Supplementary Material *for*
SRWarp: Generalized Image Super-Resolution under Arbitrary Transformation

Sanghyun Son Kyoung Mu Lee
ASRI, Department of ECE, Seoul National University, Seoul, Korea
{thstkdgus35, kyoungmu}@snu.ac.kr

S1. Details about the DIV2KW Dataset

We synthesize the proposed DIV2KW dataset with various random projective transformations. Warping parameters are determined by combining sheering, rotation, scaling, and projection matrices, denoted as H, R, S, and P, respectively. We construct M_i^{-1} first and inverse the matrix to implement the actual transform M_i to construct feasible transformations. Detailed specifications of the transformations are described as follows:

$$
\begin{aligned}
M_i^{-1} &= HRSP, \\
H &= \begin{pmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\
R &= \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\
S &= \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\
P &= \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ p_x & p_y & 1 \end{pmatrix},
\end{aligned}
\tag{S1}
$$

where the variables are randomly sampled from uniform ($\mathcal{U}$) or normal ($\mathcal{N}$) distributions. Table S1 shows parameters of the random distributions we use. We note that the projection matrix P varies depending on the size of the HR sample $\mathbf{I}_{\text{HR}}$ to normalize image shapes after warping.

For training, we randomly crop N many 384×384 patches $\mathbf{I}_{\text{HR}}$ from 800 images in the DIV2K [1] dataset and apply arbitrary M_i^{-1} to get corresponding LR samples $\mathbf{I}_{\text{LR}}$. We leverage a widely-used bicubic interpolation for the synthesis. A cropped version of the LR image $\mathbf{I}_{\text{LR-crop}}$ has a maximum size of 96×96, which is randomly acquired by ignoring *void* pixels. For evaluation, we follow the similar pipeline but center-crop one 384×384 patch from each DIV2K validation image, i.e., from '*0801.png*' to '*0900.png*.' 100 transformation matrices M_i is assigned

Variable(s)	Sampling distribution	Note
h_x, h_y	$\mathcal{U}(-0.25, 0.25)$	Random sheering
θ	$\mathcal{N}(0, 15^{\circ 2})$	Random rotation
s_x, s_y	$\mathcal{U}(0.35, 0.5)$	Random scaling
t_x	$\mathcal{U}(-0.75w, 0.125w)$	Random projection
t_x	$\mathcal{U}(-0.75h, 0.125h)$	
p_x	$\mathcal{U}(-0.6w, 0.6w)$	
p_y	$\mathcal{U}(-0.6h, 0.6h)$	

Table S1: **Random variable specifications for the transformation matrix.** $h \times w$ denotes the resolution of an HR patch or image.

(a) Training patches and transforms

(b) Validation patches and transforms

(c) Test images and transforms

Figure S1: **Illustration of the DIV2KW data splits.** We use the same images for validation and test, but their sizes and corresponding matrices are different.

to each image, and the cropped region $\mathbf{I}_{\text{LR-crop}}$ is also fixed to ensure reproducibility.

To fairly compare our SRWarp model against the conventional warping method, we use full-size images $\mathbf{I}_{\text{HR}}$ as test inputs. The smallest HR image in the DIV2K validation dataset has a resolution of 816×2040, and the largest one is 2040×2040. Since the transformation matrix M_i depends on the image resolution, each of the 100 test examples has assigned fixed warping parameters similar to the validation case. We also note that the test transformations are totally different from training and evaluation due to the size difference. Figure S1 demonstrates a visual comparison between the training, validation, and test splits.

S2. Details about the AWL

We concretely describe how the adaptive sampling coordinate in Section 3.2 of our main manuscript is calculated. A unit circle $x'^2 + y'^2 = 1$ on the target domain is mapped to a general ellipse E on the source coordinate by the affine transform $J = \begin{pmatrix}\mathbf{u}^\mathsf{T} & \mathbf{v}^\mathsf{T}\end{pmatrix}$. For simplicity, we will formulate the shape E as a rotated version of an axis-aligned ellipse, which can be described as follows:

$$\frac{(x\cos\omega + y\sin\omega)^2}{A^2} + \frac{(y\cos\omega - x\sin\omega)^2}{B^2} = 1, \quad \text{(S2)}$$

where A and B are lengths of principal axes, and ω denotes the rotated angle, respectively, as shown in Figure S2. By assuming that $\mathbf{u} = (u_x, u_y)$ and $\mathbf{v} = (v_x, v_y)$, we get the following equation:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{1}{D}\begin{pmatrix} v_y & -v_x \\ -u_y & u_x \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, \quad \text{(S3)}$$

where $D = \det J = u_x v_y - v_x u_y$. By substituting x' and y' in the equation of the unit circle with (S3), we get the following:

$$(v_y x - v_x y)^2 + (u_x y - u_y x)^2 = D^2. \quad \text{(S4)}$$

Since (S2) and (S4) are *equivalent*, the following three identites can be derived by developing the equations:

$$\begin{aligned} \frac{\cos^2\omega}{A^2} + \frac{\sin^2\omega}{B^2} &= \frac{u_y^2 + v_y^2}{D^2}, \\ \frac{\sin^2\omega}{A^2} + \frac{\cos^2\omega}{B^2} &= \frac{u_x^2 + v_x^2}{D^2}, \\ \frac{2\cos\omega\sin\omega}{A^2} - \frac{2\cos\omega\sin\omega}{B^2} &= -\frac{2u_x u_y + 2v_x v_y}{D^2}, \end{aligned} \quad \text{(S5)}$$

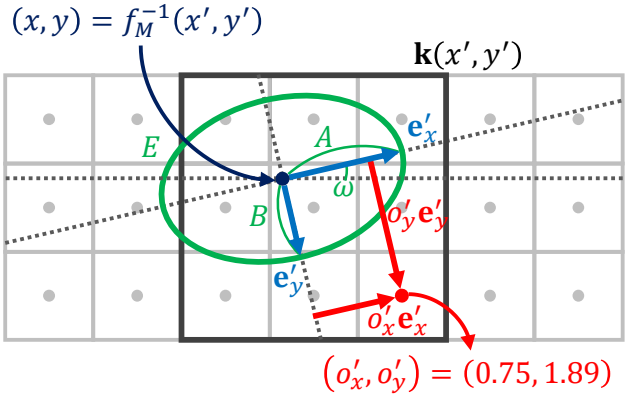


Figure S2: **Detailed illustration of the adaptive resampling coordinate.** We find the **green** ellipse E for each position (x', y') on the target coordinate. Best viewed with Figure 3 in our main manuscript. (o'_x, o'_y) denotes a relative offset vector, i.e., $\mathbf{o}'_{11}$, of the example point in **red** with respect to the reference (x, y) in **navy** on the adaptive grid.

where right-hand-side terms are constant. Using trigonometric identities, we can reduce (S5) as follows:

$$\begin{aligned} \frac{1}{A^2} + \frac{1}{B^2} &= \frac{\|J\|_F^2}{D^2}, \\ \cos 2\omega\left(\frac{1}{A^2} - \frac{1}{B^2}\right) &= \frac{u_y^2 + v_y^2 - u_x^2 - v_x^2}{D^2}, \\ \sin 2\omega\left(\frac{1}{A^2} - \frac{1}{B^2}\right) &= -\frac{2u_x u_y + 2v_x v_y}{D^2}, \end{aligned} \quad \text{(S6)}$$

where $\|J\|_F^2 = u_x^2 + v_x^2 + u_y^2 + v_y^2$. Using (S6), it is possible to represent three unknowns A, B, and ω with respect to J in a straighforward fashion as follows:

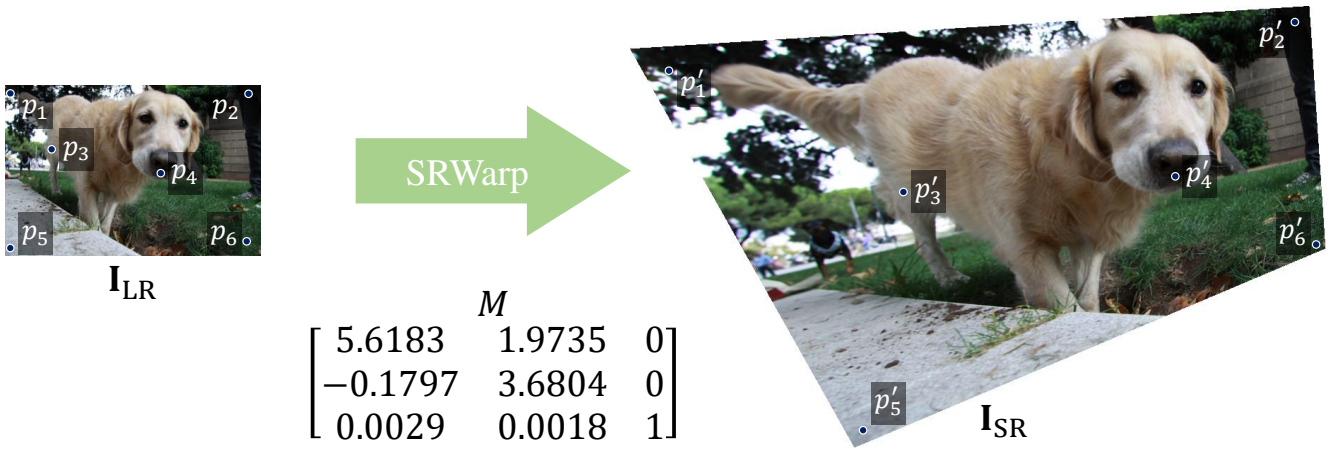
$$\begin{aligned} \omega &= \frac{1}{2}\arctan\frac{2u_x u_y + 2v_x v_y}{u_x^2 + v_x^2 - u_y^2 - v_y^2}, \\ A &= \sqrt{\frac{D^2\cos 2\omega}{\cos^2\omega\,\|J\|_F^2 - u_x^2 - v_x^2}}, \\ B &= \sqrt{\frac{D^2\cos 2\omega}{\cos^2\omega\,\|J\|_F^2 - u_y^2 - v_y^2}}. \end{aligned} \quad \text{(S7)}$$

Then, the basis vectors $\mathbf{e}'_x$ and $\mathbf{e}'_y$ of the adaptive grid can be calculated as follows:

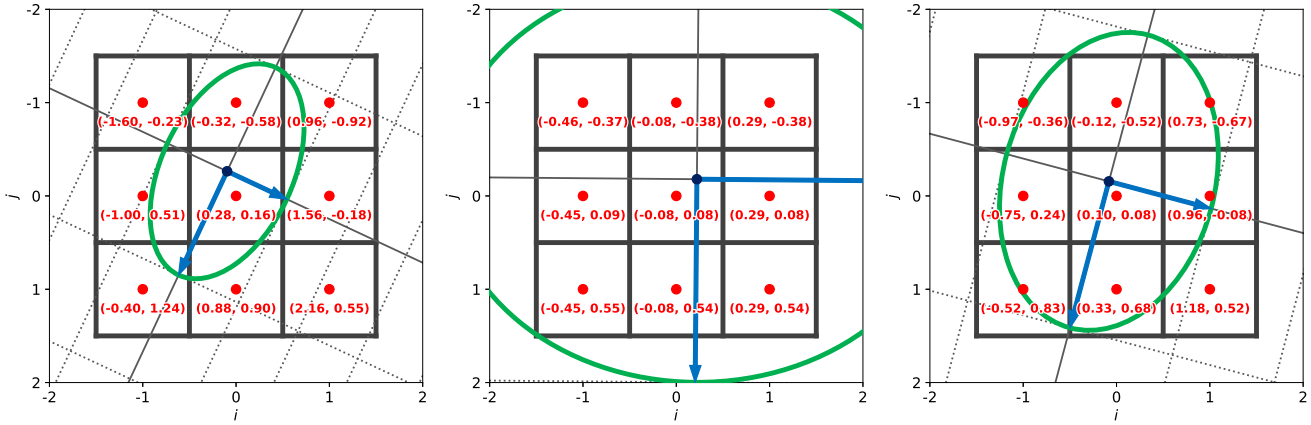
$$\begin{aligned} \mathbf{e}'_x &= (A\cos\omega, A\sin\omega), \\ \mathbf{e}'_y &= (-B\sin\omega, B\cos\omega). \end{aligned} \quad \text{(S8)}$$

Figure S2 shows a visualization of the ellipse E with its principal axes on the source domain.

A bold window in Figure S2 visualizes a $k \times k$ kernel support where the resampling weights $\mathbf{k}(x', y')$ is evaluated on. As described in (2) of our main manuscript,



(a) Correspondences between points in $\mathbf{I}_{\text{LR}}$ and $\mathbf{I}_{\text{SR}}$



(b) Resampling coordinate at p_1 (c) Resampling coordinate at p_2 (d) Resampling coordinate at p_3

(e) Resampling coordinate at p_4 (f) Resampling coordinate at p_5 (g) Resampling coordinate at p_6

Figure S3: **Visualization of the spatially-varying adaptive grid.** (a) We note that the correspondence is calculated as $Mp_i = p'_i$. (b−g) We display locally-varying sampling grids for the proposed AWL. The $\times 4$ feature $\mathbf{F}_{\times 4}$ is used as a source domain for visualization. A bold 3×3 grid shows a resampling kernel $\mathbf{k}\left(x', y'\right)$ on the source domain. We visualize the relative offsets in **red** with respect to the adaptive coordinate. Our kernel estimator $\mathcal{K}$ takes the 3×3 offset vectors and predicts appropriate resampling weights. Then, the output value $\mathbf{W}_{\times 4}\left(x', y'\right)$ is calculated by combining 3×3 points in **red** on the source domain with the corresponding kernel. The sampling grid tends to be shrunk if the feature is locally enlarged and vice versa.

we first locate $k \times k$ points in the window, i.e., $\mathcal{P} = \{p_{ij} = (\lfloor x \rceil + i, \lfloor y \rceil + j) \,|a \leq i, j \leq b\}$. For the case in Figure S2, we note that $a = -1$ and $b = 1$ and therefore $|\mathcal{P}| = 9$. Then, we calculate *relative* offsets of each point with respect to the reference (x, y), i.e., $\mathbf{o}_{ij} = p_{ij} - (x, y)$, on a regular grid. Finally, we find $k \times k$ many vectors $\mathbf{o}'_{ij}$, which is a representation of the offsets $\mathbf{o}_{ij}$ using the orthogonal basis $\{\mathbf{e}'_x, \mathbf{e}'_y\}$ as follows:

$$\mathbf{o}_{ij}^{'\mathsf{T}} = \begin{pmatrix} A^{-1} & 0 \\ 0 & B^{-1} \end{pmatrix} \begin{pmatrix} \cos\omega & \sin\omega \\ -\sin\omega & \cos\omega \end{pmatrix} \mathbf{o}_{ij}^{\mathsf{T}}. \tag{S9}$$

Using the calculated offset vector $\mathbf{o}_{ij}^{'\mathsf{T}}$, we rescale the original offsets and evaluate the kernel function as described in our main manuscript. We note that in traditional bicubic interpolation method, resampling basis is fixed to $\{(1, 0), (0, 1)\}$ across all output positions (x', y'). For the Adaptive configuration in Table 2 of our main manuscript, we use the conventional cubic spline to calculate the weight for each offset vector $\mathbf{o}_{ij}^{'\mathsf{T}}$. In the proposed AWL, we concatenate $k \times k$ many vectors and fed them to a learnable network $\mathcal{K}$. By doing so, a $k \times k$ resampling kernel $\mathbf{k}(x', y')$ can be calculated at once. We provide more detailed visualization of AWL in Section S3 and implementation of the kernel estimation module $\mathcal{K}$ in Section S4.

S3. Visualizing Spatially-Varying Property

We analyze how the proposed SRWarp operates in a spatially-varying manner. Figure S3 shows our adaptive resampling grids for different local regions. Specifically, when a local region is shrunk to a certain direction, the green ellipse is stretched to the particular axis. Then, points in such direction are pulled to the origin when calculating the kernel function and contribute more, i.e., have larger weights. The same thing happens oppositely in the case of locally enlarging distortions. Our SRWarp can reconstruct high-quality images without severe artifacts and aliasing by considering the spatially-varying coordinate system in the resampling process. Combined with the learnable kernel estimator $\mathcal{K}$, the adaptive grid can improve our SRWarp model without requiring *any* additional parameters.

In Figure S4, we demonstrate how the multiscale blending coefficients vary depending on different contents and local distortions. As shown in Figure S4c and Figure S4d, $\times 2$ and $\times 4$ features play an important role around edges. On the other hand, the activation of the $\times 1$ feature in Figure S4b is widely distributed across the scene for low-frequency structures. We note that our multiscale blending module places a high priority on fine-grained textures and details regardless of the feature scale s, showing that single-scale information may not be enough for the warping task. We also validate the effectiveness of the scale feature $\mathbf{S}$ in (7) of our main manuscript by ignoring $\mathbf{C}$ from the blending module. Since the $\times 4$ feature plays a critical role in upsampling a given image, it is equally weighted across the target domain, as shown in Figure S4g. In contrast, relative contributions of $\times 1$ and $\times 2$ features show spatially-varying behavior, demonstrating the importance of considering local distortion. Specifically, the $\times 1$ feature becomes less weighted with larger distortion (bottom right of Figure S4e), while it can provide meaningful information to our SRWarp model if the deformation is not significant (top left of Figure S4e).

S4. Detailed Model Architectures

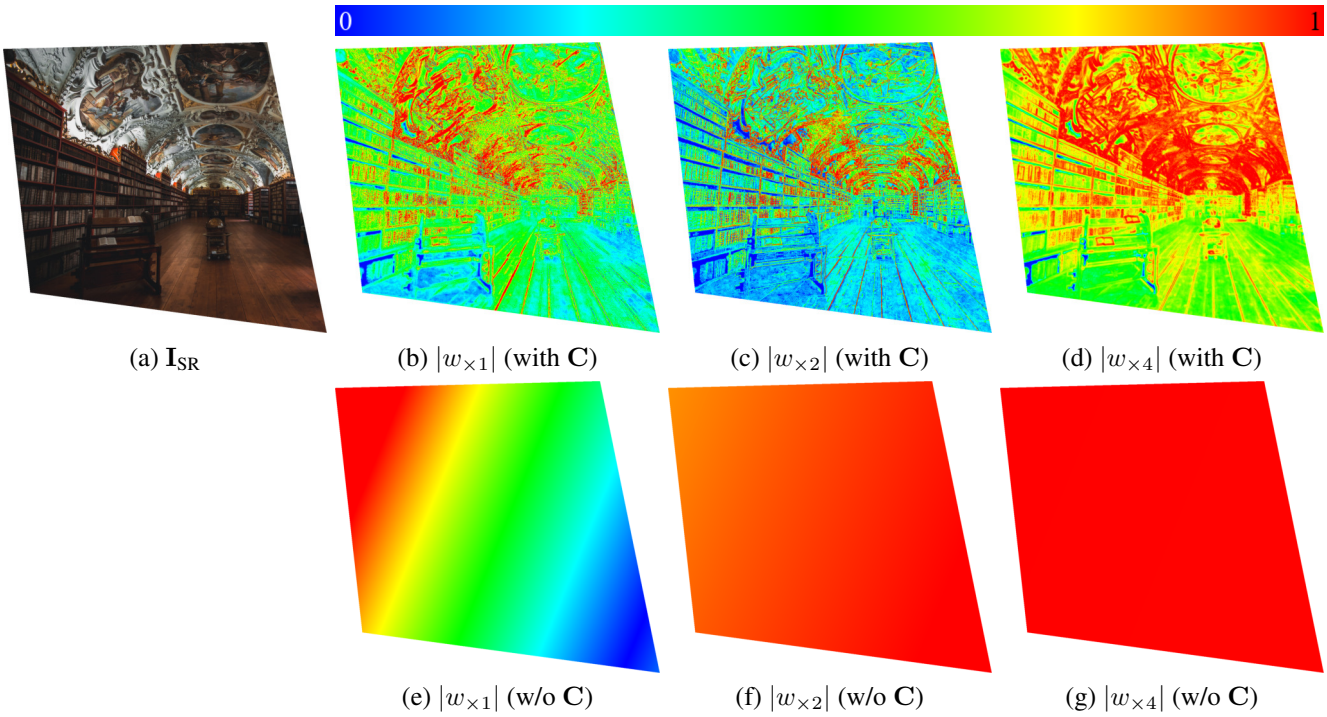
Multiscale feature extractor. Figure S5 demonstrates the modified MDSR [9] and MRDB backbone architectures we introduce for efficient multiscale respresentation. Since the proposed SRWarp model is designed to handle the SR task of arbitrary scales and shapes, we remove the scale-specific branch in front of the original MDSR structure. We also replace $\times 2$, $\times 3$, and $\times 4$ upscaling modules to $\times 1$, $\times 2$, and $\times 4$ feature extractors. Following the baseline model by Lim *et al*. [9], we adopt 16 ResBlocks with 64 channels each for the main branch. Our MRDB model is a multiscale version of the state-of-the-art RRDB [14] network. We change the last upscaling layer to the proposed scale-specific feature extractors, similar to the modified MDSR model.

SRWarp architecture. Figure 4 in our main manuscript illustrates the overall pipeline of the proposed SRWarp method. We construct the main modules with residual blocks [8, 9] and skip connections [7] for stable and efficient training. Figure S6, Figure S7, and Figure S8 show how the actual implementations of our kernel estimator, multiscale blending, and reconstruction modules are, respectively.

Architectures for the ablation study. Figure S9 describes detailed model structures for Table 1 in the main manuscript. We note that $\times 1$ feature $\mathbf{F}_{\times 1}$ is fed to the AWL since the module replaces conventional upsampling layers in SR networks.

S5. Detailed Training Configurations

For stable convergence, we pre-train the SRWarp network on the fractional-scale DIV2K dataset with a mini-batch size of $N = 16$. The model is updated for 600 epochs, where each epoch consists of 1,000 iterations. We set the initial learning rate to 10^{-4} and halve it after 150, 300, and 450 epochs. Then, we use a mini-batch of size $N = 4$ to fine-tune our SRWarp model on the DIV2KW dataset. The network is updated for 200 epochs with an initial learning rate of 5×10^{-5}, which is halved after 100, 150, and 175 epochs. For the small modified MDSR architecture, the training takes about 12 hours on a single RTX 2080 Ti GPU. With the larger MRDB backbone, we optimize the model for about 60 hours on two RTX 2080 Ti GPUs.



(a) $\mathbf{I}_{\text{SR}}$ (b) $|w_{\times 1}|$ (with $\mathbf{C}$) (c) $|w_{\times 2}|$ (with $\mathbf{C}$) (d) $|w_{\times 4}|$ (with $\mathbf{C}$)

(e) $|w_{\times 1}|$ (w/o $\mathbf{C}$) (f) $|w_{\times 2}|$ (w/o $\mathbf{C}$) (g) $|w_{\times 4}|$ (w/o $\mathbf{C}$)

Figure S4: **Spatially-varying property of multi-scale blending weights.** We normalize the map $w_{\times s}$ *within* each sample for better visualization. (e−g) We do not use the content feature $\mathbf{C}$ to estimate the weights $w_{\times s}$ and observe how the scale feature $\mathbf{S}$ affects the blending coefficients. The weights may not represent the relative importance of different scales s as the features $\mathbf{W}_{\times s}$ are not normalized. Instead, the values indicate which regions are weighted more in each different scale.

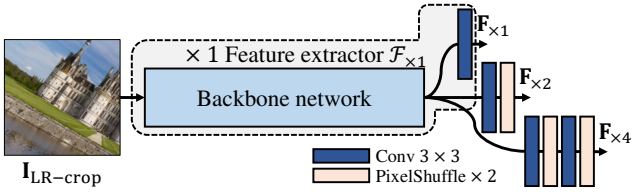


Figure S5: **Multiscale feature extractor $\mathcal{F}_{\times s}$ in our SRWarp model.** For the modified MDSR structure, we adopt the MDSR [9] model without scale-specific branches as the backbone network. In the MRDB architecture, the state-of-the-art RRDB [14] model without upsampling module is used for the backbone. We note that our feature extractors $\mathcal{F}_{\times s}$ share many parts in common across different scales s for efficient computation [9]. The $\times 1$ feature extractor $\mathcal{F}_{\times 1}$ is emphasized in the figure as an example. Each of the scale-specific features $\mathbf{F}_{\times s}$ has 64 channels.

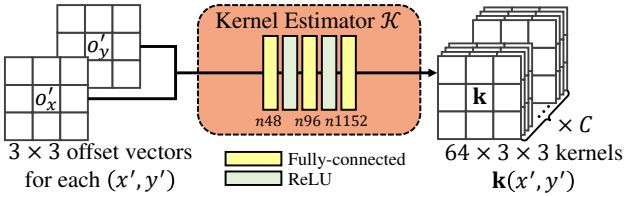


Figure S6: **Structure of the kernel estimator $\mathcal{K}$ in our adaptive warping layer.** The module takes 18 input values for each point (x', y') in the target coordinate. n describes the output dimension of fully-connected layers. We use $C = 64$ channels for dynamic kernels.

S6. Efficiency of the Proposed Method

Table S2 compares the actual runtime and peak memory usage of the proposed SRWarp against the meta-upscaling approaches [4] on the arbitrary-scale SR task. We use a single RTX 2080 Ti GPU for all executions to fairly compare different methods in a unified environment. The SRWarp model requires much lower memory than Meta-EDSR and Meta-RDN while achieving comparable performances, as shown in Table 4 in our main manuscript. Notably, the proposed method is about 2.5 times faster than the Meta-RDN model while achieving better performance.

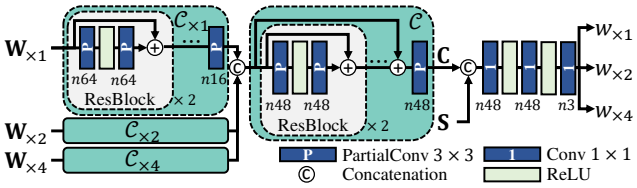


Figure S7: **Weight prediction in our multiscale blending module.** We omit the input mask $\mathbf{m}$ for the partial convolutional layers for simplicity. The blended feature $\mathbf{W}_{\text{blend}}$ is calculated from the warped features $\mathbf{W}_{\times s}$ and their corresponding weights $w_{\times s}$ as described in (8). Best viewed with Section 3.3 in our main manuscript.

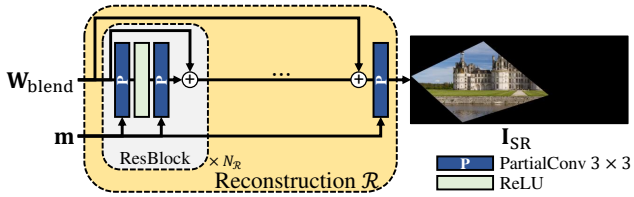
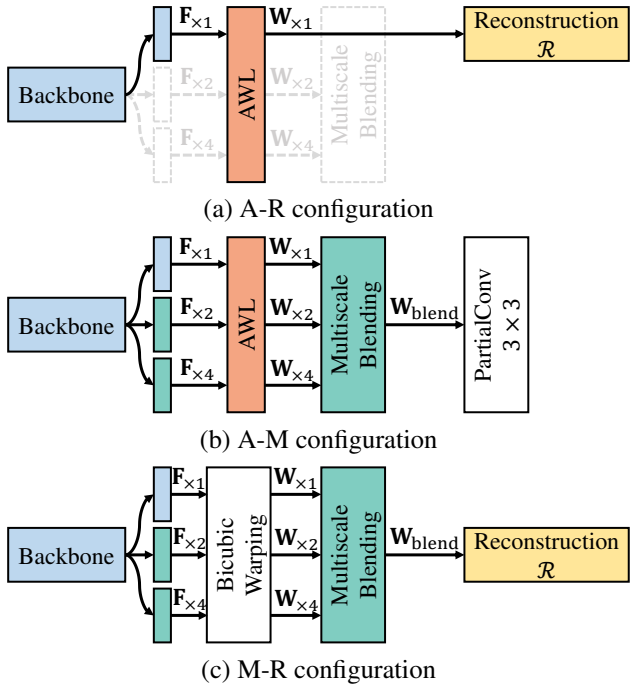


Figure S8: **Structure of the reconstruction module $\mathcal{R}$.** We use $N_{\mathcal{R}} = 5$ many ResBlocks wich 64 output channels.



(a) A-R configuration

(b) A-M configuration

(c) M-R configuration

Figure S9: **Examples of the SRWarp architectures $\mathcal{S}$ for the ablation study.** Best viewed with Figure 4 and Table 1 in our main manuscript. (c) Bicubic warping denotes a differentiable warping layer with the conventional bicubic interpolation, without our adaptive resampling grid and kernel estimator $\mathcal{K}$.

Method	Peak Mem. (GB)	Runtime (ms)
Meta-EDSR [4]	4.96	218
Meta-RDN [4]	5.15	253
SRWarp (MRDB)	3.08	155

Table S2: **Efficiency comparison between the proposed method and meta-upscaling.** The evaluation is done on the fractional-scale B100 [10] dataset with a scale factor of $\times 3$. The runtime is measured over 100 test images in a unified environment and averaged *excluding* initialization, I/O, and the other overheads. We also note that the peak memory usage does not correspond to the exact amount of memory each model consumes due to GPU-level optimization, caching, and many other practical reasons.

S7. Additional Results

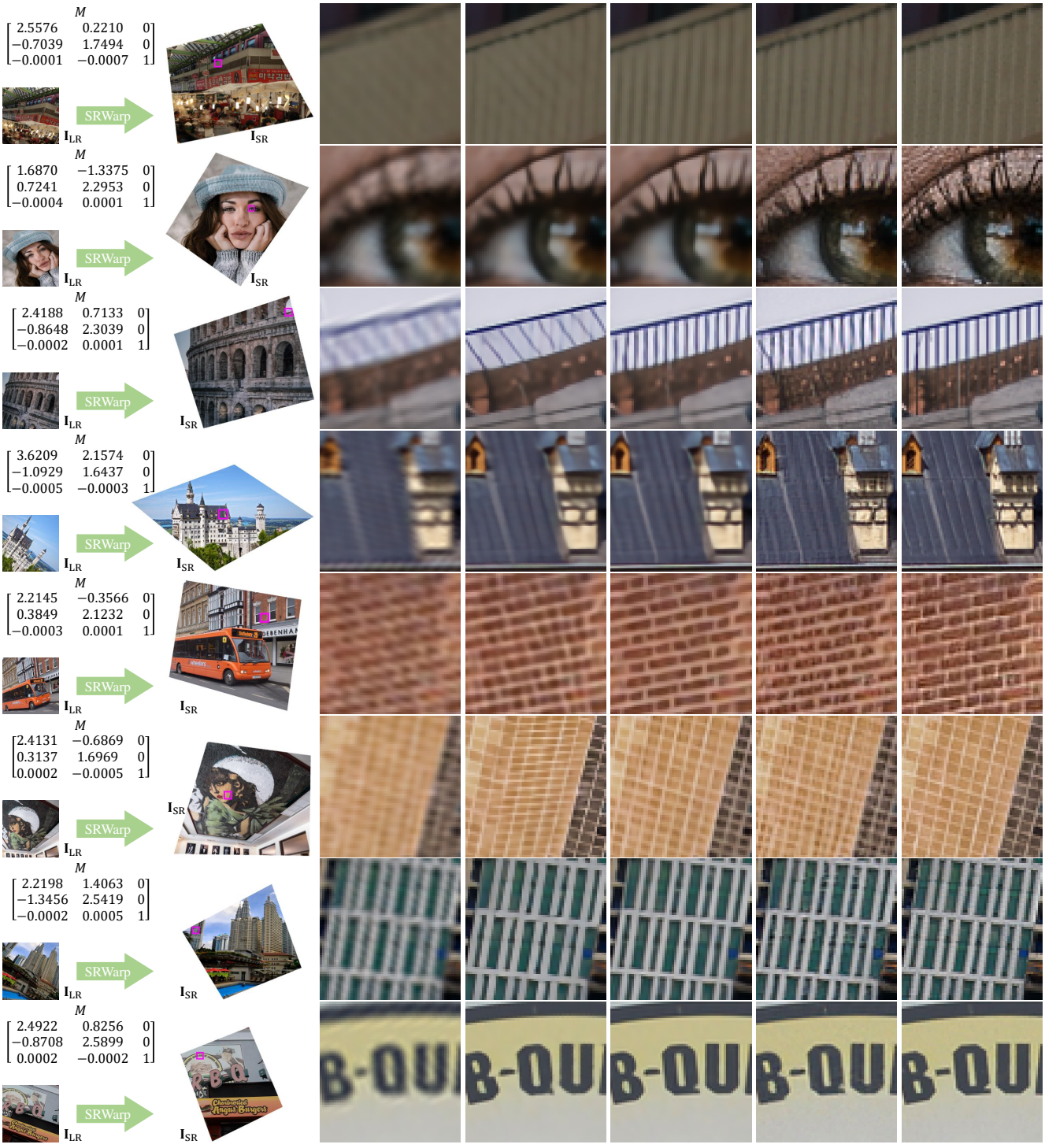
Figure S10 shows additional qualitative results from the proposed warping method. Since we optimize the widely-used L_1 objective function to train our SRWarp, it is straightforward to introduce perceptual [6] and adversarial [3, 12] loss terms to acquire photo-realistic [8, 14] warped results (SRWarp$^+$). We first construct a discriminator network $\mathcal{D}$ following Ledig *et al.* [8], while the fully-connected layers at the end of the original model are replaced to 1×1 convolutions [5]. The spectral normalization [11] is also applied for stable training. We update the discriminator network by the corresponding loss term $\mathcal{L}_{\text{dis}}$, which is defined as follows:

$$\mathcal{L}_{\text{dis}} = -\log \mathcal{D}\left(\mathbf{I}_{\text{HR-crop}}\right) - \log\left(1 - \mathcal{D}\left(\mathbf{I}_{\text{SR-crop}}\right)\right), \quad \text{(S10)}$$

where we crop 96×96 patches $\mathbf{I}_{\text{HR-crop}}$ and $\mathbf{I}_{\text{SR-crop}}$ from the original images $\mathbf{I}_{\text{HR}}$ and $\mathbf{I}_{\text{SR}}$ to avoid void regions to be fed to the following discriminator model. We omit averaging over the batch dimension for simplicity. Then, we alternately optimize the perceptual objective function $\mathcal{L}_{\text{per}}$ for the SRWarp model which is defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{per}} = \alpha \frac{1}{\|\mathbf{m}\|_0} \left\|\mathbf{m} \odot \left(\mathbf{I}_{\text{SR}} - \mathbf{I}_{\text{HR}}\right)\right\|_1 \\ + \left\|\mathcal{V}_{54}\left(\mathbf{I}_{\text{SR-crop}}\right) - \mathcal{V}_{54}\left(\mathbf{I}_{\text{HR-crop}}\right)\right\|_1 \\ - \beta \log \mathcal{D}\left(\mathbf{I}_{\text{SR-crop}}\right),\end{aligned} \quad \text{(S11)}$$

where $\mathcal{V}_{54}$ is a pre-trained VGG-19 [13, 6] network up to `conv5_4` layer, $\alpha = 0.002$ and $\beta = 0.01$ are hyperparameters, respectively. We note that the cropped images $\mathbf{I}_{\text{SR-crop}}$ and $\mathbf{I}_{\text{HR-crop}}$ are aligned so that the VGG-loss term can be used to improve the perceptual quality of the SR results.



(a) $\mathbf{I}_{LR}$, $\mathbf{I}_{SR}$, and M (b) cv2 (c) RRDB (d) **SRWarp** (e) **SRWarp**$^{+}$ (f) GT

Figure S10: **More qualitative warping results on various images.** RRDB denotes the method which combines the state-of-the-art $\times 4$ RRDB [14] model with the bicubic warping algorithm from OpenCV [2]. **Top four rows:** To show the generalization ability of the proposed method, we choose random DIV2K images ('*0831.png*,' '*0855.png*,' '*0864.png*,' and '*0887.png*') and transformation matrices to test the proposed SRWarp method. **Bottom four rows:** We construct the novel Flickr2KW$_{\text{Test}}$ dataset from the high-quality Flickr2K [9] images, following the same pipeline of the DIV2KW$_{\text{Test}}$. Patches are cropped from the Flickr2KW$_{\text{Test}}$ '*000004.png*,' '*000957.png*,' '*001219.png*', and '*001290.png*,' respectively.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017. 1

[2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 7

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014. 6

[4] Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-SR: A magnification-arbitrary network for super-resolution. In *CVPR*, 2019. 5, 6

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-To-Image translation with conditional adversarial networks. In *CVPR*, 2017. 6

[6] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 6

[7] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 4

[8] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 4, 6

[9] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 4, 5, 7

[10] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 6

[11] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 6

[12] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv*, 2015. 6

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[14] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: enhanced super-resolution generative adversarial networks. In *ECCV Workshops*, 2018. 4, 5, 6, 7