Extreme View Synthesis
Supplementary Material

Inchang Choi[1,2] Orazio Gallo[1] Alejandro Troccoli[1] Min H. Kim[2] Jan Kautz[1]
[1]NVIDIA [2]KAIST

1. Visualization of Parallax

Sometimes it is difficult to appreciate the amount of parallax in the synthesized views. To make parallax more apparent, we can first align the extreme views at a pixel in the background and flip between them. Figure 1 shows examples of this. Please view the figure in Adobe Reader, or any other media-enabled reader to play the animation.

Figure 1: Synthesized views aligned at the background to better show Parallax. **The images will flip between the two extreme views each time they are clicked on it in Acrobat Reader.**

2. Animations for the results in Figure 10

In Figure 10 of the paper we only show the extreme views that we generated. Please refer the videos accompanying this Supplementary document to see all of the intermediate frames we generate, which also empirically indicate that our method is temporally stable and that the synthesized images are physically accurate.

3. Comparison to Stereo Magnification

To showcase the ability of our algorithm to deal with extreme viewpoint changes, in the paper we show $30\times$ magnification factors on the data by Stereo Magnification [2]. For convenience, we also show these results in Figure 2-4 here. However, it is also worth showing that at $4.5\times$ the baseline magnification, both the method by Zhou *et al*. and ours work fine, as shown here in Figure 5-7.

4. The Threshold and the Weights for View Synthesis

The Threshold In Section 5 of the paper, we introduce the equation for synthesizing a view. The threshold t for the view synthesis is set as:

$$t = 0.075 \cdot \max\left(\mathcal{D}^{\text{NV}}(x, y, \cdot)\right), \tag{1}$$

where $\max\left(\mathcal{D}^{\text{NV}}(x, y, \cdot)\right)$ indicates the maximum depth probability for pixel (x, y) in the novel view.

The Weights R in Equation 1 of the paper is a function that merges pixels from $\mathcal{I}_i$ weighting them based on the distance of the cameras' centers, and the angles between the cameras' principal axes. It is defined as:

$$\sum_{i=1}^{N} \frac{w_{d_i} w_{a_i} \mathcal{I}_i(x_i, y_i)}{w_{d_i} w_{a_i}}, \tag{2}$$

where w_{d_i} and w_{a_i} is the distance weight and the angle weight of i-th input camera, respectively, with respect to the camera of the novel view. Further, the distance weight is defined as

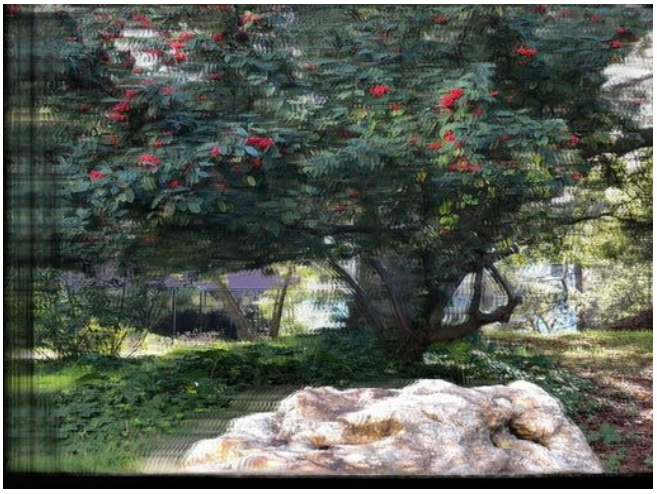
$$w_{d_i} = \alpha \cdot \exp(-\text{dist}\left(p_i, p^{\text{NV}}\right)/\sigma_d^2), \tag{3}$$

where p_i and p^{NV} are the positions of the i-th input camera and the novel camera. The function dist$(\cdot)$ computes the Euclidean distance between two point. Similarly,

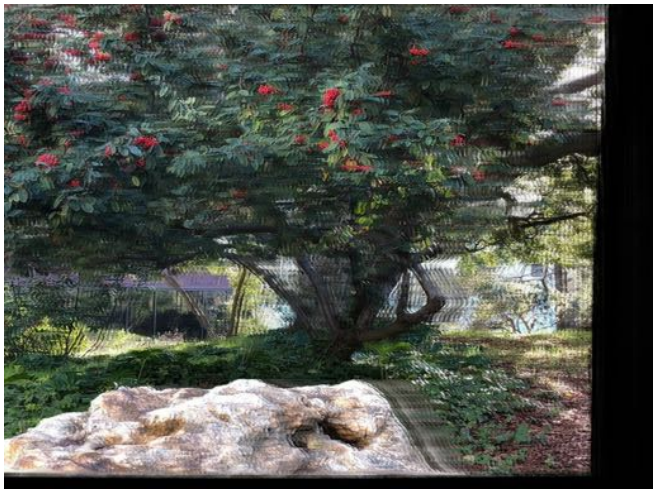
$$w_{a_i} = \alpha \cdot \exp(-\text{ang}\left(\mathbf{v}_i, \mathbf{v}^{\text{NV}}\right)/\sigma_a^2), \tag{4}$$

(a) input image (left)

(b) input image (right)



(c) 30x Stereo Magnification (left)

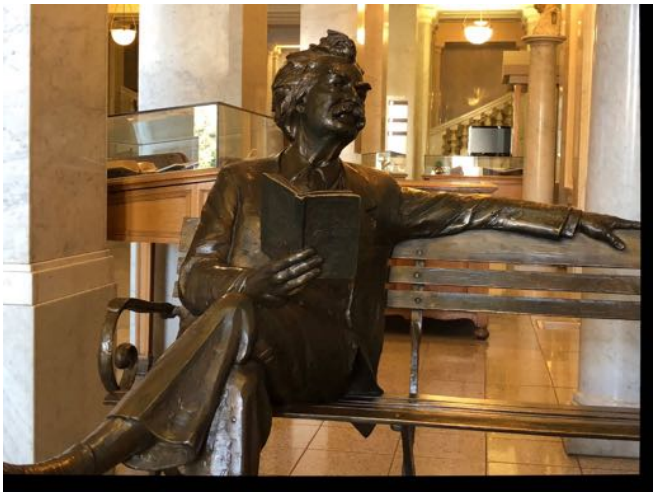


(d) 30x Stereo Magnification (right)

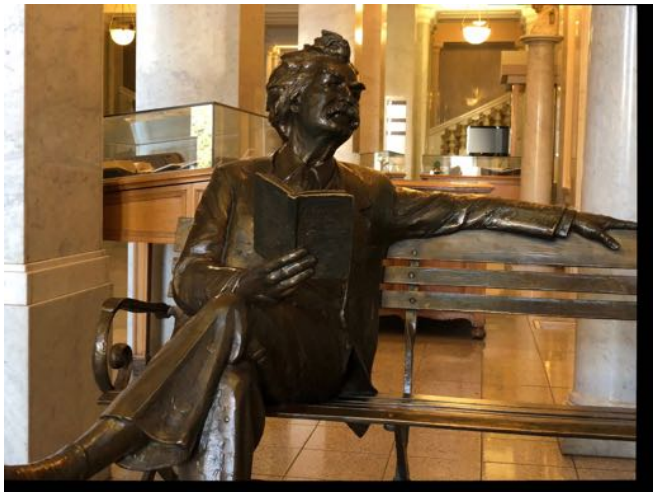
(e) 30x ours (left)

(f) 30x ours (right)

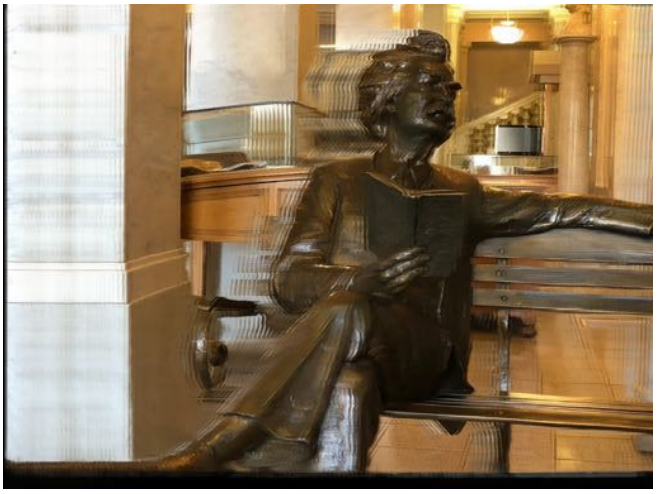
Figure 2: We extended the baseline of a stereo image pair by 30 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.



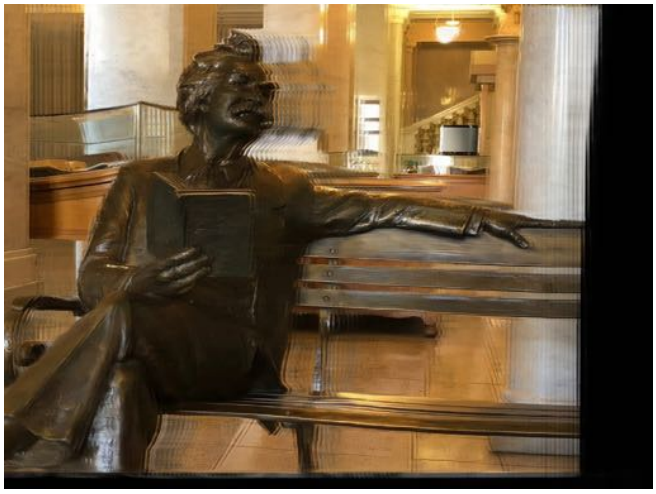
(a) input image (left)



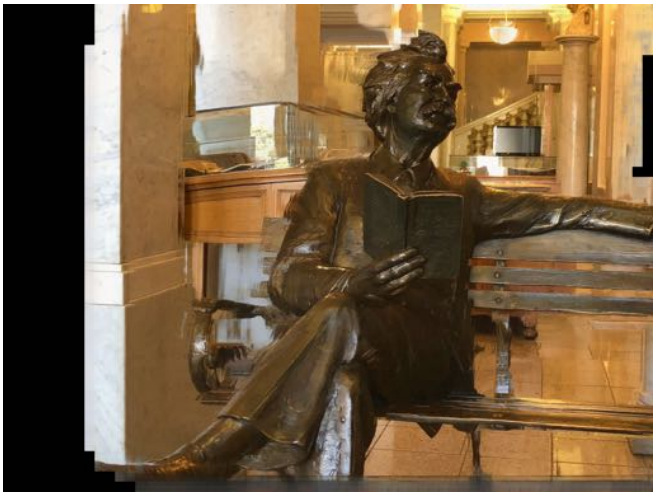
(b) input image (right)



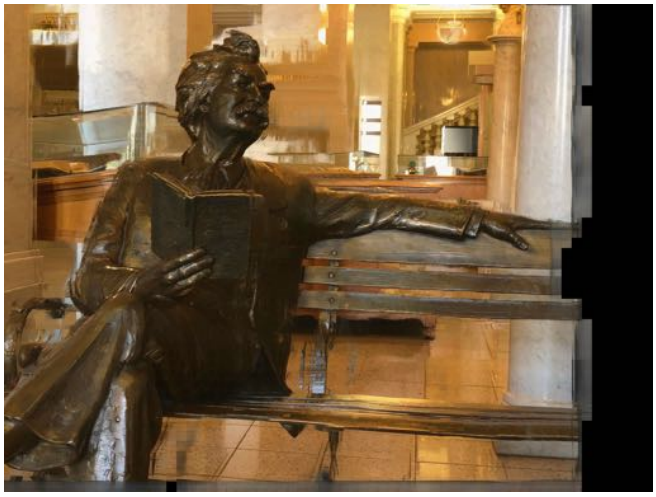
(c) 30x Stereo Magnification (left)



(d) 30x Stereo Magnification (right)



(e) 30x ours (left)

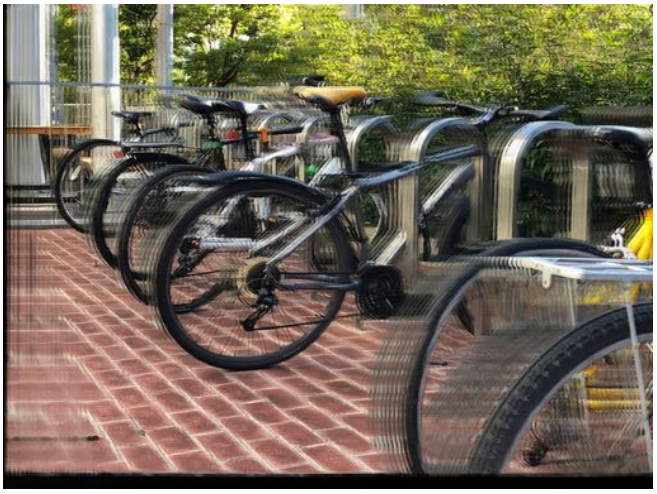


(f) 30x ours (right)

Figure 3: We extended the baseline of a stereo image pair by 30 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.

(a) input image (left)

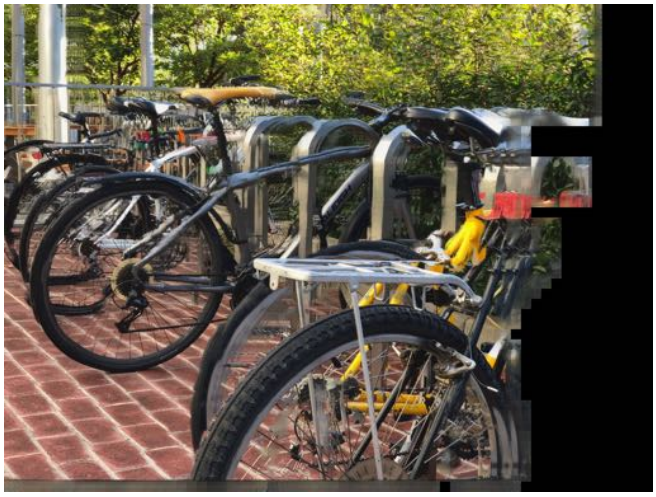
(b) input image (right)



(c) 30x Stereo Magnification (left)

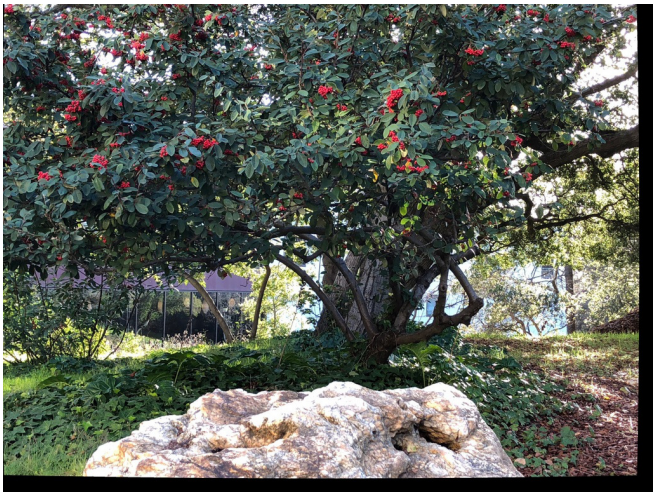
(d) 30x Stereo Magnification (right)

(e) 30x ours (left)



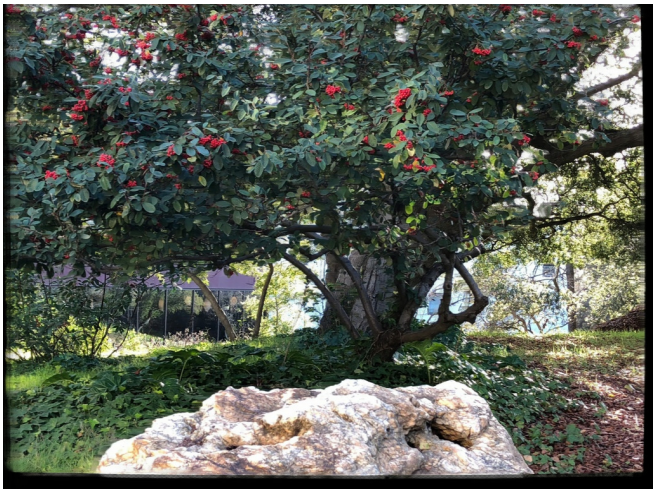
(f) 30x ours (right)

Figure 4: We extended the baseline of a stereo image pair by 30 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.



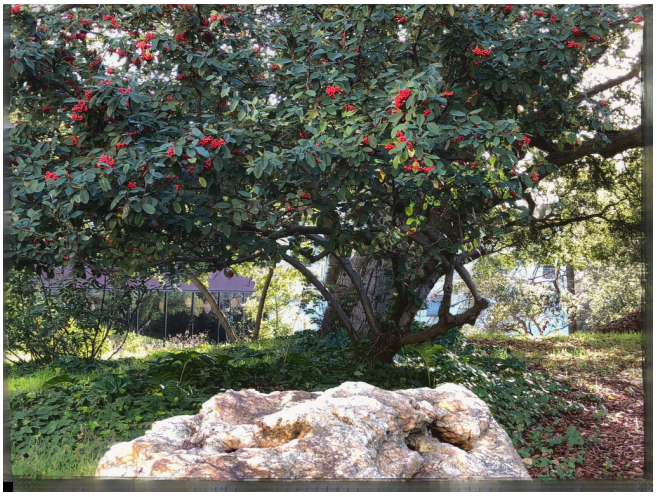
(a) input image (left)

(b) input image (right)



(c) 4.5x Stereo Magnification (left)

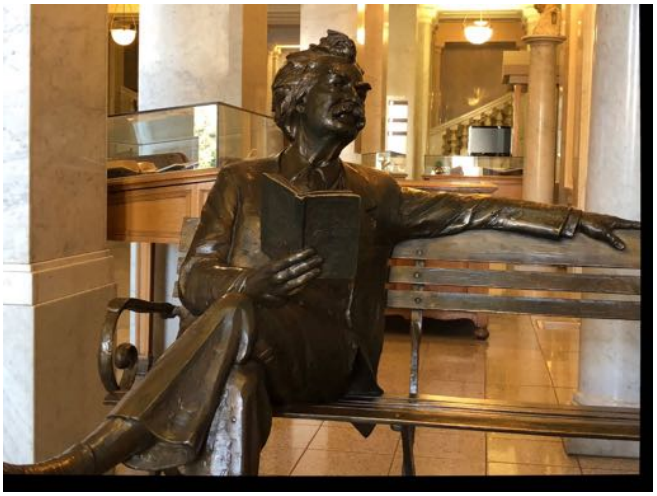
(d) 4.5x Stereo Magnification (right)



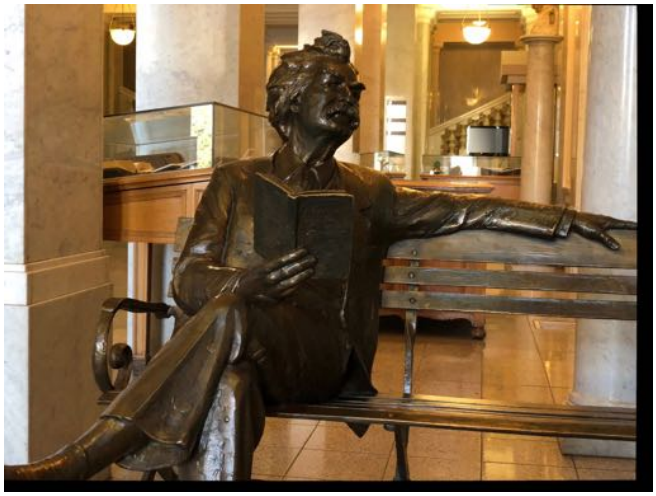
(e) 4.5x ours (left)

(f) 4.5x ours (right)

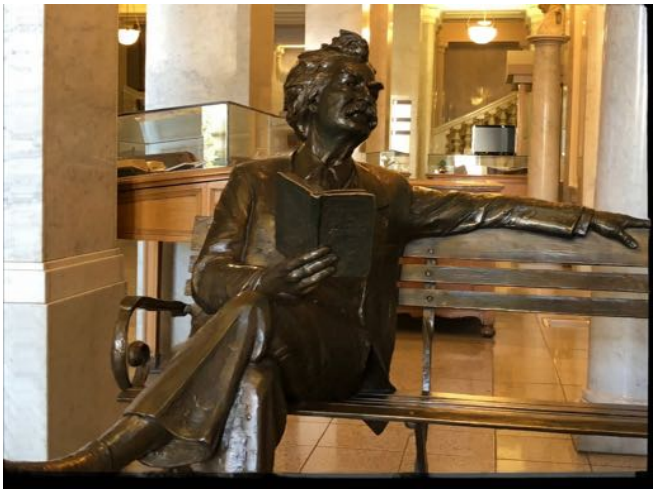
Figure 5: We extended the baseline of a stereo image pair by 4.5 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.



(a) input image (left)

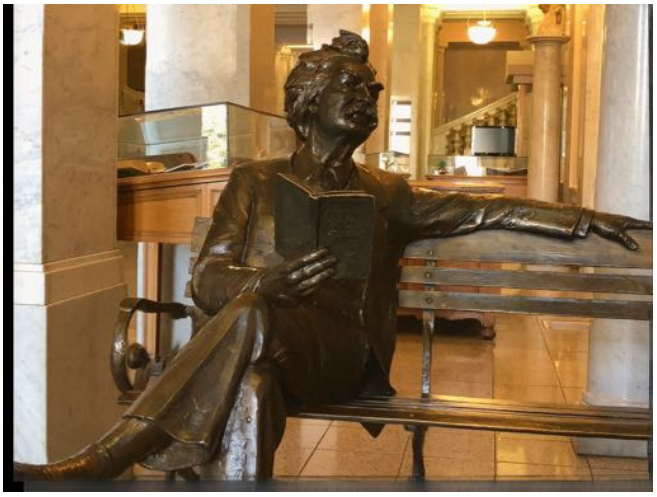


(b) input image (right)

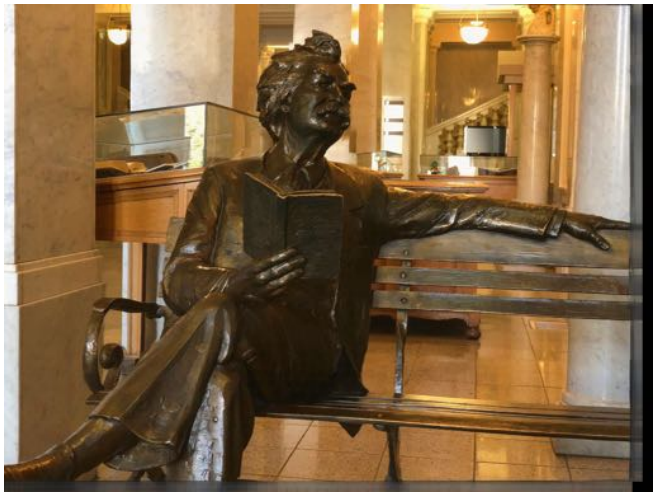


(c) 4.5x Stereo Magnification (left)

(d) 4.5x Stereo Magnification (right)



(e) 4.5x ours (left)



(f) 4.5x ours (right)

Figure 6: We extended the baseline of a stereo image pair by 4.5 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.

(a) input image (left)

(b) input image (right)

(c) 4.5x Stereo Magnification (left)

(d) 4.5x Stereo Magnification (right)

(e) 4.5x ours (left)

(f) 4.5x ours (right)

Figure 7: We extended the baseline of a stereo image pair by 4.5 times. The input images pair is shown in the first row in (a) and (b). The results from Stereo Magnification is shown in (c) and (d). Our view synthesis are placed in (e) and (f) of the last row.

where $\mathbf{v}_i$ and $\mathbf{v}^{\text{NV}}$ are the unit view vector of the i-th input camera and the novel camera. We assume that the angle between two view vectors is smaller than 90 degree. The function ang $(\cdot)$ corresponds to $\left|\mathbf{v}_i \cdot \mathbf{v}^{\text{NV}}\right|$. We set $\alpha = 100$, $\sigma_d = 0.2$, and $\sigma_a = 0.4$.

5. The Refinement Network

The Model Our refinement network, introduced in Section 6.1 of the paper, has the form of UNet that extracts multi-level features and has skip connections from the encoder to decoder. The details of each layer in our model is summarized in Table 1.

Training The weights are initialized by Xavier initialization. Batch normalization layers are attached after every convolution layers. We use ADAM optimizer. The initial learning rate lr_0 is set to 0.0001. We schedule the learning rate to reduce every epoch using the exponential decay as follows:

$$\text{lr}_i = \text{lr}_0 \cdot 0.92^i, \tag{5}$$

where i refers to the epoch index. We perform the training for 40 epochs. To train we use the MVS-Synth dataset [1]. From MVS-Synth dataset, we take 9 consecutive frames $\{F_1, F_2, ..., F_9\}$, and set $\{F_3, F_4, F_6, F_7\}$ as source views. Setting the novel perspective to the rest of the frames, we synthesize new views $\{\widetilde{F}_1, \widetilde{F}_2, \widetilde{F}_5, \widetilde{F}_8, \widetilde{F}_9\}$ to use them as inputs to the refinement network.

References

[1] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 8

[2] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning view synthesis using multiplane images. In *ACM Transactions on Graphics (SIGGRAPH)*, 2018. 1

Layer	k	s	in	out	bnorm	input
enc_1_1	3	1	3	128	N	an image patch $\widetilde{P}$
enc_1_2	3	1	128	128	Y	enc_1_1
enc_1_3	3	2	128	256	Y	enc_1_2
enc_2_1	3	1	256	256	Y	enc_1_3
enc_2_2	3	2	256	512	Y	enc_2_1
enc_3_1	3	1	512	512	Y	enc_2_2
enc_3_2	3	2	512	512	Y	enc_3_1
dec_1_1	3	1	1024	512	Y	$\widetilde{P}^{\text{NV}}$_enc_3_2 + M_enc_3_2
dec_1_2	3	1	512	512	Y	dec_1_1
dec_1_3	3	**up**	512	512	Y	dec_1_2
dec_2_1	3	1	1536	512	Y	$\widetilde{P}^{\text{NV}}$_enc_3_1 + M_enc_3_1 +dec_1_3
dec_2_2	3	1	512	512	Y	dec_2_1
dec_2_3	3	**up**	512	256	Y	dec_2_2
dec_3_1	3	1	768	256	Y	$\widetilde{P}^{\text{NV}}$_enc_2_1 + M_enc_2_1 +dec_2_3
dec_3_2	3	1	256	256	Y	dec_3_1
dec_3_3	3	**up**	256	128	Y	dec_3_2
dec_4_1	3	1	374	128	Y	$\widetilde{P}^{\text{NV}}$_enc_1_2 + M_enc_1_2 +dec_3_3
dec_4_2	3	1	128	128	Y	dec_4_1
dec_4_3	3	1	128	3	N	dec_4_2

Table 1: The encoder architecture of our refinement network in Section 6 of the paper, where **Layer** is the name of layers, **k** is the kernel size, **s** is the stride, **in** and **out** are the number of input and output channels, **bnorm** indicates whether there is a batch-normalization layer attached, and **input** refers to the input of each layer. The layers in the encoder have the name beginning with **enc**, and those in the decoder take **dec** as a prefix. The layers, dec_1_1, dec_2_1, dec_3_1, and dec_4_1, take the feature of $\widetilde{P}^{\text{NV}}$ and the maxpooled features from $\widehat{P}_{i,j}$. The names of these features are prefixed with $\widetilde{P}^{\text{NV}}$ and M, respectively. **up** in the column **s** indicates that 2x-upsampling is performed on the input of the layer. Every layer except the last layer dec_4_3 is activated by ReLU activation function.