A. Experimental Details

A.1. Synthetic Dataset

To sample a random image, we started with a 9×9 grid, where each grid cell is 16×16 pixels. We randomly sample a program $P = ((s_1, c_1), ..., (s_k, c_k))$ (for $k = 12$), where each perceptual component c is a randomly selected MNIST image (downscaled to our grid cell size and colorized). To create correlations between different parts of P, we sample (s_h, c_h) depending on $(s_1, c_1), ..., (s_{h-1}, c_{h-1})$. First, to sample each component c_h, we first sample latent properties of c_h (i.e., its MNIST label $\{0, 1, ..., 4\}$ and its color $\{\text{red}, \text{blue}, \text{orange}, \text{green}, \text{yellow}\}$). Second, we sample the parameters of s_h conditional on these properties. To each of the 25 possible latent properties of c_h, we associate a discrete distribution over latent properties for later elements in the sequence, as well as a mean and standard deviation for each of the parameters of the corresponding sketch s_h.

We then render P by executing each (s_h, c_h) in sequence. However, when executing (s_h, c_h), on each iteration (i, j) of the for-loop, instead of rendering the sub-image c_h at each position in the grid, we randomly sample another MNIST image $c_h^{(i,j)}$ with the same label as c_h, recolor $c_h^{(i,j)}$ to be the same color as c_h, and render $c_h^{(i,j)}$. By doing so, we introduce noise into the programmatic structure.

A.2. Generation from Scratch

SGM architecture. For the first stage of SGM (i.e., generating the program $P = (s, c)$), we use a 3-layer LSTM encoder $p_\phi(s, c \mid z)$ and a feedforward decoder $q_{\tilde{\phi}}(z \mid s, c)$. The LSTM includes sequences of 13-dimensional vectors, of which 6 dimensions represent the structure of the for-loop being generated, and 7 dimensions are an encoding of the image to be rendered. The image compression was performed via a convolutional architecture with 2 convolutional layers for encoding and 3 deconvolutional layers for decoding.

For the second stage of SGM (i.e., completing the structure rendering x_{struct} into an image x), we use a VED; the encoder $q_\theta(w \mid x_{\text{struct}})$ is a CNN with 4 layers, and the decoder $p_\theta(x \mid w)$ is a transpose CNN with 6 layers. The CycleGAN model has a discriminator with 3 convolutional layers and a generator which uses transfer learning by employing the pre-trained ResNet architecture.

Baseline architecture. The architecture of the baseline is a vanilla VAE with the same as the architecture as the VED we used for the second stage of SGM, except the input to the encoder is the original training image x instead of the structure rendering x_{struct}. The baselines with CycleGAN also use the same architecture as SGM with CycleGAN/GLCIC. The Spatial GAN was trained with 5 layers each in the generative/discriminative layer, and 60-dimensional global and 3-dimensional periodic latent vectors.

A.3. Image completion.

SGM architecture. For the first stage of SGM for completion (extrapolation of the program from a partial image to a full image), we use a feedforward network with three layers. For the second stage of completion via VAE, we use a convolutional/deconvolutional architecture. The encoder is a CNN with 4 layers, and the decoder is a transpose CNN with 6 layers. As was the case in generation, the CycleGAN model has a discriminator with 3 convolutional layers and a generator which uses transfer learning by employing the pre-trained ResNet architecture.

Baseline architecture. For the baseline VAE architecture, we used a similar architecture to the SGM completion step (4 convolutional and 6 deconvolutional layers). The only difference was the input, which was a partial image rather than an image rendered with structure. The CycleGAN architecture was similar to that used in SGM (although it mapped partial images to full images rather than partial images with structure to full images).

B. Additional Results

In Figure 6, we show examples of how our image completion pipeline is applied to the facades dataset, and in Figure 7, we show examples of how our image completion pipeline is applied to our synthetic dataset.

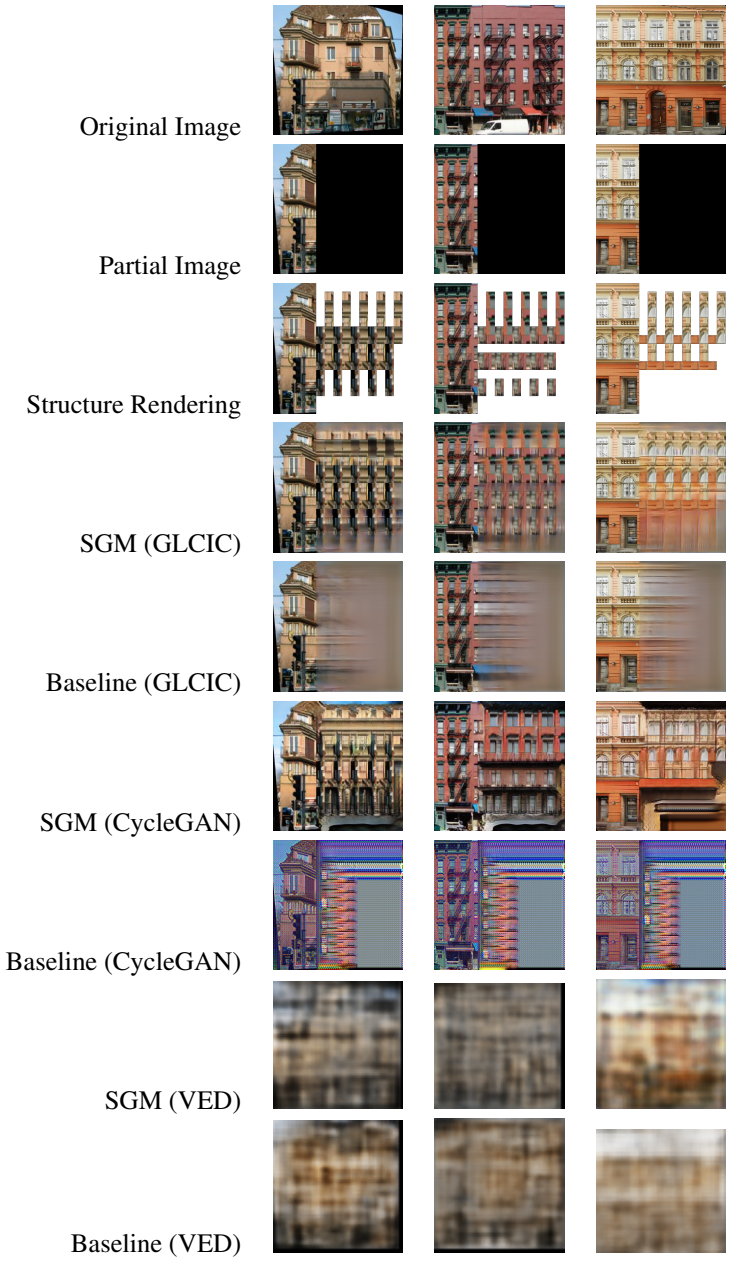

Figure 6. Examples of our image completion pipeline on the facades dataset.

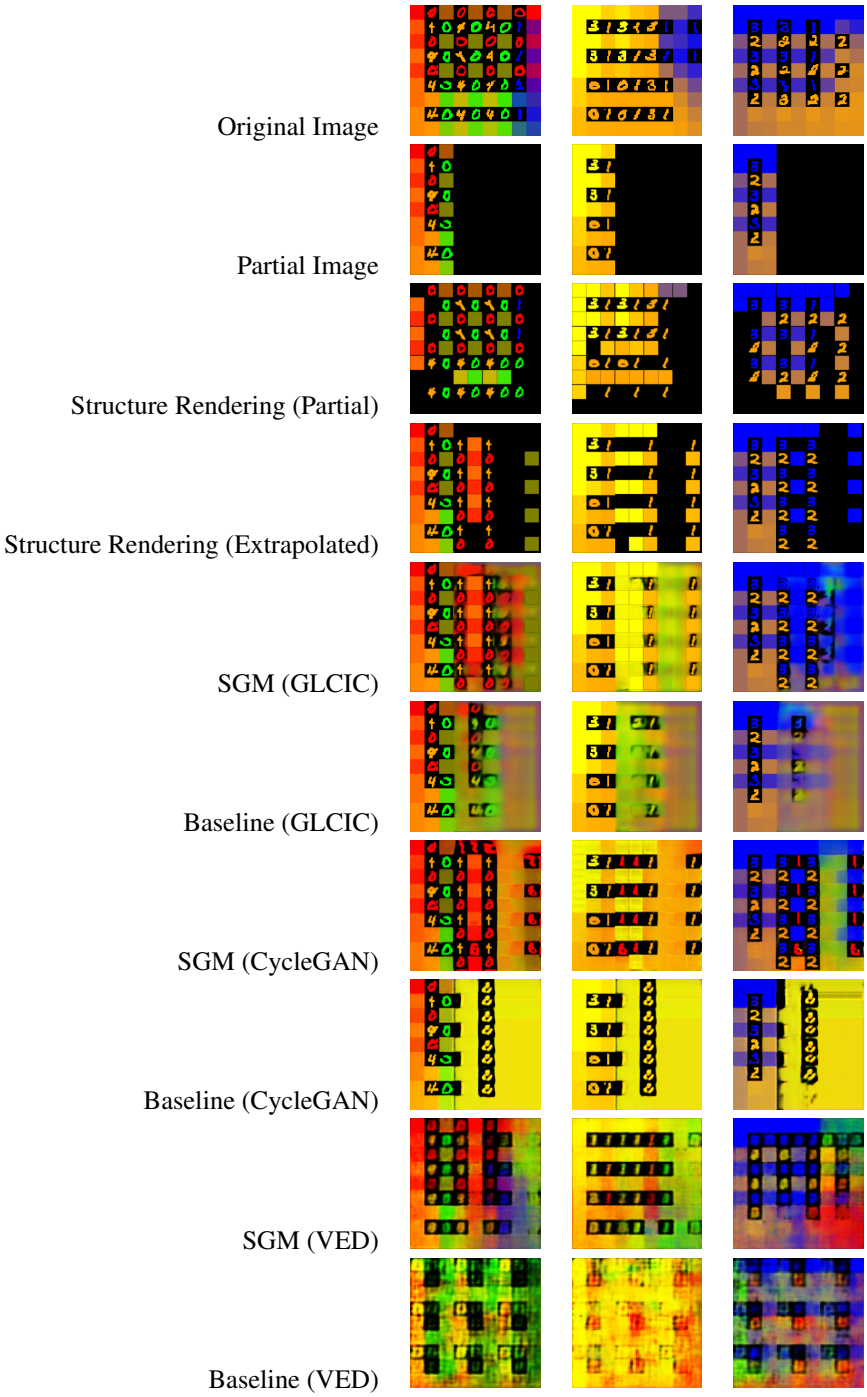

Figure 7. Examples of our image completion pipeline on our synthetic dataset.