Spatially-Adaptive Pixelwise Networks for Fast Image Translation Supplementary Materials

Tamar Rott Shaham[1] Michaël Gharbi[2] Richard Zhang[2] Eli Shechtman[2] Tomer Michaeli[1]

[1]Technion [2]Adobe Research

1. Implementation Details

1.1. Training Details

We use the Adam optimizer with $\beta_1 = 0$, $\beta_1 = 0.999$, and learning rates of 0.0001 and 0.0004 for the generator and discriminator, respectively. We initialize our model using the Xavier method, and train it for 300 epochs, with linear learning rate decay over the last 100 epochs. The weights of the feature matching loss and the VGG loss are $\lambda_{\text{feat}} = 10$ and $\lambda_{\text{vgg}} = 10$, respectively.

1.2. Low-resolution network architecture

Our low resolution network gets a bilinearly downsampled version of the input image, and further reduces the spatial dimensions by a factor of 16, as follows. Let `CkSl` denote a 3×3 Convolution-InstanceNorm-ReLU layer with `k` filters, and stride of `l`. Then, the architecture of our low resolution network is: `C64S1-C64S2-C64S2-C64S2-C64S2-C1024S2-C1024S1-C1024S1-C1024S1-C1024S1-C1024S1-C1024S1`. We use one additional 1×1 convolutional layer with C_{total} channels, where C_{total} is the total number of parameters per spatial location of each pixelwise MLP network (see 1.3 for additional details). We use reflection padding in all layers to reduce boundary artifacts.

1.3. Pixelwise networks architecture

Our network processes each of the input pixels by an MLP. Let `fck` denote a fully-connected-LeakyReLU layer with a `k`-dimensional output, where the leaky ReLU has a slope of 0.01. Then Each of the pixelwise networks has the following architecture: `fc64-fc64-fc64-fc64-fc3`. In the last layer we use Tanh instead of leaky ReLU. That is, $C_{\text{total}} = \sum_n [C_n C_{n-1} + C_n]$, including biases (*e.g.* for input with 28 channels, $C_{total} = 28 \times 65 + 3 \times 64 \times 65 + 64 \times 4$).

1.4. Discriminator architecture

We use a multi-scale discriminator [6], with a 70×70 Patch-GAN [2] at each scale. For images of size 512×512 we use 2 scales, and for size 1024×1024 we use 3 scales. Each of the discriminators has an identical architecture. Specifically, let `Ck` denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with `k` filters and stride 2, where the leaky ReLU has a slope of 0.2. Then the architecture of each of the discriminators is: `C64-C128-C256-C512-C1`, where we do not use InstanceNorm after the first convolution layer.

1.5. Computational complexity.

The table below compares the number of multiply–accumulate (MAC) operations and the runtime memory consumed when processing 256×512 images. As can be seen, our method requires fewer MACs and less memory than SPADE, and it even requires slightly fewer MACs than pix2pix (despite achieving much better FID). Moreover, many of the MAC operations in our case are fully parallelizable (see next paragraph). Please note that this efficiency doesn't come on the expense of expressiveness, as the number of trainable parameters in our model is larger than in pix2pix and almost the same as in SPADE (54.4M for pix2pix, 83.2M for ASAP, 93M for SPADE).

method	MACs (G)	runtime memory (GB)
SPADE [4]	281.0	1.19
pix2pix [2]	56.8	0.28
ASAP (ours)	55.1	0.55

1.6. Achieving high throughput with pixel-wise MLPs.

The implementation of the pixel-wise MLPS uses parallel matrix–matrix multiplies on the GPU, between the flattened versions of the input tensor and the 1×1 MLPs kernels. Therefore, all pixel locations are processed at once. That is, ϕ computes the parameters for all pixel locations, then all the functions f_p operate in parallel on their corresponding pixel locations p.

2. Additional ablations

2.1. ASAP-Net vs. slimmer baseline variants.

We gradually reduce the number of parameters in SPADE [4] and pix2pixHD [6] until reaching the same runtime as our ASAP-Net model. The resulting FID is shown in Fig. 1. As can be seen, for both Facades 512×512 and Facades 1024×1024, the slimmer SPADE and pix2pixHD models indeed have shorter runtimes, but also significantly worse FID scores comparing to our model.

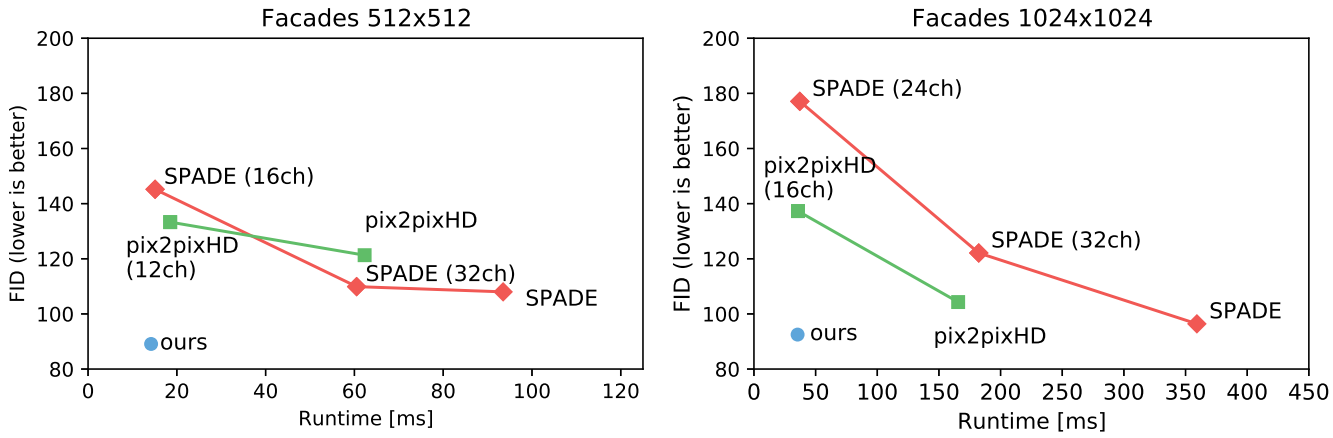


Figure 1: **ASAP vs. Baselines variations.** When gradually cutting down the number of parameters of SPADE [4] and pix2pixHD [6] to have the same runtime as our ASAP model, both incur degradation in FID scores for both Facades 512×512 and 1024×1024 images.

2.2. Network design

We quantify the effect of our network design on its performance, in terms of both runtime and visual quality (measured here by FID scores).

Low-resolution network ablation. We test additional two model designs; a low-resolution network with 256 channels and 15 layers, and one with 512 channels and 10 layers. As can be seen in Fig. 2, these two models achieve shorter runtime compared to our final model (which has 7 layers with 1024 channels), but this comes at the cost of degradation in visual quality (quantified by FID scores).

Pixelwise networks ablation. Here we test the the effect of the number of channels in the pixelwise MLP networks on the model's performance. As can be seen from Fig. 2, reducing the number of channels from 64 (as in our final design) to 32 drastically affects the FID score. This trend continues when we further reduce the channels to only 16.

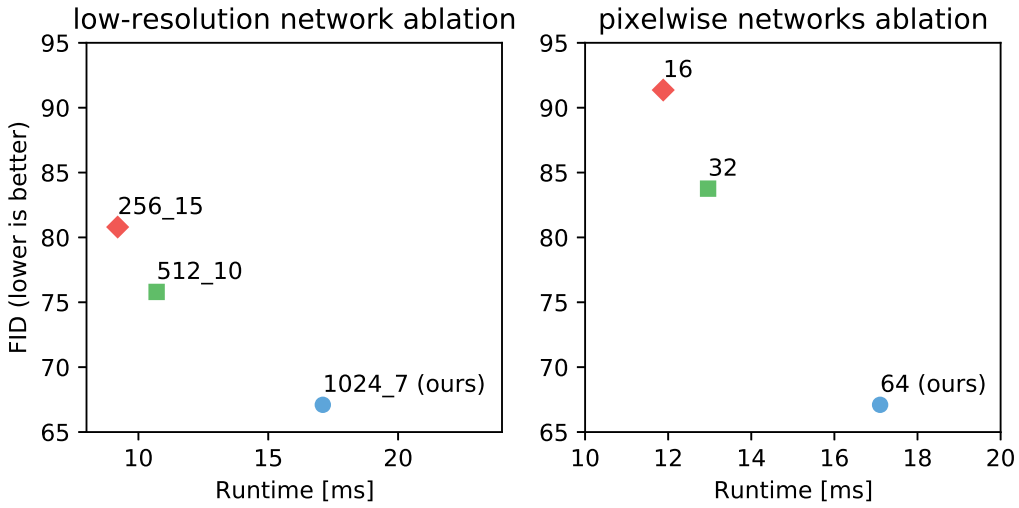


Figure 2: **Network architecture ablation.** We test how the low-resolution network design and the pixelwise MLP architecture affect the performance of our ASAP model (on Cityscapes 256×512.).

3. Additional results

3.1. Visual comparisons

Figures 3,4,5,6 present additional visual comparisons with CC-FPSE [3], SPADE [4] and pix2pixHD [6] for the Cityscapes dataset [1] (for images of size 256×512 and 512×1024) and with SPADE [4] and pix2pixHD [6] for the Facades datates [5] (for images of size 512×512 and 1024×1024).

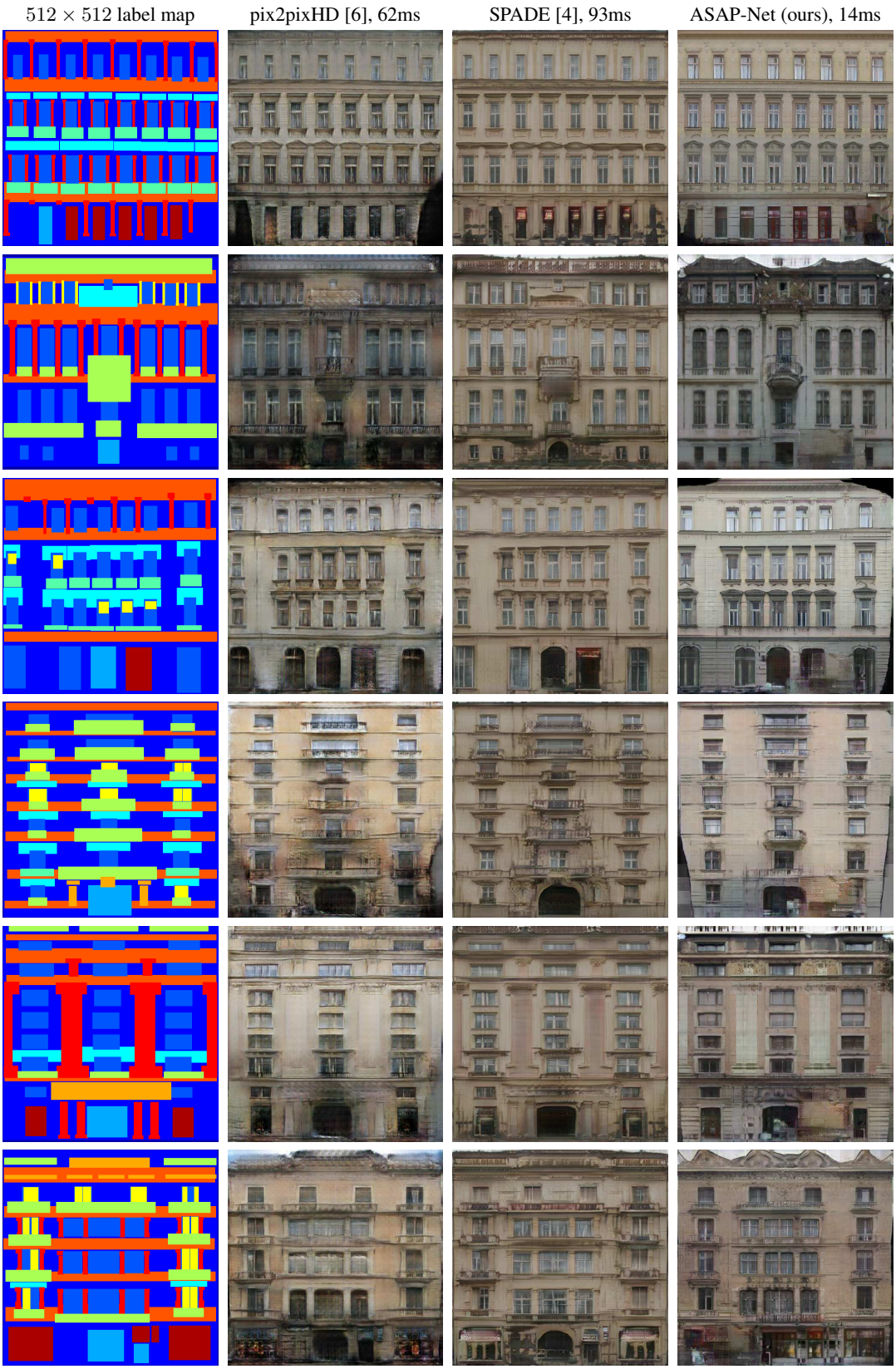


Figure 3: **Facades 512 × 512.**

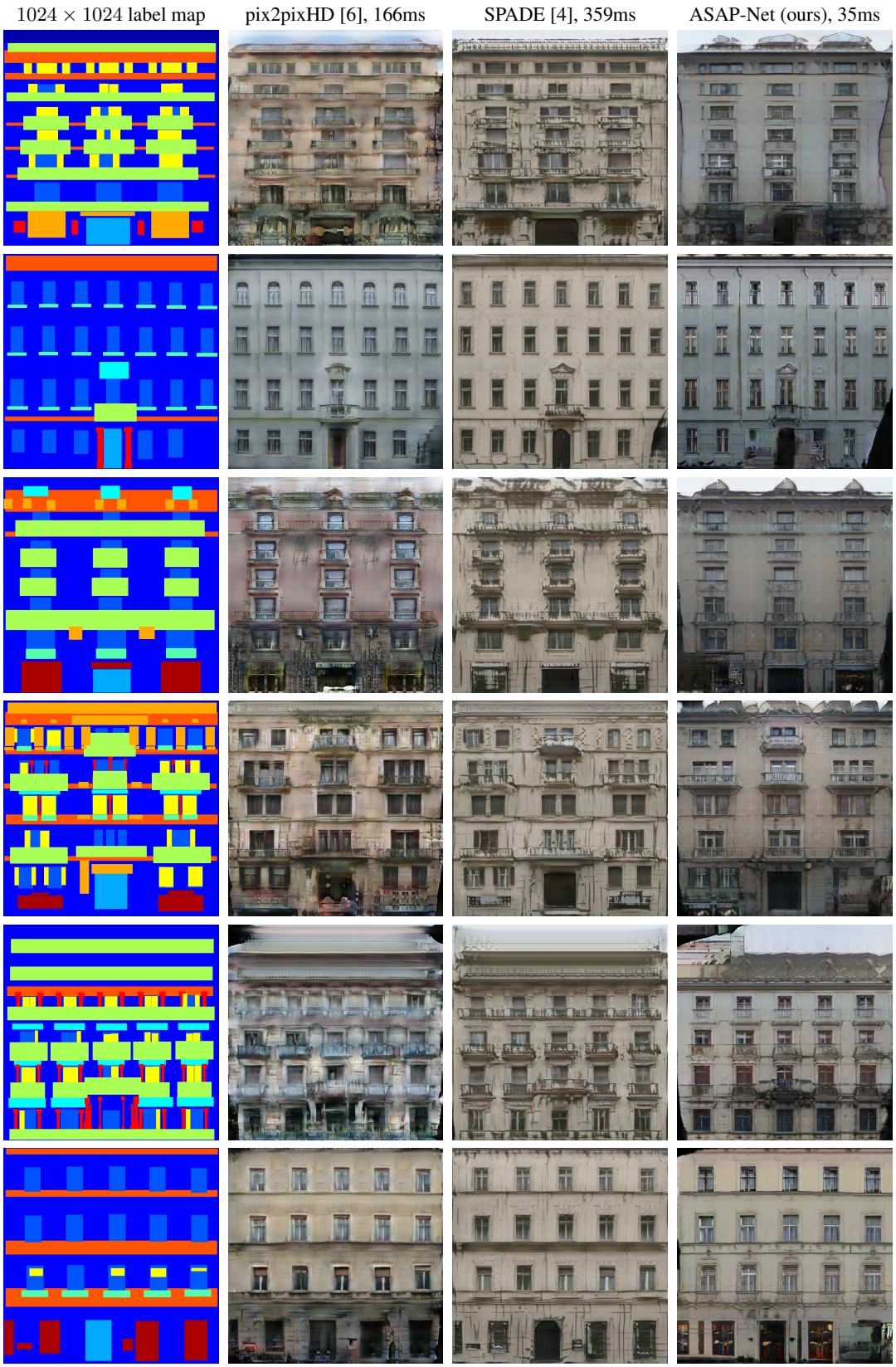


Figure 4: **Facades 1024 × 1024**.

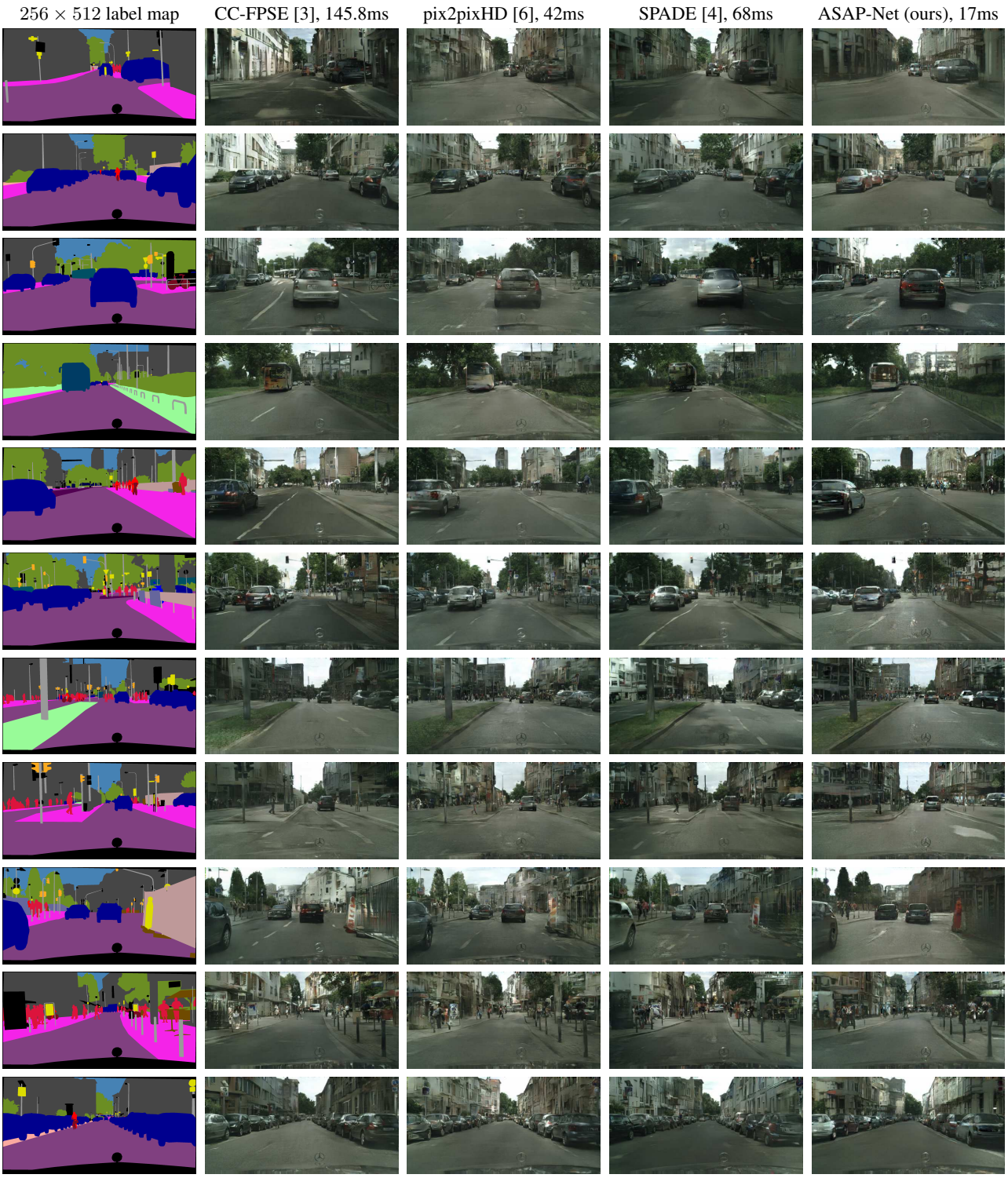


Figure 5: **Cityscapes 256 × 512.**

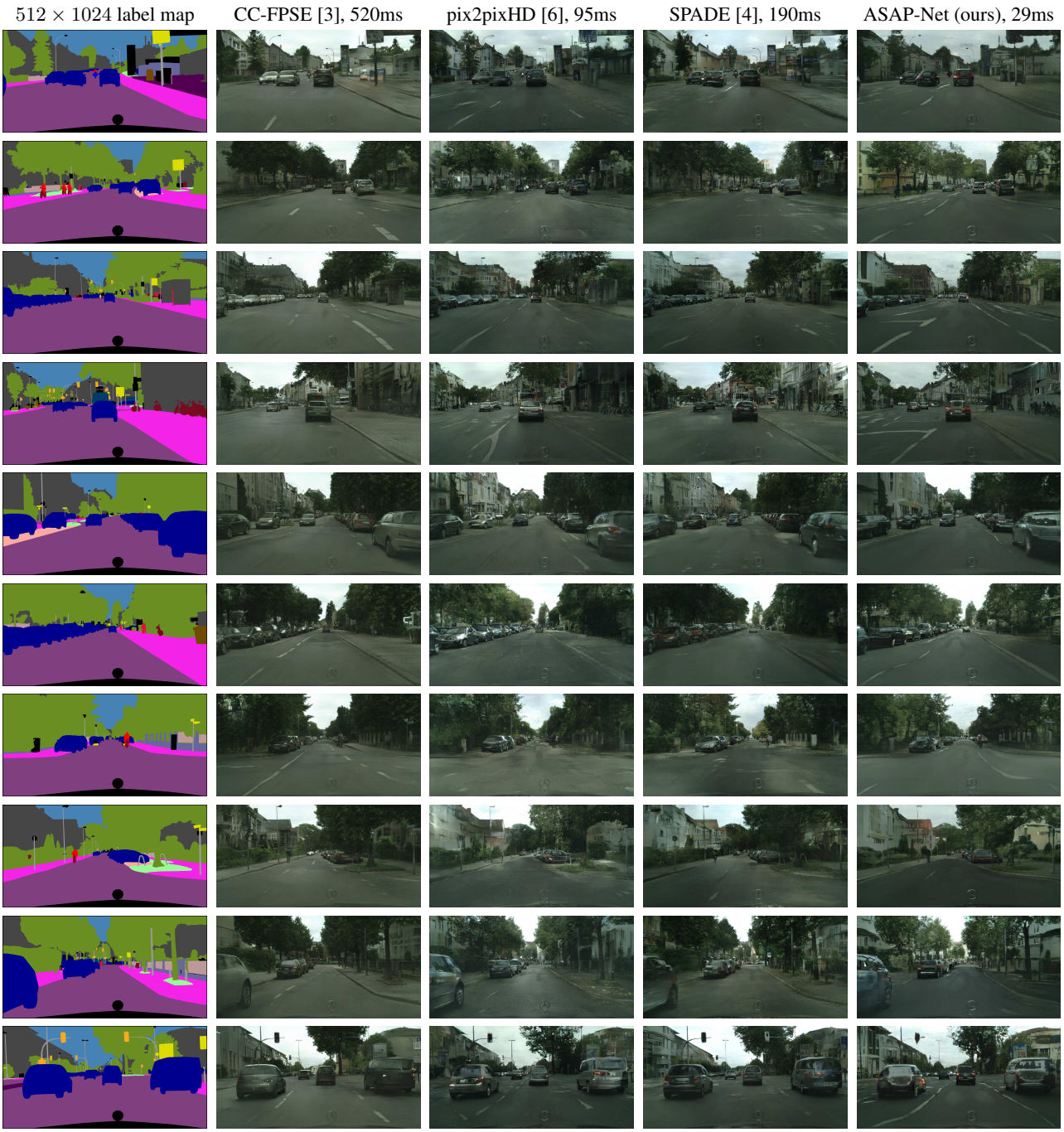


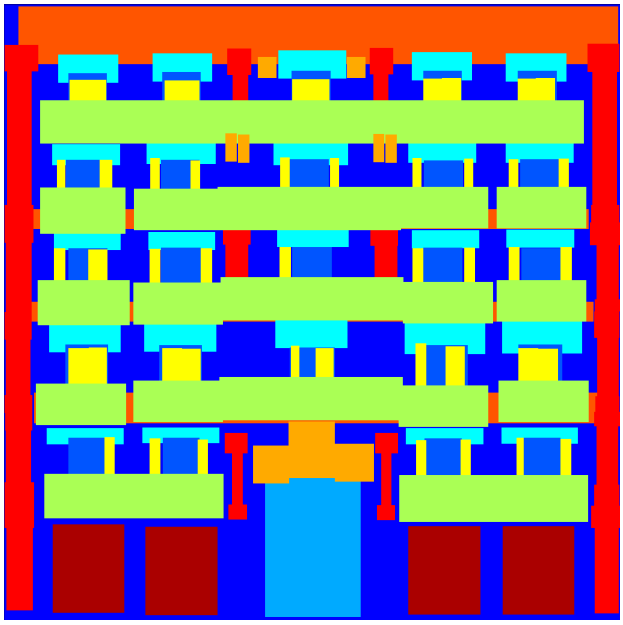
Figure 6: **Cityscapes 512 × 1024.**

3.2. 4Mpix results

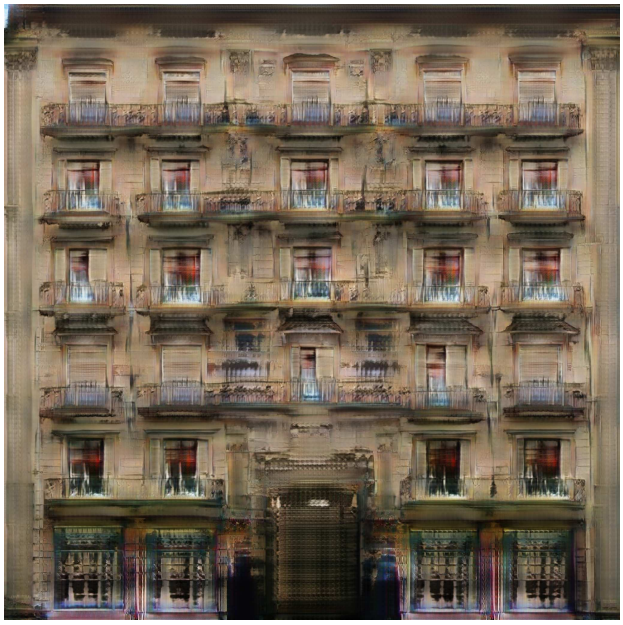
To showcase the effectiveness of our method for high-resolution images, we generated 2048×2048 label maps of Facades by upsampling the original maps [5]. We fed those into our model, which was trained on 1024×1024 images, as well as to SPADE [4] and pix2pixHD [6] trained for the same image size. As can be seen in Figs. 7,8, our model achieves comparable visual quality to the two baselines but has a significantly shorter runtime[1]. Please note that since the models were trained on 1024×1024 images, objects in the 2048×2048 maps are larger than those the network trained to produce.

[1]For SPADE, the model cannot process such large images on the Nvidia 2080ti GPU (which we used for all our other experiments). We therefore run it on an RTX8000 GPU with 48Gb memory. Although such a GPU should be able to process the image faster, our ASAP-Net still achieves a significant speed up.

2048 × 2048 label map



pix2pixHD [6], 711ms

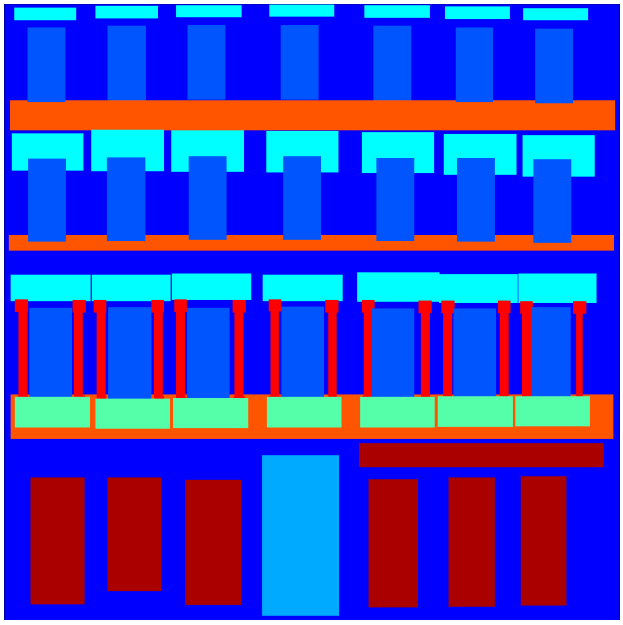


SPADE [4], 1326ms

ASAP-Net (ours), 130ms

Figure 7: **4Mpix facades.**

2048 × 2048 label map



pix2pixHD [6], 711ms

SPADE [4], 1326ms

ASAP-Net (ours), 130ms

Figure 8: **4Mpix facades.**

3.3. Beyond labels.

In Fig. 9 we show additional results for the task of depth estimation, comparing to SPADE [4]. The architecture of SPADE is designed specifically for the task of translating labels into images, and is therefore less successful in task like depth estimations. Our approach, on the other hand, manages to preform well on this task.

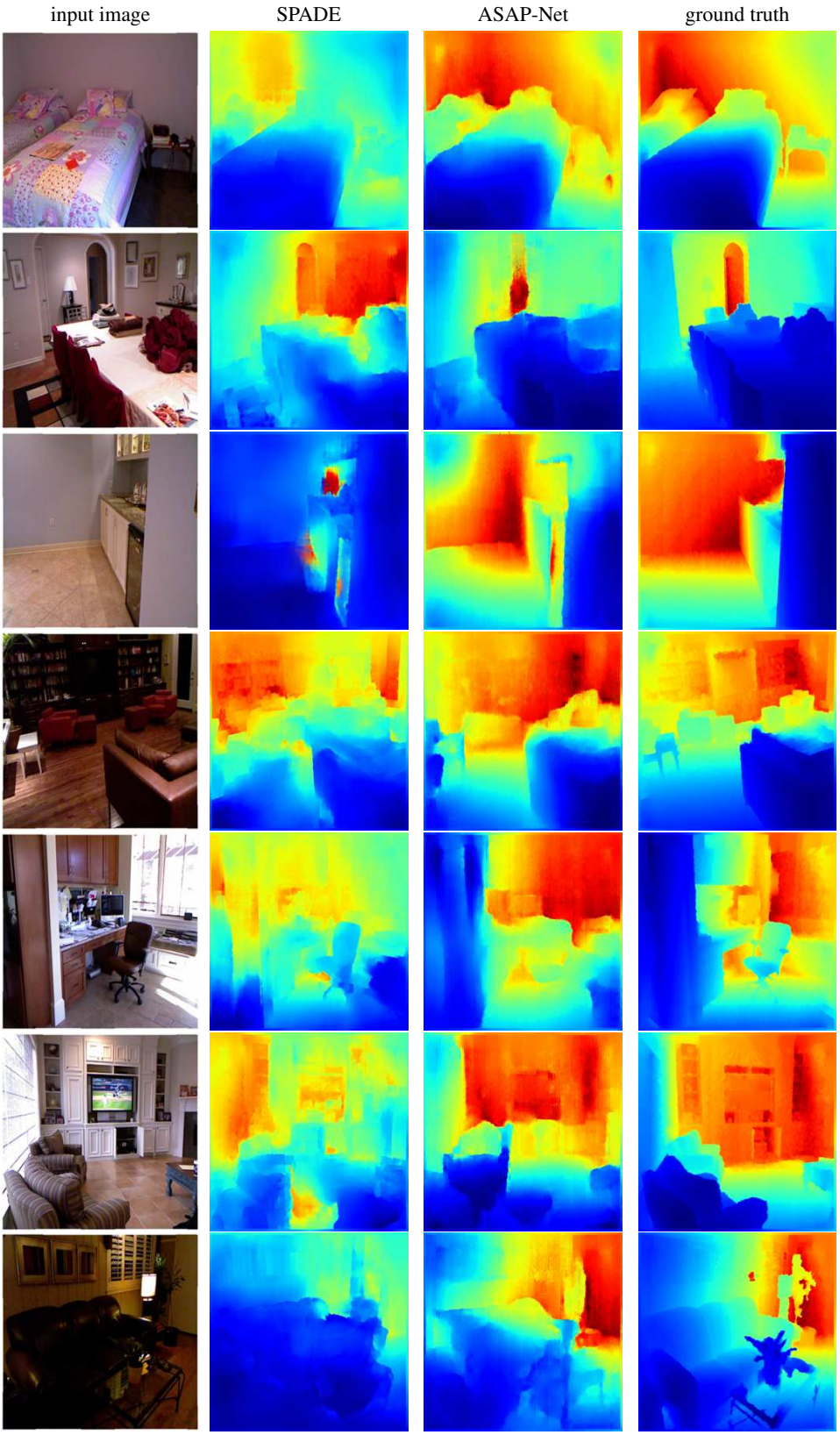


Figure 9: **Beyond labels.**

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

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

[3] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, and Hongsheng Li. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *Advances in Neural Information Processing Systems*, 2019.

[4] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[5] Radim Tyleček and Radim Šára. Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*. Springer, 2013.

[6] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, 2018.