Wide2Long: Learning Lens Compression and Perspective Adjustment for Wide-Angle to Telephoto Translation

Supplementary Material

1. Additional Details and Experiments

1.1. Details of Data Acquisition

As mentioned in Section 5.1 of the main paper, we use both real-world and synthetic image pairs in our dataset on lens compression for $f_t = 2f_s$ change in focal length. For capturing real-world images, we use a *Panasonic Lumix G7* mirrorless camera, with a micro four-thirds image sensor. The sensor has a crop factor of 2 when compared to a full frame (35 mm) sensor, implying that the equivalent focal length of any lens used on the camera is doubled.

For the source images we use a *Sigma* 18-35 mm, F/1.8 lens with a $0.7\times$ speed-booster, giving us an equivalent focal length of 25.2 mm on the wider end. For the target images, we use a *Panasonic Lumix* 25 mm, F/1.7 lens, giving us an equivalent focal length of 50 mm. We follow the methodology for capturing images as outlined in Section 3 of the main paper. To keep the size of the subject in both source and target images constant, we use visual markers (e.g. masking tapes) on the camera monitor or within the scene (which we remove while capturing) to serve as a reference. All camera parameters like aperture, shutter speed, ISO, white balance, etc. are kept constant while capturing an image-pair. 168 image pairs among the 600 in our dataset are captured manually.

Figure 1. Source-target pair examples in our dataset, captured manually.

To capture synthetic images we use a combination of two software platforms - Unity[4] and Taitopia[5]. Both are 3D ray-tracing, modeling and scene rendering applications, with access to a huge repository of open-source, pre-designed 3D scenes and assets, although Taitopia requires a rendering fee for each image. With Unity, we set the source and target focal lengths as 25 mm and 50 mm respectively, and capture the corresponding frames by using markers much like we did for the real-world images. In Taitopia, we had the added option of choosing a particular part of the scene to be in sharpest focus, and retain the size of that part when switching between focal lengths, making the image capturing process faster.

Figure 2. Source-target pair examples in our dataset, captured synthetically.

Both Unity and Taitopia generate accurate lens compression effects by leveraging advanced rendering pipelines and precise camera simulation algorithms[6]. In Unity[7], features like the High-Definition Render Pipeline (HDRP) enable accurate modeling of lens behaviors, including focal length, field of view, and distortion effects that replicate real-world lens compression. This is achieved by adjusting the perspective projection matrix and dynamically simulating how objects appear closer or further apart based on the camera's settings. Taitopia complements these techniques with specialized 3D capture technology and algorithms[8] that refine spatial accuracy and depth mapping, ensuring photorealistic compression effects. Together, these tools create a robust platform for producing lifelike visuals where lens compression accurately mimics physical optics. As before, we kept all other camera parameters constant for any source-target pair.

Out of all image-pairs acquired, we randomly pick 100 image-pairs to form our test set, on which we evaluate Wide2Long's performance in terms of SSIM, PSNR and LPIPS in Section 5.2 of the main paper.

[4]https://unity.com/
[5]https://taitopia.design/

[6]Gregory J. Ward. The radiance lighting simulation and rendering system. *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 459–472, 1994.
[7]https://docs.unity.com/
[8]https://docs.taitopia.design/

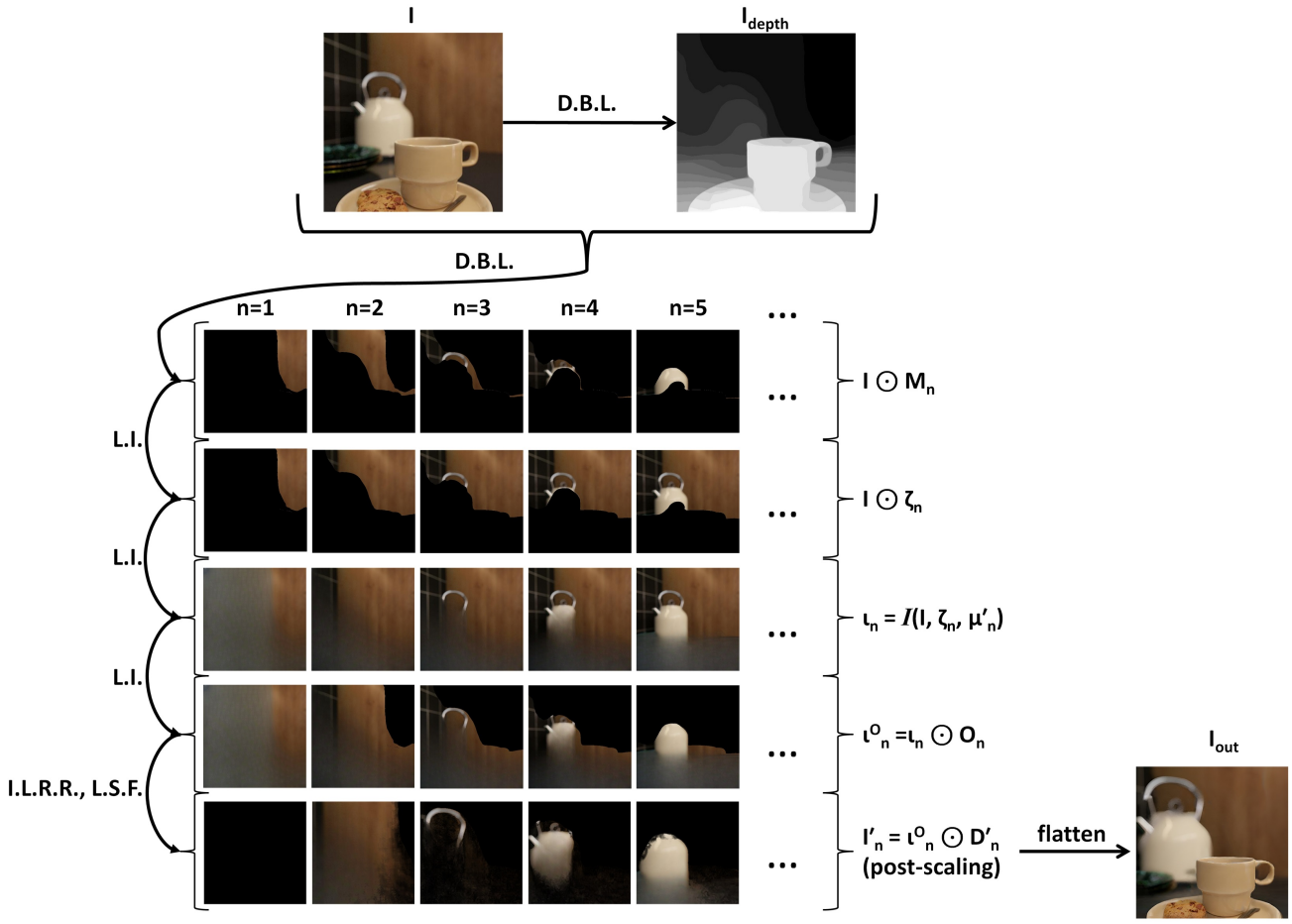

Figure 3. Visual example of the W2L pipeline in operation. Note that the final row shows the inpainted layers after applying the difference map and scaling.

1.2. Wide2Long: Layer Generation Example

We provide an instance of the Wide2Long pipeline being applied to an image I (same as in Figure 1 of the main paper). The outputs after each stage (or a part of it) is shown step by step in Figure 3, right up to the formation of the scaled layers, which will be flattened to give the output image. We use the same notations and acronyms as in Section 4.2 of the main paper.

After generating the depth-map I_{depth} from I, the depth-masks M_n generated ($n = 0, 1, ..., N$) segregate I into $(N + 1)$ image regions. Each n^{th} image region is then stacked with the regions preceding it (if any) to give sufficient context input to the inpainting model $\mathcal{I}$ to fill the occlusion region (shown in black in the second row in Figure 3). This is the same as inpainting the occlusion in $(I \odot \zeta_n)$ using $\mathcal{I}$, with $(I \odot \mu'_n)$ serving as the dilated inpainting mask (as detailed in Section 4.2 of the main paper). The resultant inpainted outputs ι_n are then masked to subtract the additional stacked regions and produce the in-painted image regions ι^o_n. These ι^o_n are then masked in turn using the D'_n maps (computed as in Section 4.3 of the main paper) to give I'_n which are then scaled by an amount ρ_n, defined by Equation 12 in the main paper. The scaled I'_n's are then overlaid and flattened to give the final output image I_{out}.

1.3. How γ can vary with N

As stated in Section 4.5 of the main paper, the data-driven γ can scale differently for different values of N during the scaling factor computation.

To show this, we rewrite Equation (12) of scaling factor computation from the main paper for the case where a change in the value of $(N + 1)$ has resulted in a change in the difference between depth layer indices as follows:

$$\rho'(i, j) = e^{\gamma' \tau (n_f - n_{(i,j)})} \tag{1}$$

where the difference in the depth layer indices has changed from $(n_f - n_{(i,j)})$ to $\tau(n_f - n_{(i,j)})$, with $\tau \in \mathbb{R}^+$. As $(N + 1)$ is the number of depth layers, the difference between any

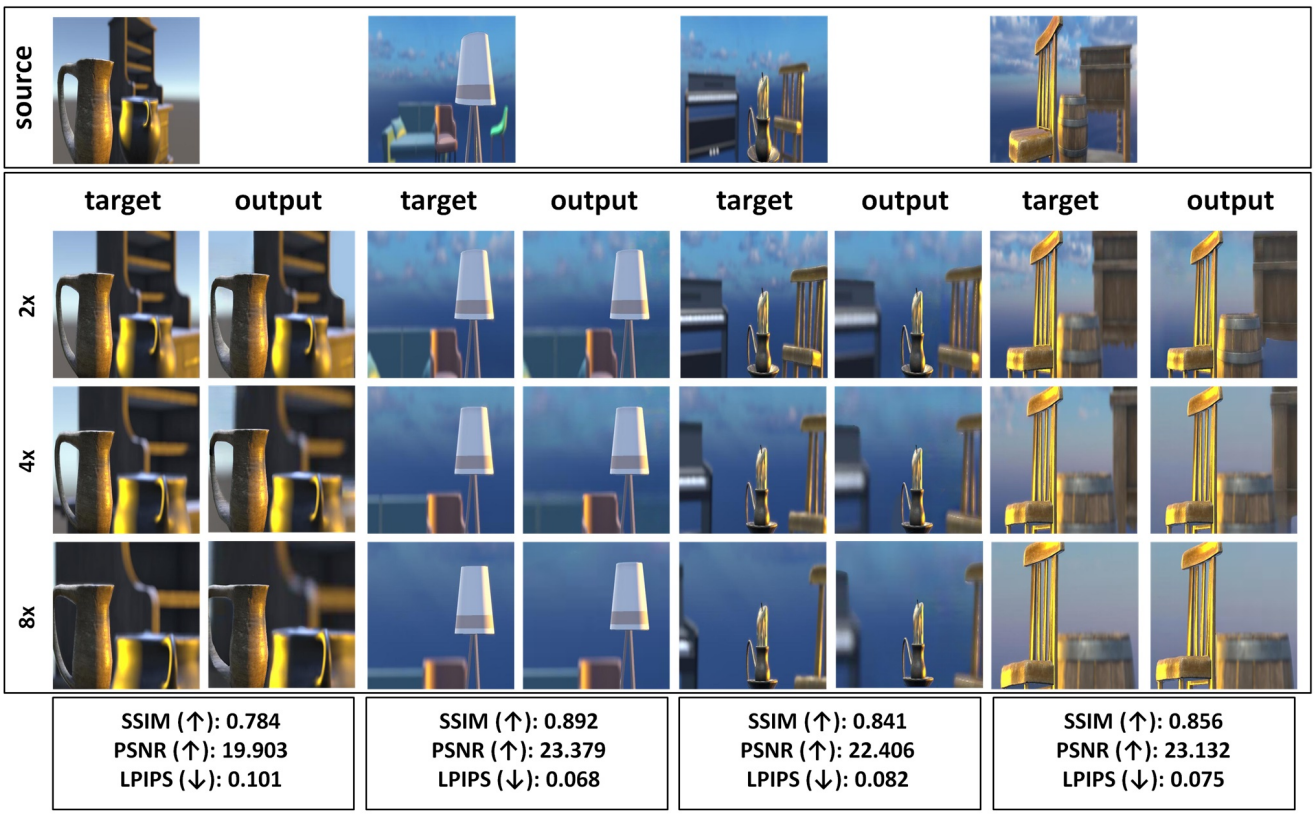

Figure 4. Qualitative results showing the performance of our W2L pipeline in simulating lens compression associated with $f_t = 2^p f_s$. To demonstrate $p = 1, 2, 3$ are chosen, and the metric values shown at the bottom of each column is the average obtained for the corresponding targeted $2\times$, $4\times$ and $8\times$ changes in focal length. The results for a particular $f_t = 2^p f_s$ has been obtained applying our W2L method iteratively p times.

two layer indices will obviously be scaled as per the change in the N value.

Now, if the appropriate scaling factor for pixel (i, j) is obtained when N has a particular value, it must not change with change in the value of N. So, we have $\rho'(i, j) = \rho(i, j)$. Therefore, comparing Equation (1) to Equation (12) in the main paper, we immediately have $\gamma' = (1/\tau)\gamma$.

This means that we can conveniently change the value of $(N + 1)$ (number of depth layers) during inference as per the user's choice and the nature of the data, while scaling the learnt γ by $(1/\tau)$. Thus, the extent of layering can be varied as per the requirements of the image data without the need to retrain the I2P model instances again.

1.4. Additional Lens Compression Results

We apply our Wide2Long pipeline to a few other $f_s \rightarrow f_t = 2^p f_s$ examples where $p = 1, 2, 3$. This experiment is to observe the performance of our pipeline in producing the necessary optical transformations for lens compression related to different p values. Figure 4 shows the qualitative and quantitative performance on such image-pairs with multiple $(f_s, 2^p f_s)$. The metric values at the bottom of each sample shows the averaged values of SSIM, PSNR and LPIPS across $p = 1, 2, 3$. Since our pipeline has been trained to simulate lens compression for a $2\times$ change in f_s, the first output from the pipeline (first row) serves in turn as the input to the W2L pipeline to generate the next higher order lens compression. So for a lens compression corresponding to a $2^p\times$ focal length change, we need p passes through the W2L pipeline. All images shown in the figure are generated using Unity.

As can be seen from the figure, in all the examples, the proposed pipeline performs lens compression by maintaining the size of the main subject and scaling the rest of the contents, achieving decent performance metric values. This shows that Wide2Long, although trained only once to generate $2\times$ lens compression, can be used iteratively to generate higher orders of targeted lens compression.

Performance comparison on Real and Synthetic Images All synthetic computer-generated (C. G.) images in our dataset are generated fundamentally with ray tracing[9]

[9] Andrew S. Glassner. *An Introduction to Ray Tracing.* Academic Press, 1989

algorithms, based on the principles of physical optics. So, the properties of Lens Compression should hold true for all synthetic images in our dataset, as it does for the real images. To further validate this point, we apply our Wide2Long pipeline on the real and synthetic image pairs in the test-set separately, and we observe that the performance of W2L remains same for real as well as for C.G. images. Table 6 reports these results.

Metric	Real (56 pairs)	C.G. (44 pairs)
SSIM (↑)	0.766	0.770
PSNR (↑)	20.078	19.613
LPIPS (↓)	0.140	0.138

Table 6. Performance on real and computer-generated image pairs.

Focal length-agnostic W2L. Further, it can be seen from Figure 4 that W2L can be applied to implement lens compression corresponding for any $2\times$ change in focal length. The results show that W2L can produce outputs consistent with the target image for focal length jumps such as $25 \rightarrow 50$ mm, $50 \rightarrow 100$ mm and $100 \rightarrow 200$ mm, and indicates that the Wide2Long pipeline is indeed focal length-agnostic and learns only the amount of lens compression, irrespective of the focal length values of the lenses.

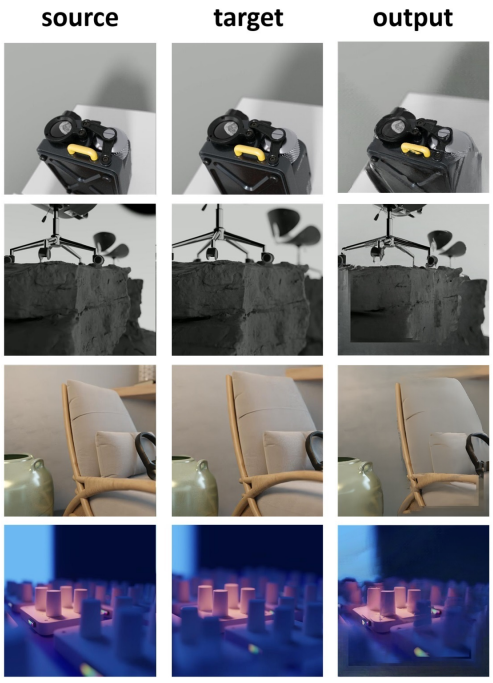

Figure 5. Some failure cases of the Wide2Long pipeline.

2. Failure Cases

We provide qualitative examples of failure cases across the test samples in our W2L dataset, as shown in Figure 5. These are attributed primarily due to improper parameter estimation, such as the scaling parameter γ and the in-painting parameter λ, for such samples. These result in undesired scaling of the depth-layers from foreground to background, along with gaps in the in-painting for cases where the lens compression might reveal objects or portions thereof which are outside frame in the source image. Also, inaccurate depth-estimation results in incorrect segregation of objects across layers, which leads to discrepancies.