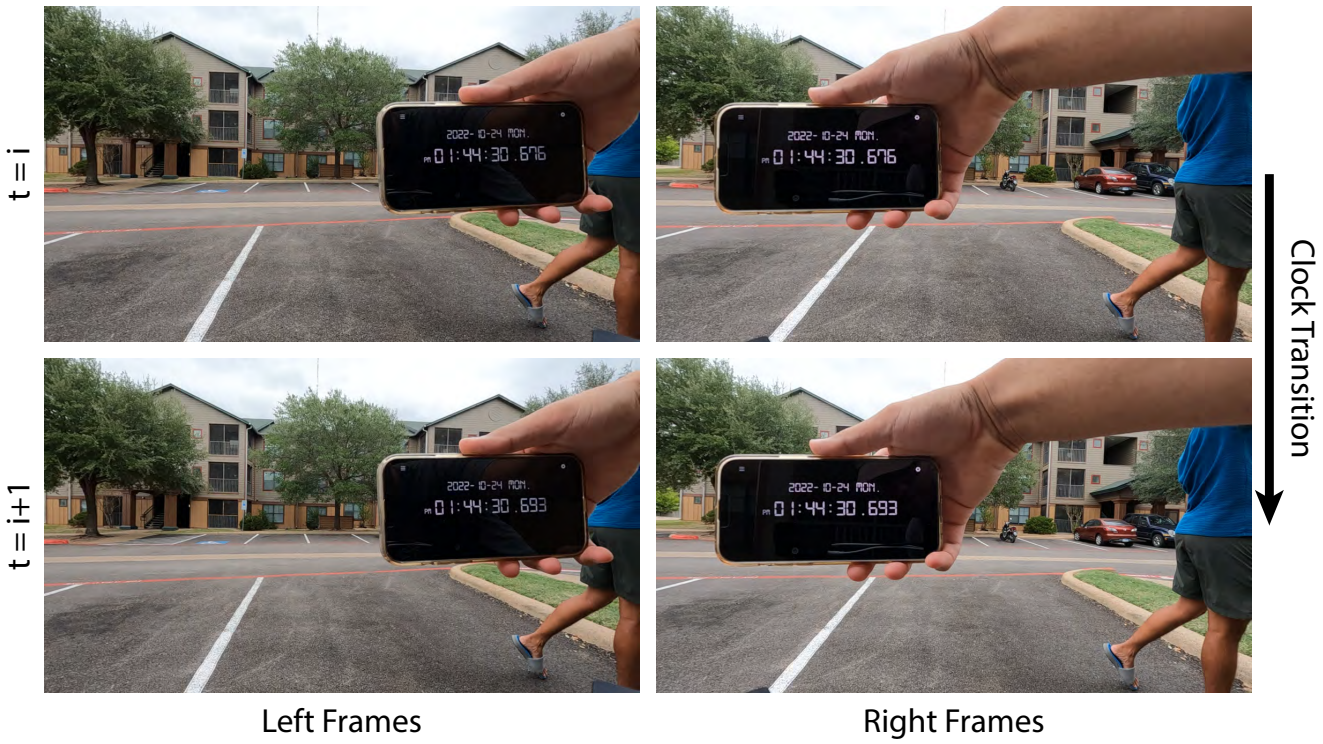

Figure 9. We observe the frames where the clock transitions to the next time stamp in both left and right frames. This transition is used to synchronize the stereo videos to the closest frame.