# Supplementary File for: "Image Harmonization with Transformer"

Zonghui Guo[1] Dongsheng Guo[1] Haiyong Zheng[1,*] Zhaorui Gu[1] Bing Zheng[1,2] Junyu Dong[3]
[1]Underwater Vision Lab (http://ouc.ai), Ocean University of China
[2]Sanya Oceanographic Institution, Ocean University of China
[3]College of Computer Science and Technology, Ocean University of China

We introduce more details about network architectures and additional experiments of our paper.

## A. Evaluation Metrics

In addition to MSE and PSNR, we also report foreground MSE (fMSE) and foreground PSNR (fPSNR) to measure how well the foreground is harmonized. MSE and PSNR essentially evaluate harmonization performance over all pixels across the dataset (dataset-level), while fMSE and fPSNR measure the harmonization over each single image (with different sizes of foreground) averaging on the dataset (image-level). We argue that image-level fMSE and fPSNR are more suitable to evaluate the harmonization generalization ability since many pixels (background) are unchanged and the sizes of foreground are different in terms of each image. Given the real image $\mathbf{H}$ and the harmonized image $\hat{\mathbf{H}}$, we provide the details of these four metrics as follows.

### A.1. MSE vs. fMSE

We compute MSE by:

$$\begin{aligned}\text{MSE}(\hat{\mathbf{H}}, \mathbf{H}) &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{3K}\sum_{k=1}^{K}\left\|\hat{\mathbf{H}}_k^{(n)} - \mathbf{H}_k^{(n)}\right\|_2\right) \\ &= \frac{1}{3KN}\sum_{n=1}^{N}\sum_{k=1}^{K}\left\|\hat{\mathbf{H}}_k^{(n)} - \mathbf{H}_k^{(n)}\right\|_2, \end{aligned} \quad (1)$$

where $K$ is the pixel number of image ($k$ is the pixel index), $N$ is the image number of dataset ($n$ is the image index), and 3 means three RGB channels of image.

And we compute our fMSE by:

$$\text{fMSE}(\hat{\mathbf{H}}, \mathbf{H}) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{3K_{fg}^{(n)}}\sum_{k=1}^{K}\left\|\hat{\mathbf{H}}_k^{(n)}\mathbf{M}_k^{(n)} - \mathbf{H}_k^{(n)}\mathbf{M}_k^{(n)}\right\|_2\right), \quad (2)$$

where $K_{fg}^{(n)}$ is the foreground pixel number of $n$-th image, and $\mathbf{M}$ denotes the foreground mask.

### A.2. PSNR vs. fPSNR

We compute PSNR by:

$$\text{PSNR}(\hat{\mathbf{H}}, \mathbf{H}) = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{\text{MSE}(\hat{\mathbf{H}}, \mathbf{H})}\right), \quad (3)$$

where $MAX_I$ is 255.

And we compute our fPSNR by:

$$\text{fPSNR}(\hat{\mathbf{H}}, \mathbf{H}) = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{\text{fMSE}(\hat{\mathbf{H}}, \mathbf{H})}\right). \quad (4)$$

## B. Operator Symbols

Symbols of used operators are listed as follows:

- Conv($c_{in}$, $c_{out}$, $k$, $s$, $p$): convolution with $c_{in}$ input channels, $c_{out}$ output channels, kernel size $k$, stride $s$, and padding $p$.
- ConvTranspose2d($c_{in}$, $c_{out}$, $k$, $s$, $p$): convolution with $c_{in}$ input channels, $c_{out}$ output channels, kernel size $k$, stride $s$, and padding $p$.
- Linear($f_{in}$, $f_{out}$): linear transformation with $f_{in}$ input features and $f_{out}$ output features.
- ResBlock($c_{in}$, $c_{out}$, $k$, $s$, $p$): residual block [4] with $c_{in}$ input channels, $c_{out}$ output channels, kernel size $k$, stride $s$, and padding $p$.
- Upsample($s$): nearest-neighbor upsampling with a scale factor of $s$.
- IN($n$): instance normalization [10] with $n$ dimensions.
- LN($n$): layer normalization [1] with $n$ dimensions.
- LReLU($\alpha$): Leaky ReLU [7] with a negative slope of $\alpha$.

*Corresponding author: Haiyong Zheng (zhenghaiyong@ouc.edu.cn).
This work was supported by the National Natural Science Foundation of China under Grant Nos. 61771440 and 41776113.

- AdaIN: Adaptive Instance Normalization (AdaIN) [5] layer with residual block.

- $TR(h, l, d_{token}, d_{ffn})$: transformer encoder-decoder with $h$ attention headers, $l$ layers, token embedding dimension $d_{token}$, and feed forward network dimension $d_{ffn}$. $TRE$ denotes transformer encoder and $TRD$ denotes transformer decoder.

## C. Harmonization Transformer

### C.1. Baselines and HT

The network architectures of our baseline encoder-decoder U-Net [9] (E-D U-Net), baseline encoder-decoder CNN (E-D CNN), and our harmonization Transformer (HT) are shown in Table A, Table B, and Table C, respectively.

### C.2. Transformer Input

The network architecture for analyzing Transformer image input on harmonization is listed in Table D.

### C.3. Transformer Encoder/Decoder

The network architecture for analyzing Transformer Encoder/Decoder on harmonization is listed in Table E.

### C.4. Transformer Head and Layer

The network architecture for analyzing Transformer head and layer on harmonization is listed in Table F.

## D. Disentangled Harmonization Transformer

### D.1. D-HC

The network architecture of our disentangled harmonization CNN (D-HC) is listed in Table G.

### D.2. D-HT

The network architecture of our disentangled harmonization Transformer (D-HT) is listed in Table H.

Additional qualitative comparison results of image harmonization are shown in Figures A and B. Additional harmonized results with normal masks and inverted masks are shown in Figure C. Additional light latent representation results by changing light latent code are shown in Figure D. And additional visual results transferring the light from one source image to another target image are shown in Figures E and F.

## E. Results on Real Composite Images

All visual comparison results of different methods to harmonize 99 real composite images are shown in Figures G–Q.

## F. Image Inpainting

Additional qualitative comparison results of image inpainting are shown in Figure R.

## G. Image Enhancement

Additional qualitative comparison results of image enhancement are shown in Figure S.

## References


[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Wenyan Cong, Jianfu Zhang, Li Niu, Liu Liu, Zhixin Ling, Weiyuan Li, and Liqing Zhang. DoveNet: Deep image harmonization via domain verification. In *CVPR*, pages 8394–8403, 2020.

[3] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A Efros. What makes paris look like paris? *ACM TOG*, 31(4):101, 2012.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[5] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pages 1501–1510, 2017.

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738, 2015.

[7] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICMLW*, pages 1–6, 2013.

[8] Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. DeepLPF: Deep local parametric filters for image enhancement. In *CVPR*, pages 12826–12835, 2020.

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015.

[10] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, pages 6924–6932, 2017.

| Encoder-decoder | Connection | Output Size |
|---|---|---|
| Conv(4, 64, 4, 2, 1) | 1 | 128 |
| LReLU(0.2)<br>Conv(64, 128, 4, 2, 1)<br>IN(128) | 2 | 64 |
| LReLU(0.2)<br>Conv(128, 256, 4, 2, 1)<br>IN(256) | 3 | 32 |
| LReLU(0.2)<br>Conv(256, 512, 4, 2, 1)<br>IN(512) | 4 | 16 |
| LReLU(0.2)<br>Conv(512, 512, 4, 2, 1)<br>IN(512) | 5 | 8 |
| LReLU(0.2)<br>Conv(512, 512, 4, 2, 1)<br>IN(512) | 6 | 4 |
| LReLU(0.2)<br>Conv(512, 512, 4, 2, 1)<br>IN(512) | 7 | 2 |
| LReLU(0.2)<br>Conv(512, 512, 4, 2, 1) | | 1 |
| ReLU<br>ConvTranspose2d(512, 512, 4, 2, 1)<br>IN(512) | | 2 |
| ReLU<br>ConvTranspose2d(1024, 512, 4, 2, 1)<br>IN(512)<br>Dropout(0.5) | 7 | 4 |
| ReLU<br>ConvTranspose2d(1024, 512, 4, 2, 1)<br>IN(512)<br>Dropout(0.5) | 6 | 8 |
| ReLU<br>ConvTranspose2d(1024, 512, 4, 2, 1)<br>IN(512)<br>Dropout(0.5) | 5 | 16 |
| ReLU<br>ConvTranspose2d(1024, 256, 4, 2, 1)<br>IN(256) | 4 | 32 |
| ReLU<br>ConvTranspose2d(512, 128, 4, 2, 1)<br>IN(128) | 3 | 64 |
| ReLU<br>ConvTranspose2d(256, 64, 4, 2, 1)<br>IN(64) | 2 | 128 |
| ReLU<br>ConvTranspose2d(128, 3, 4, 2, 1)<br>Tanh$\rightarrow output$ | 1 | 256 |

Table A. Network architecture of our baseline encoder-decoder U-Net (E-D U-Net).

| Encoder | Output Size |
|---|---|
| Conv(4, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |
| **Bottleneck** | **Output Size** |
| ResBlock(256, 256, 3, 1, 1) $\times 6$ | 64 |
| **Decoder** | **Output Size** |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table B. Network architecture of our baseline encoder-decoder CNN (E-D CNN).

| Encoder | Output Size |
|---|---|
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |
| **Bottleneck** | **Token Number** |
| $TRE$(2, 9, 256, 512) | 4096 |
| **Decoder** | **Output Size** |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table C. Network architecture of our harmonization Transformer (HT).

| Embedding | $d$ |
|---|---|
| FC: | |
| Linear(8*8*4, 256) | 256 |
| Conv: | |
| Conv(4, 256, 8, $s$, 0) | 256 |
| MLP: | |
| Linear(8*8*4, 512) + LN(512) + LReLU(0.2) | 512 |
| Linear(512, 256) + LN(256) + LReLU(0.2) | 256 |
| CNN: | |
| ($T = N$): | |
| Conv(4, 64, 4, 2, 0) + LN(64) + LReLU(0.2) | 64 |
| Conv(64, 128, 4, 2, 0) + LN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 0) + LN(256) + LReLU(0.2) | 256 |
| ($T \approx 4N$): | |
| Conv(4, 128, 4, 2, 0) + LN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 0) + LN(256) + LReLU(0.2) | 256 |
| ($T \approx 16N$): | |
| Conv(4, 256, 4, 2, 0) + LN(256) + LReLU(0.2) | 256 |

| Transformer | Token Number |
|---|---|
| $TRE(1, 3, 256, 512)$ | $T$ |

| Decoder | Output Size |
|---|---|
| ConvTranspose(256, 256, 8, $s$, 0) | 256 |
| IN(256) + LReLU(0.2) | 256 |
| Conv(256, 3, 3, 1, 1) + Tanh $\rightarrow output$ | 256 |

Table D. Network architecture for analyzing Transformer image input. $d$ means output dimensions, $T$ means token number, $N = 1024$, and $s = \{2, 4, 8\}$.

| Encoder | Output Size |
|---|---|
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |

| Bottleneck | Token Number |
|---|---|
| $TRE(1, 3, 256, 512)$ | 4096 |
| $TRD(1, l, 256, 512)$ | 4096 |

| Decoder | Output Size |
|---|---|
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table E. Network architecture for analyzing Transformer Encoder and Decoder.

| Encoder | Output Size |
|---|---|
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |

| Bottleneck | Token Number |
|---|---|
| $TRE(h, l, 256, 512)$ | 4096 |

| Decoder | Output Size |
|---|---|
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table F. Network architecture for analyzing Transformer head and layer.

| **Pseudo-reflectance** | |
|---|---|
| Encoder | Output Size |
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |
| Bottleneck | Output Size |
| ResBlock(256, 256, 3, 1, 1) $\times 6$ | 64 |
| Decoder | Output Size |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |
| **Pseudo-illumination** | |
| Encoder | Output Size |
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |
| Bottleneck | Output Size |
| ResBlock(256, 256, 3, 1, 1)+AdaIN $\times 6$ | 64 |
| Decoder | Output Size |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table G. Network architecture of our disentangled harmonization CNN (D-HC).

| **Pseudo-reflectance** | |
|---|---|
| Encoder | Output Size |
| Conv(3, 64, 7, 1, 3) + IN(64) + LReLU(0.2) | 256 |
| Conv(64, 128, 4, 2, 1) + IN(128) + LReLU(0.2) | 128 |
| Conv(128, 256, 4, 2, 1) + IN(256) + LReLU(0.2) | 64 |
| Bottleneck | Token Number |
| $TRE$(2, 9, 256, 512) | 4096 |
| Decoder | Output Size |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |
| **Pseudo-illumination** | |
| Encoder | Output size |
| Linear(8*8*3, 256) | 256 |
| Bottleneck | Token Number |
| $TRE_l$(2, 9, 256, 512) | 1024 |
| $TRD_l$(2, 9, 256, 512) | 27 |
| $TRD_i$(2, 9, 256, 512) | 4096 |
| Decoder | Output Size |
| Upsample(2) | 128 |
| Conv(256, 128, 3, 1, 1) + LN(128) + LReLU(0.2) | 128 |
| Upsample(2) | 256 |
| Conv(128, 64, 3, 1, 1) + LN(64) + LReLU(0.2) | 256 |
| Conv(64, 3, 7, 1, 3) + Tanh $\rightarrow output$ | 256 |

Table H. Network architecture of our disentangled harmonization Transformer (D-HT).

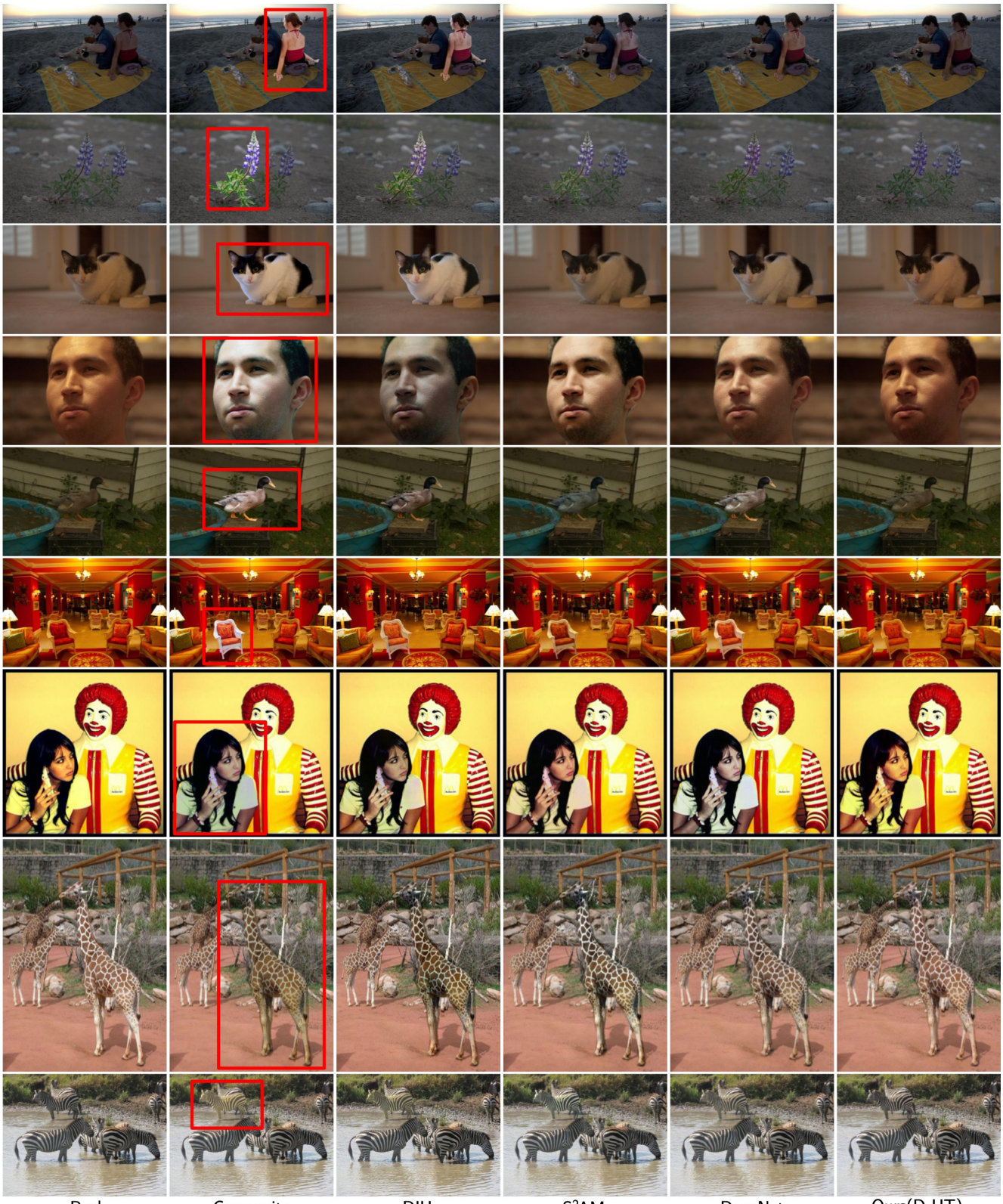


Figure A. Qualitative comparison across HCOCO and HAdobe5k sub-datasets of iHarmony4 [2]. Red boxes in composite images mark foreground.

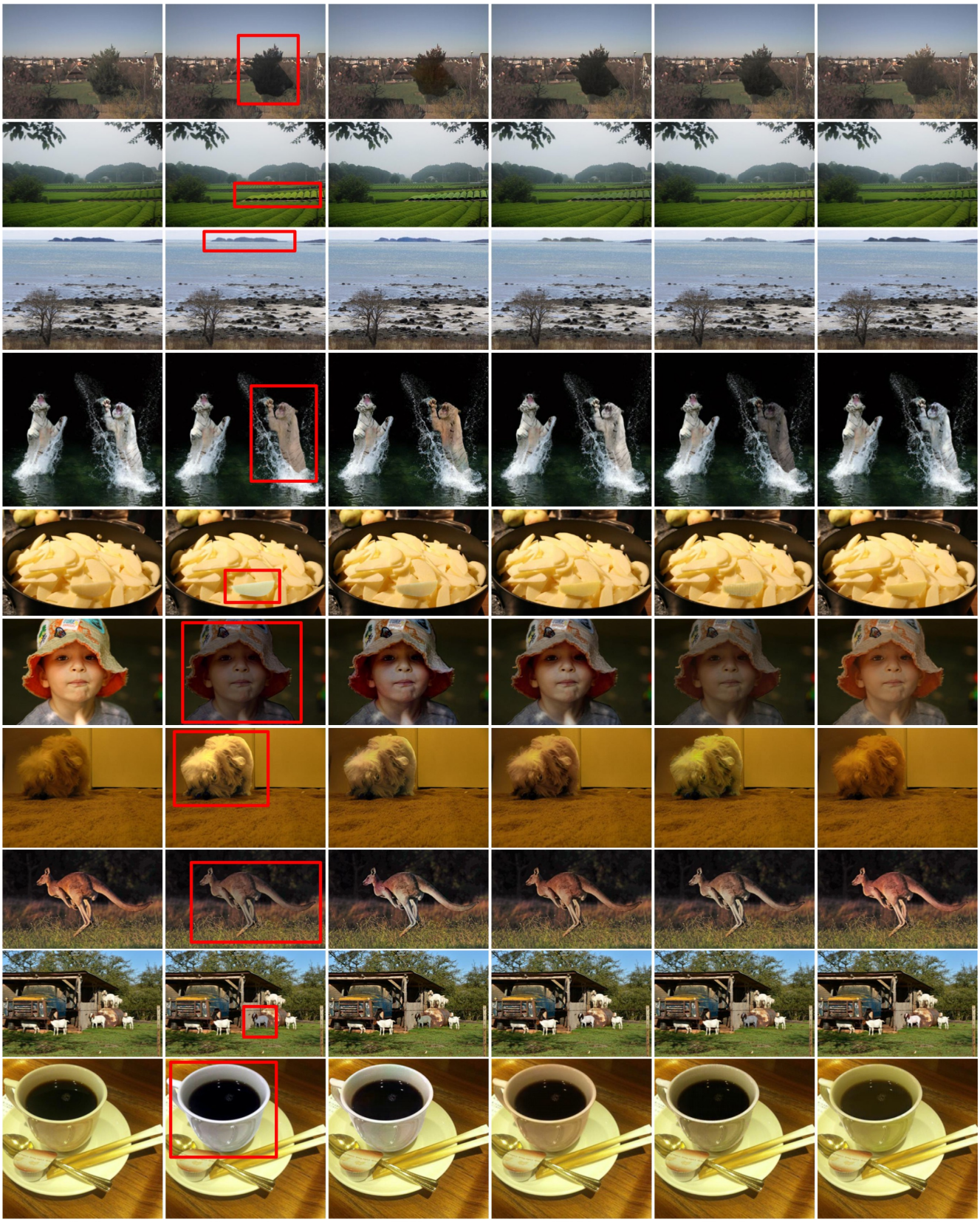


Figure B. Qualitative comparison across Hday2night and HFlickr sub-datasets of iHarmony4 [2]. Red boxes in composite images mark foreground.

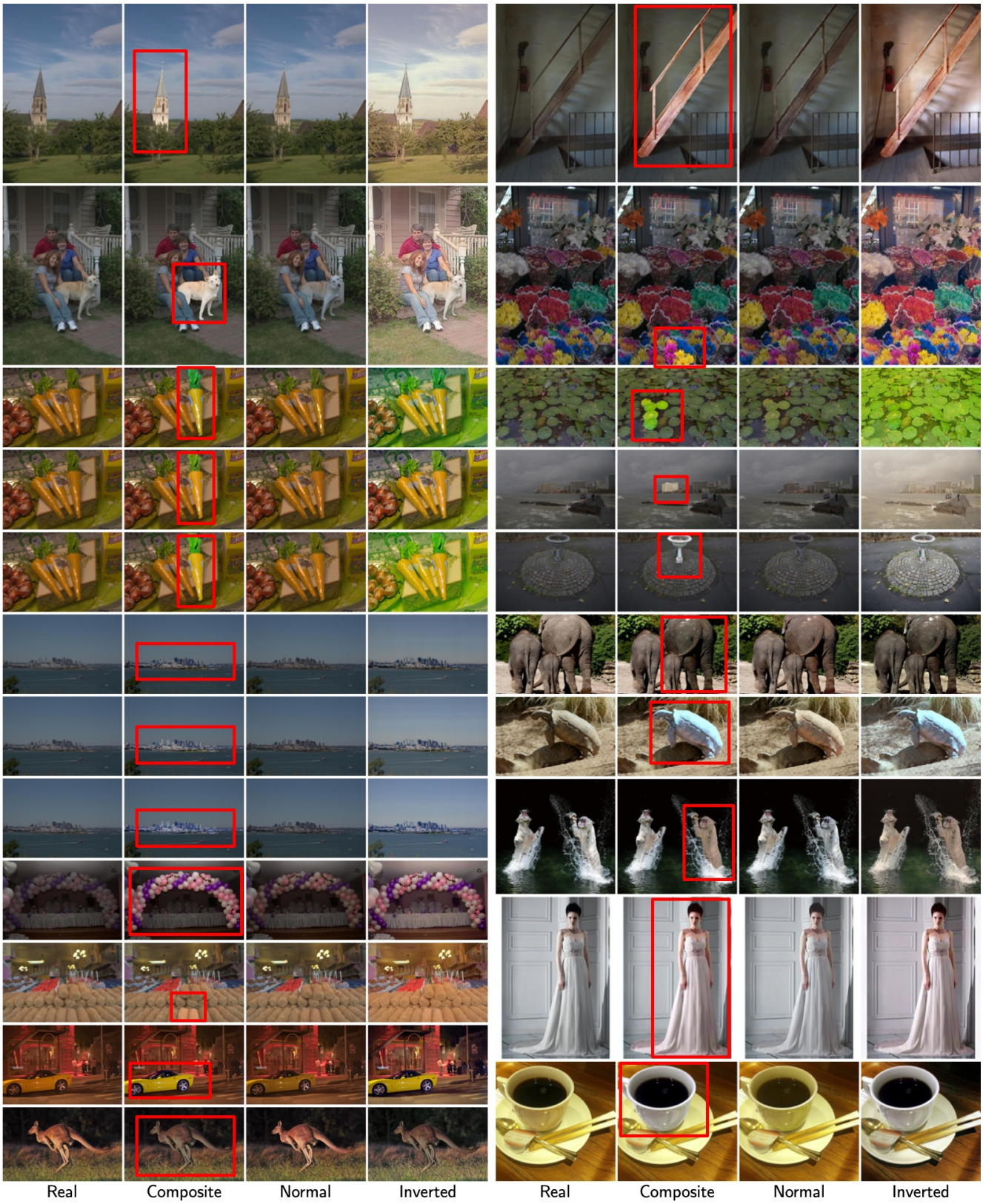


Figure C. Additional qualitative comparison results of image harmonization with normal masks and inverted masks. Red boxes in composite images mark foreground.

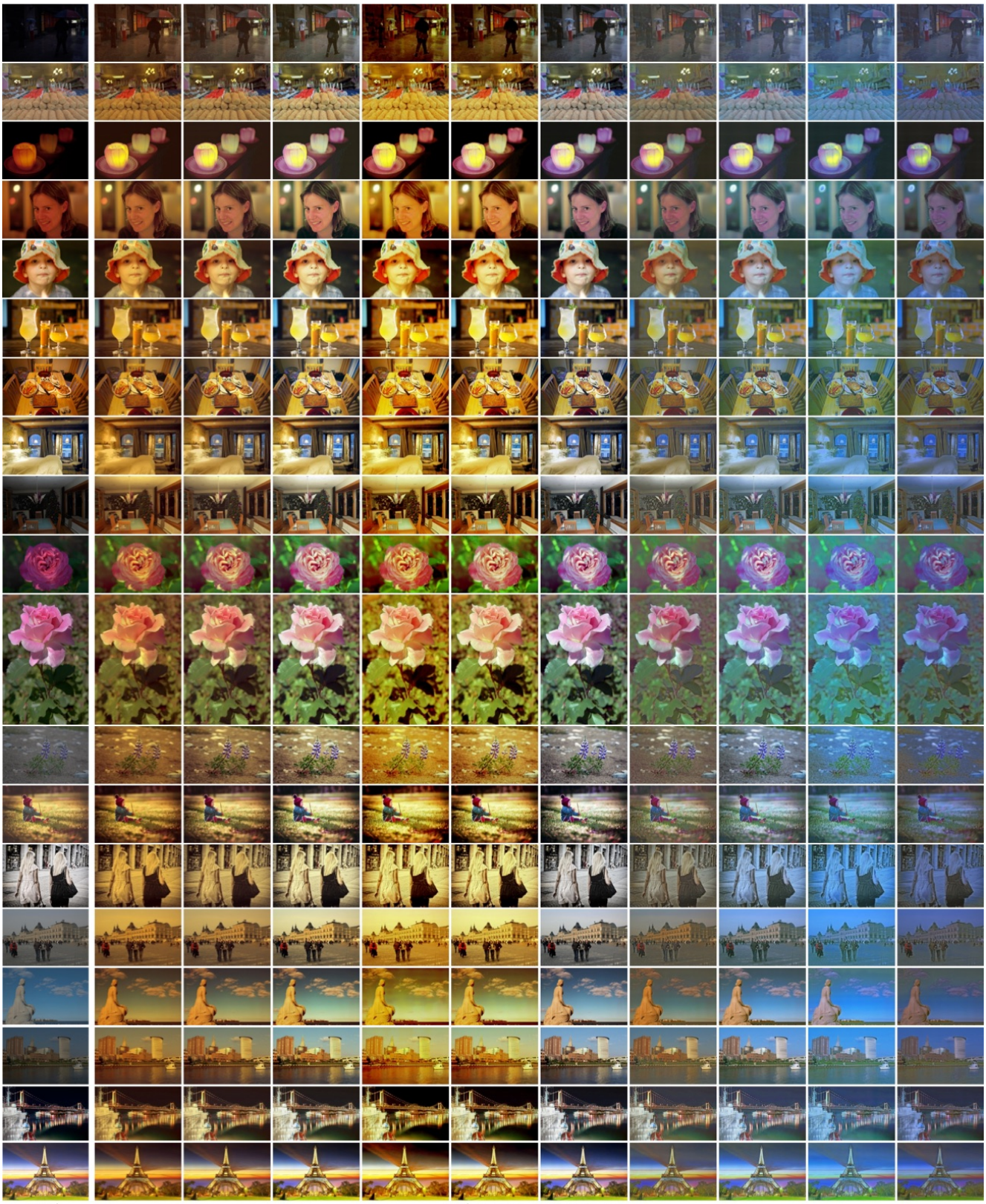


Figure D. Additional qualitative light latent representation results by changing light latent code.


Figure E. Additional qualitative results transferring the light from one source image to another target image.

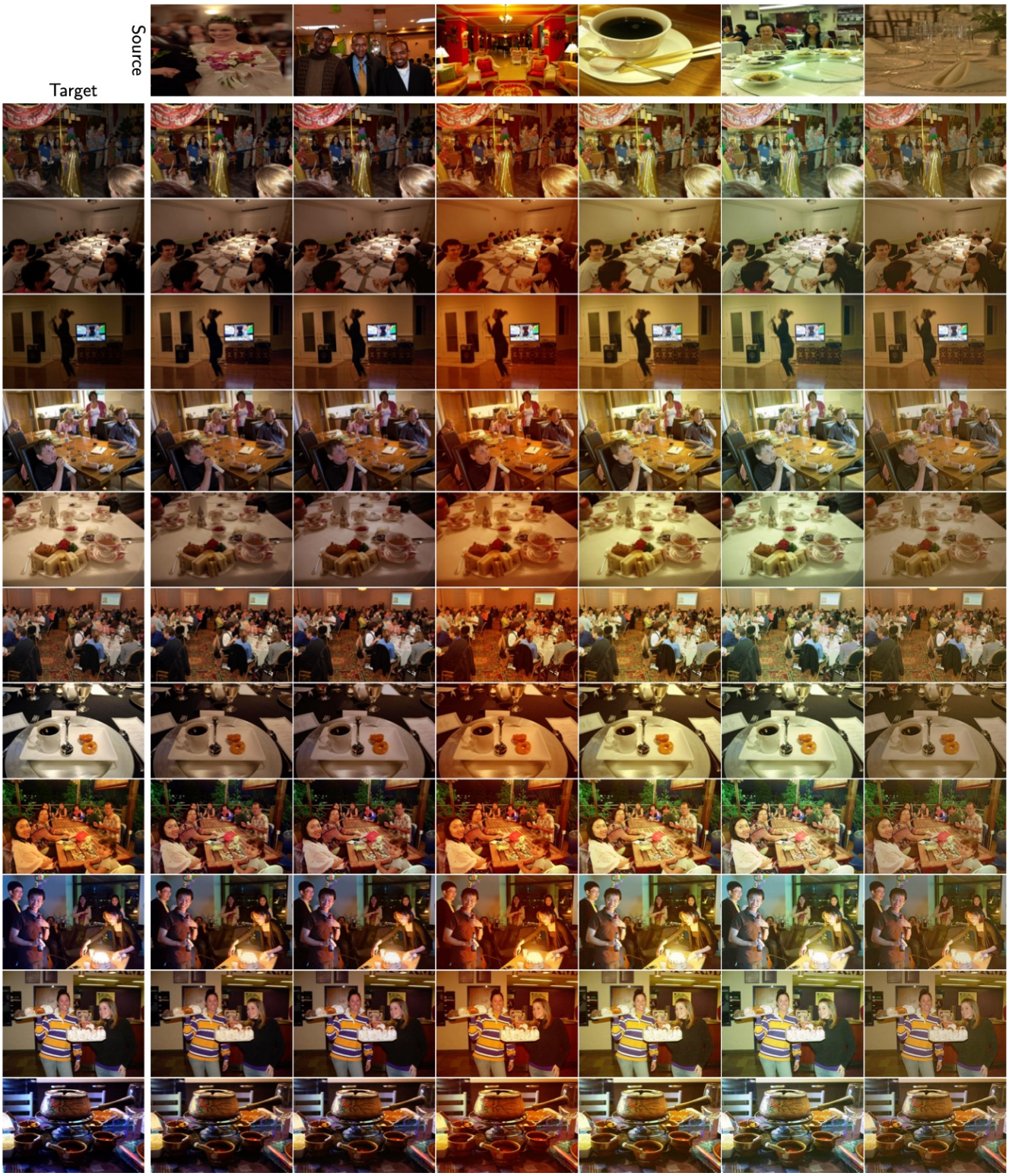


Figure F. Additional qualitative results transferring the light from one source image to another target image.


Figure G. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

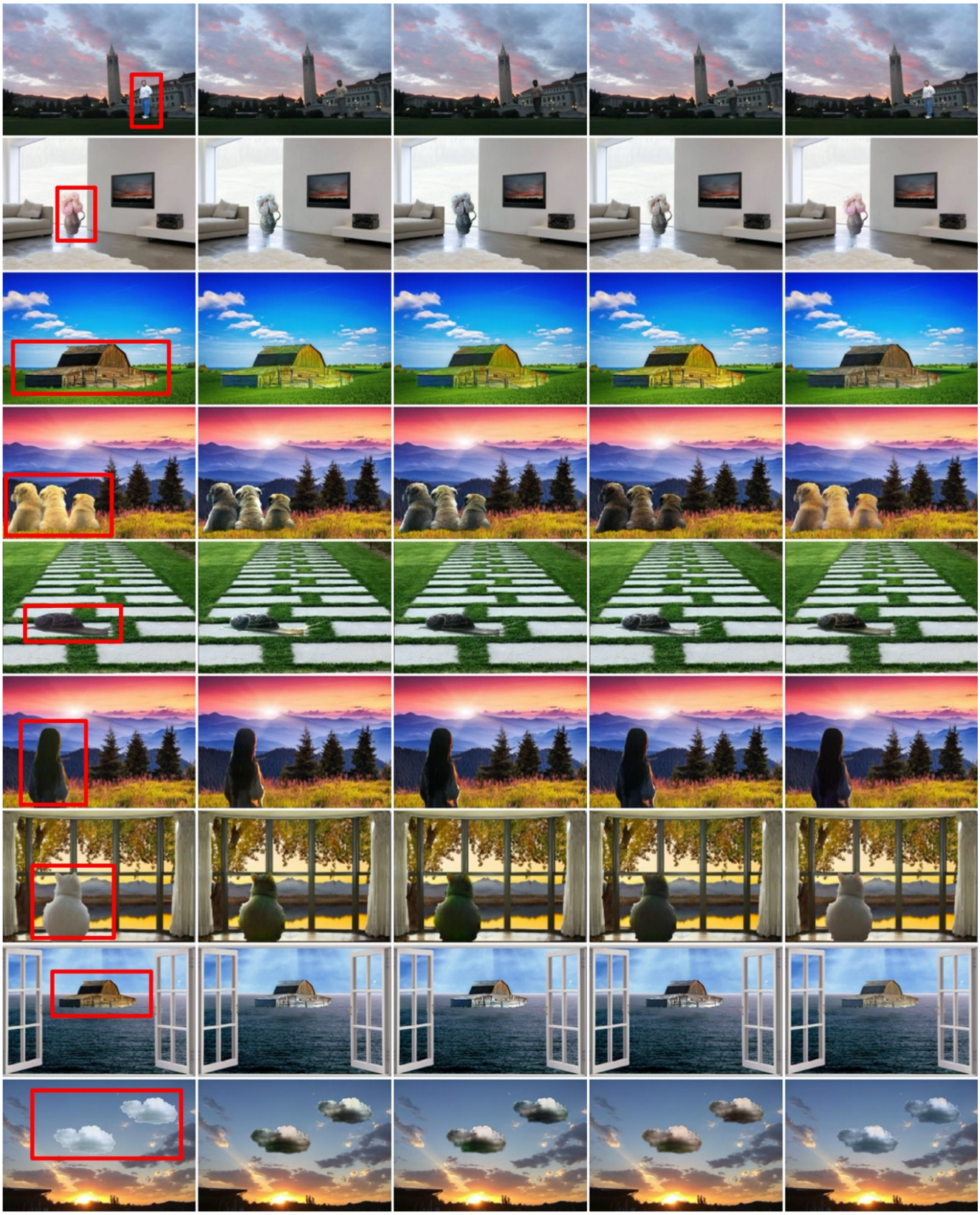


Figure H. Visual comparison results on real composite images. Red boxes in composite images mark foreground.


Figure I. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

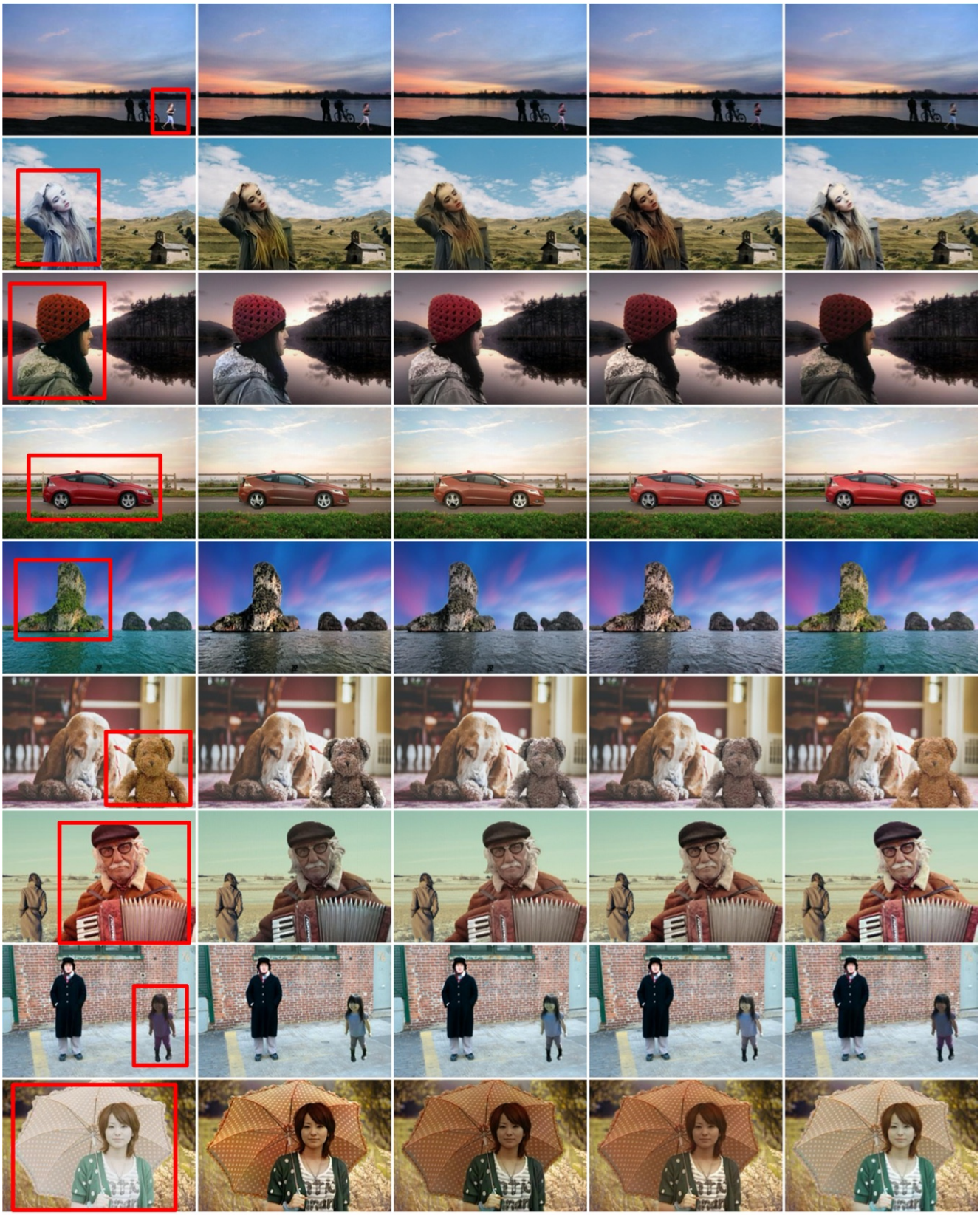


Figure J. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

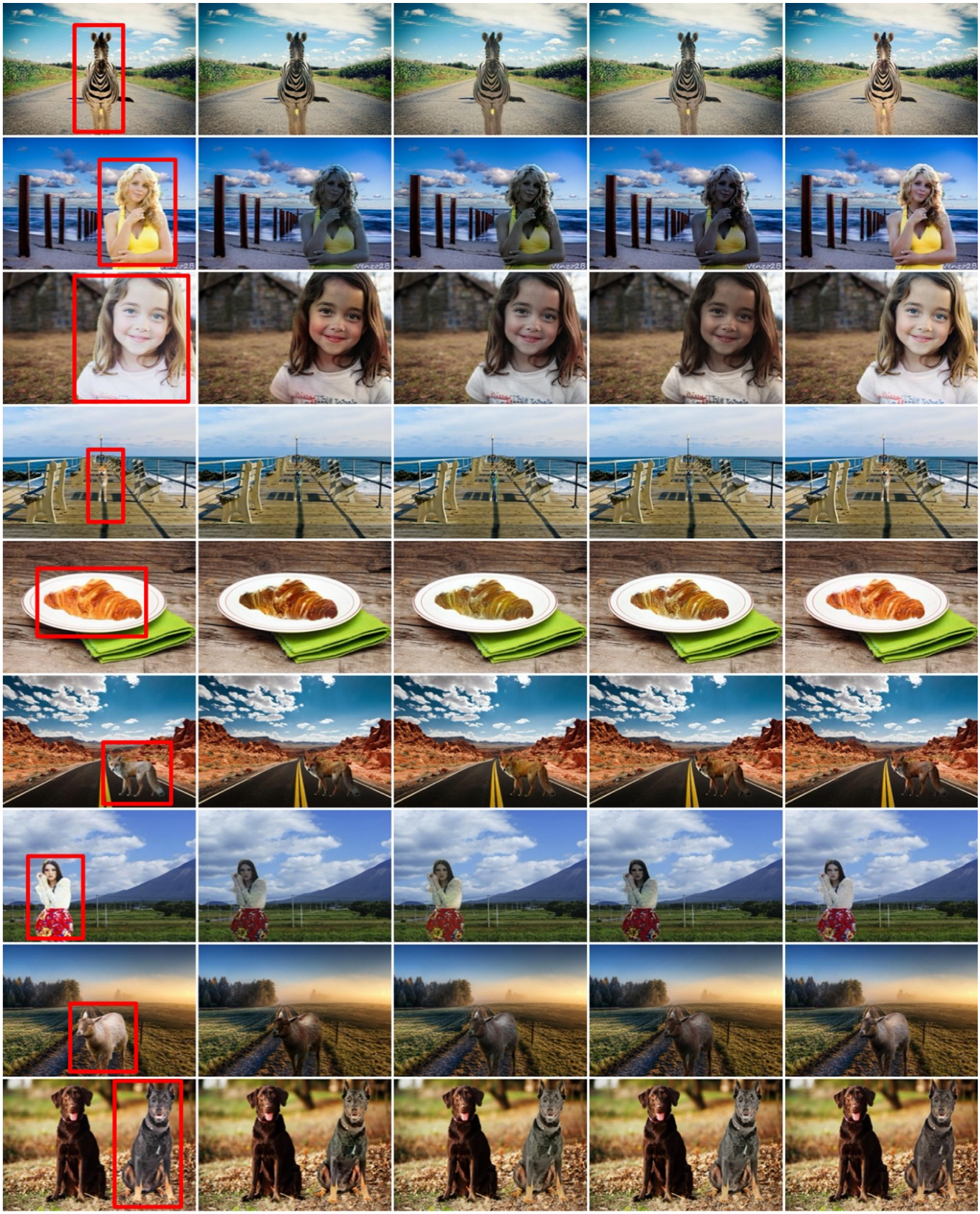


Figure K. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

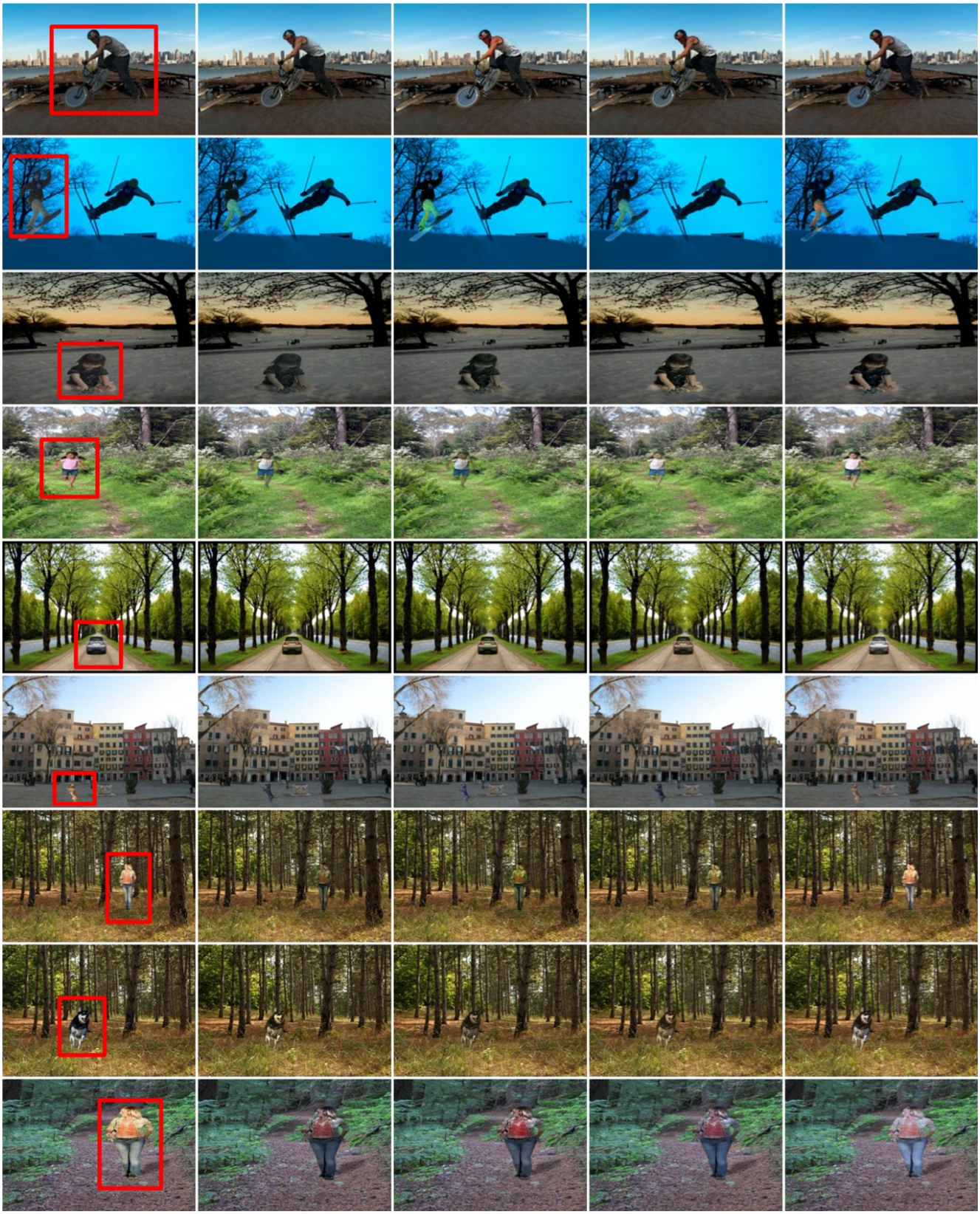


Figure L. Visual comparison results on real composite images. Red boxes in composite images mark foreground.


Figure M. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

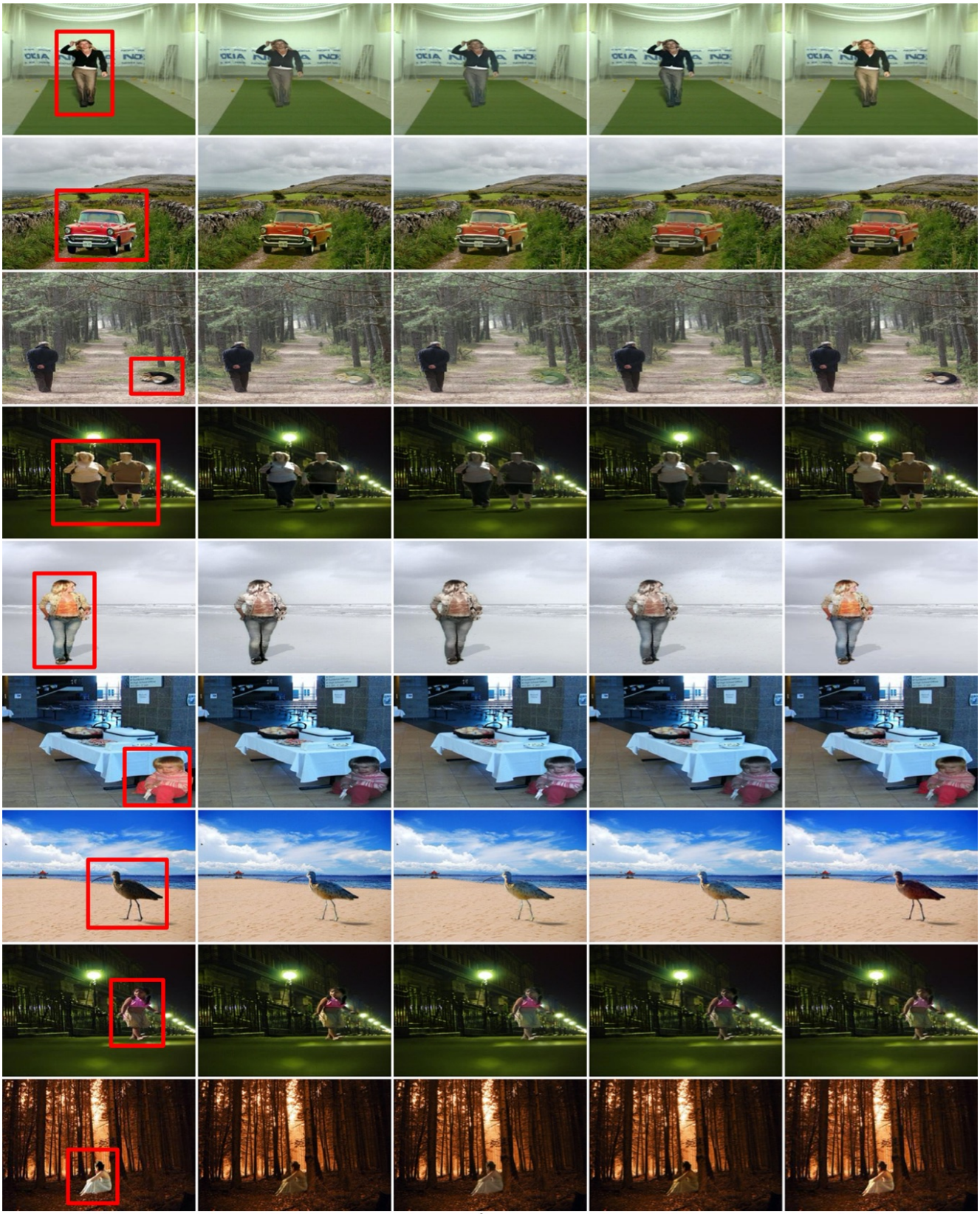


Figure N. Visual comparison results on real composite images. Red boxes in composite images mark foreground.

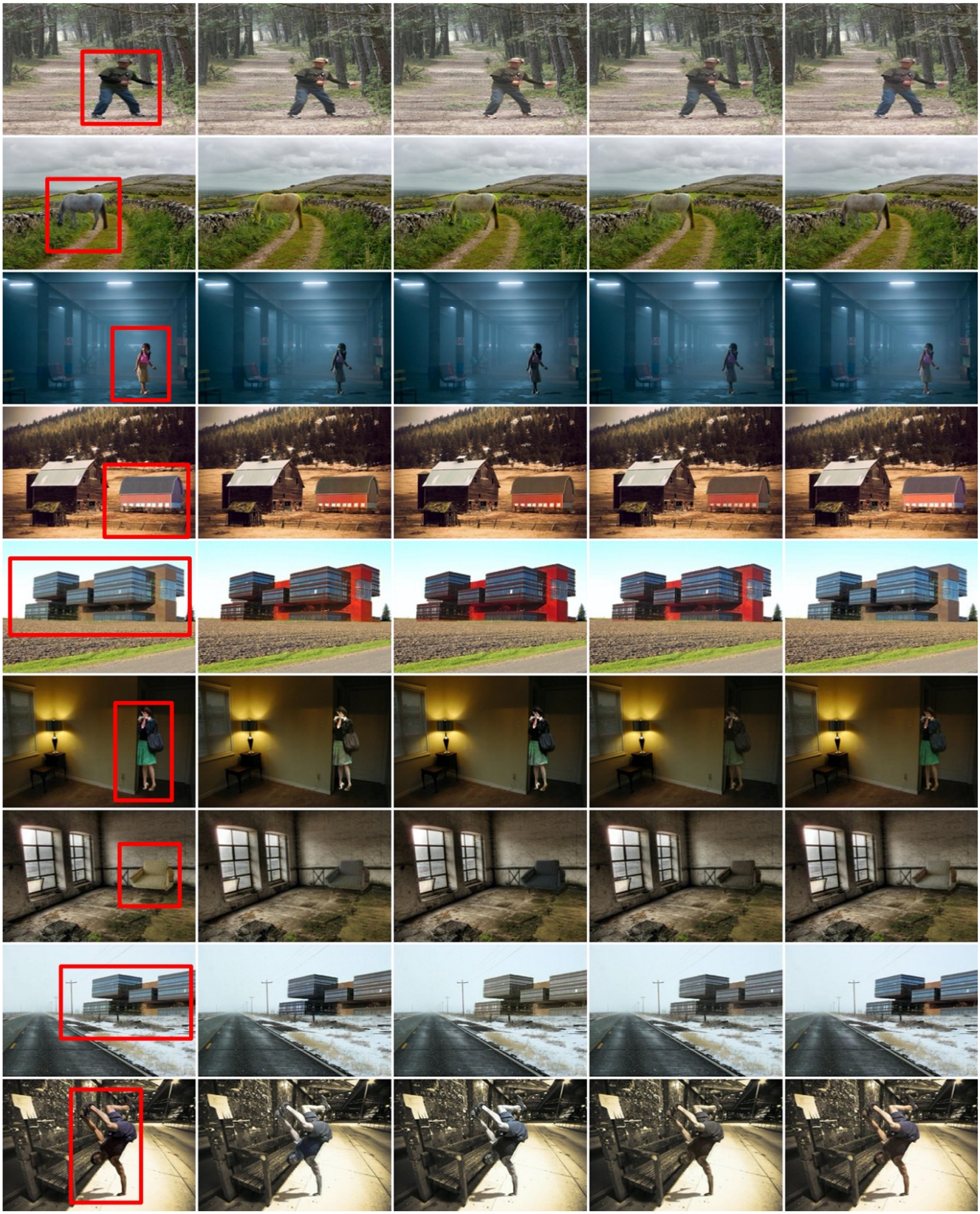


Figure O. Visual comparison results on real composite images. Red boxes in composite images mark foreground.


Figure P. Visual comparison results on real composite images. Red boxes in composite images mark foreground.


Figure Q. Visual comparison results on real composite images. Red boxes in composite images mark foreground.


Figure R. Visual comparison of image inpainting on CelebA [6] (left) and Paris StreetView [3] (right).

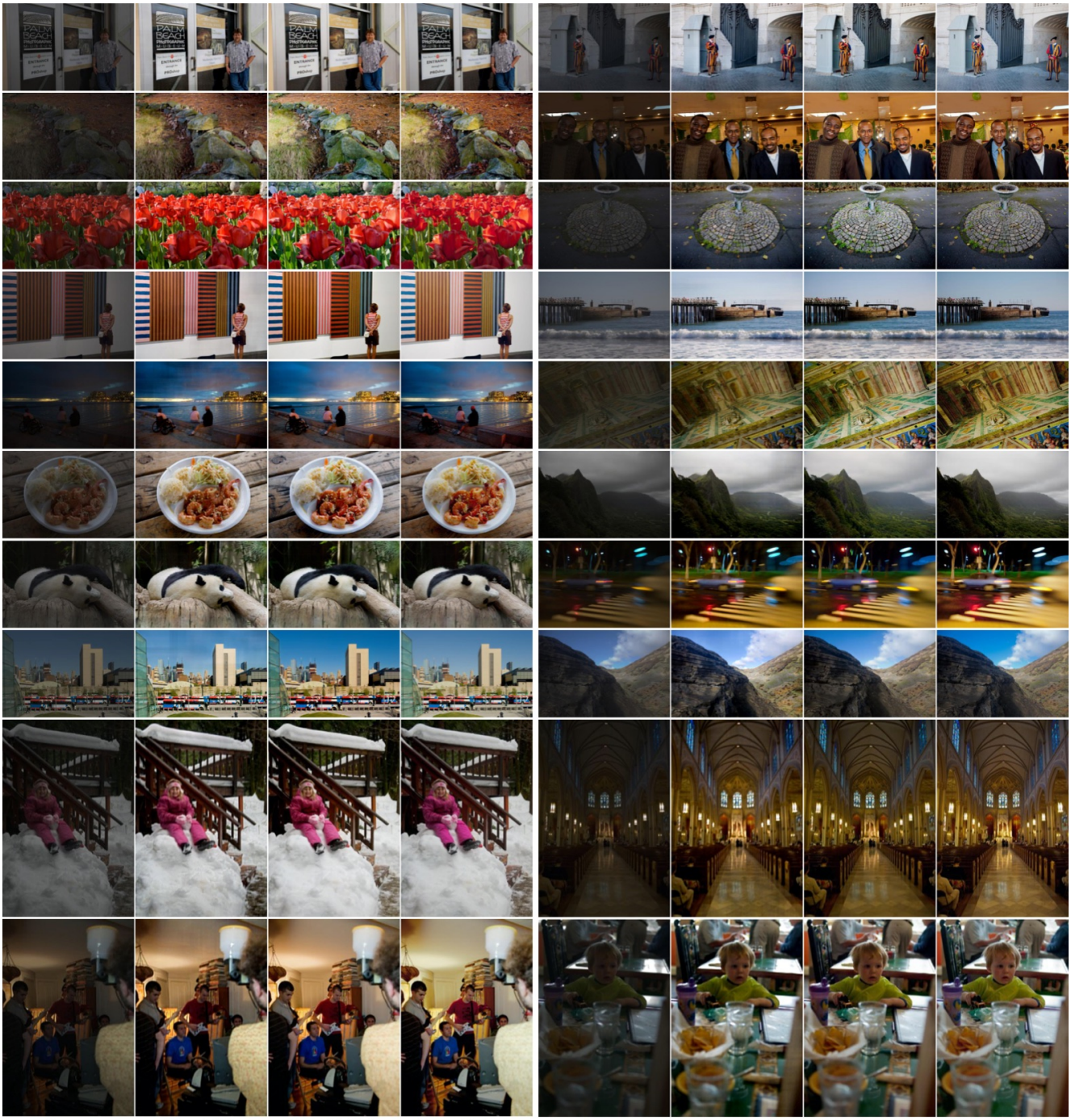


Figure S. Visual comparison of image enhancement on MIT-Adobe-5K-UPE [8].