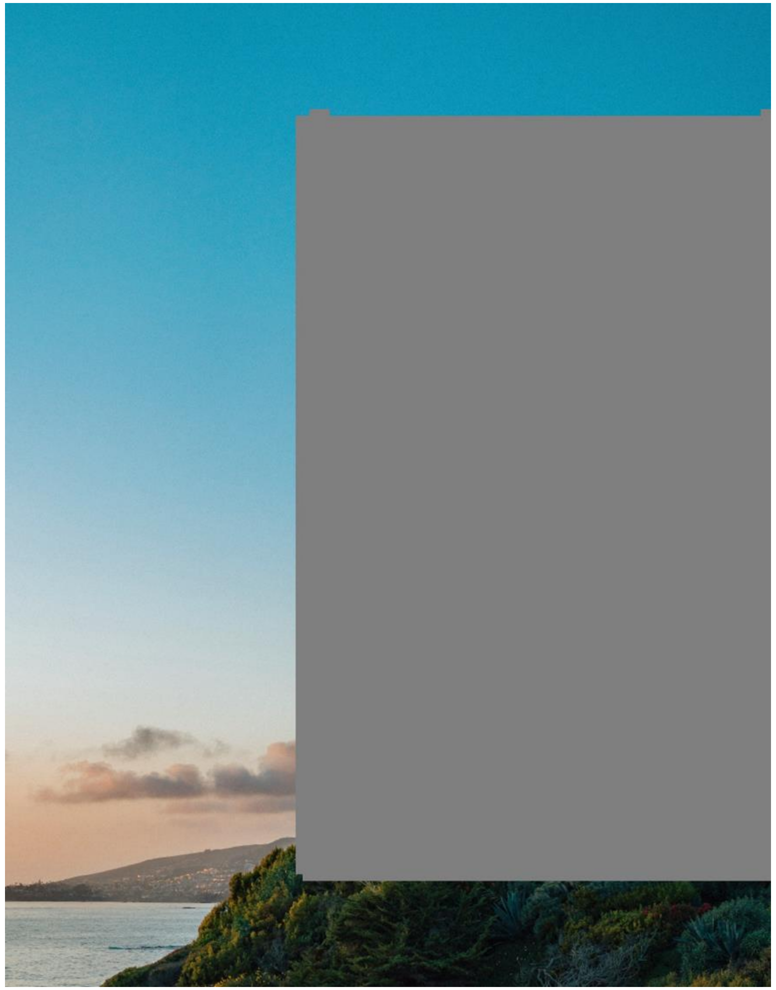
PixPerfect: Seamless Latent Diffusion Local Editing with Discriminative Pixel-Space Refinement

Haitian Zheng, Yuan Yao, Yongsheng Yu, Yuqian Zhou, Jiebo Luo, Zhe Lin

Adobe Research University of Rochester

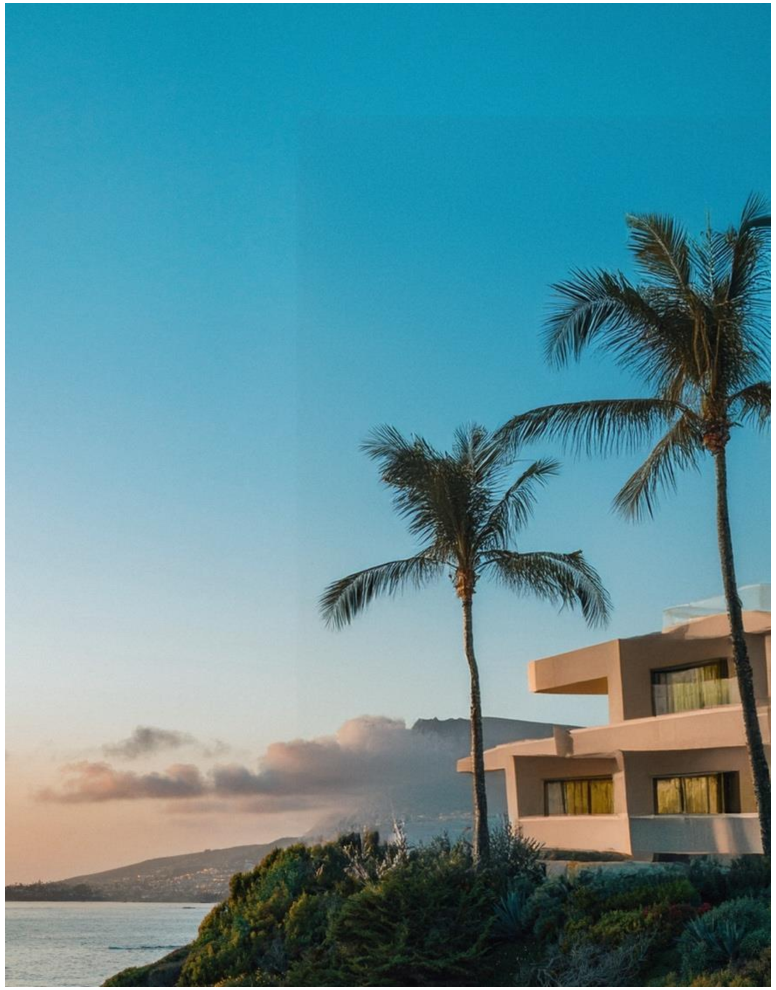
The Composition Seam Artifacts

- Discontinuity between the generated content and background contents
- Widely exists for inpainting and local editing Tasks



Input

“a resort with palm tree”

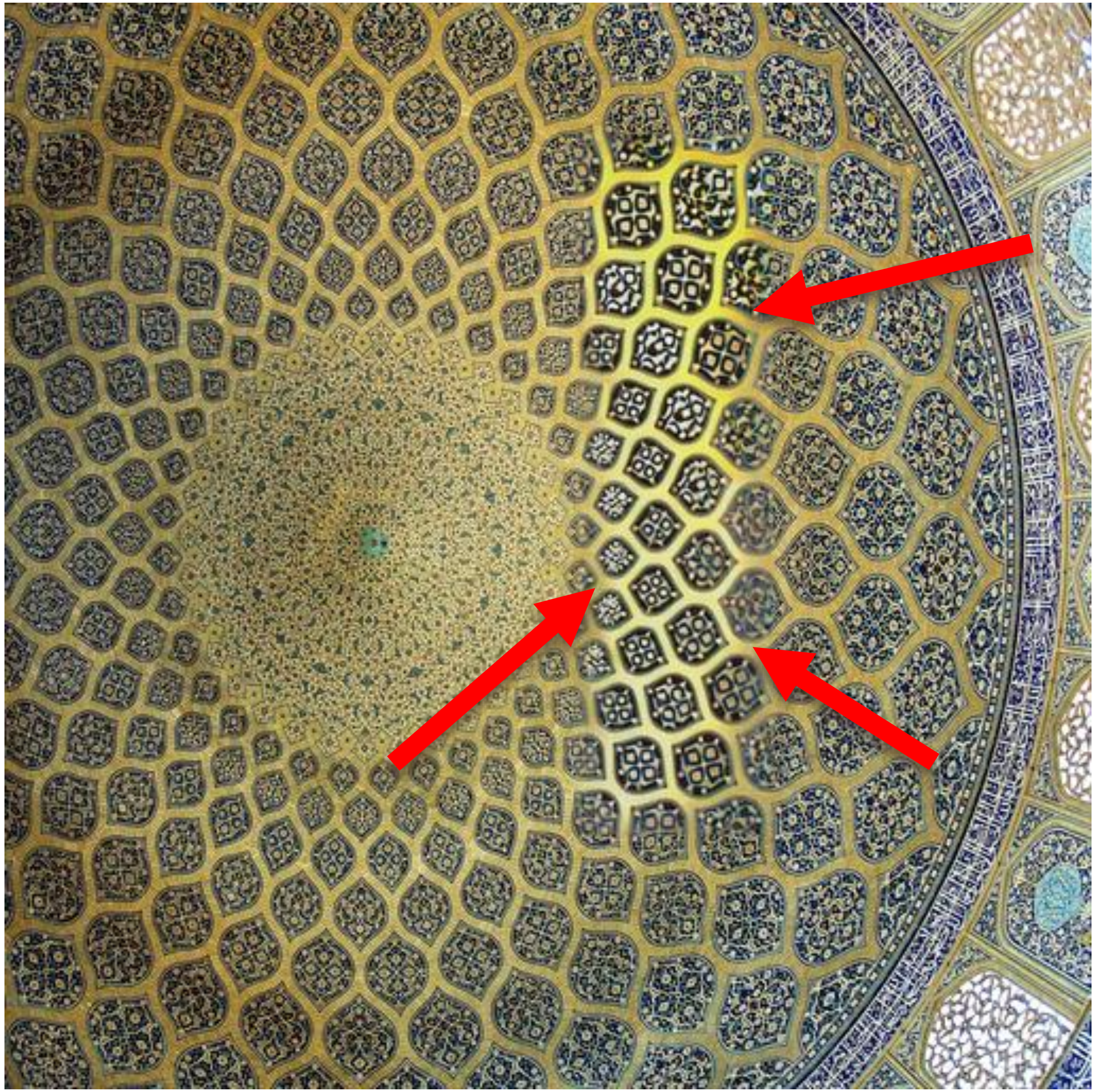


Inpainting Output

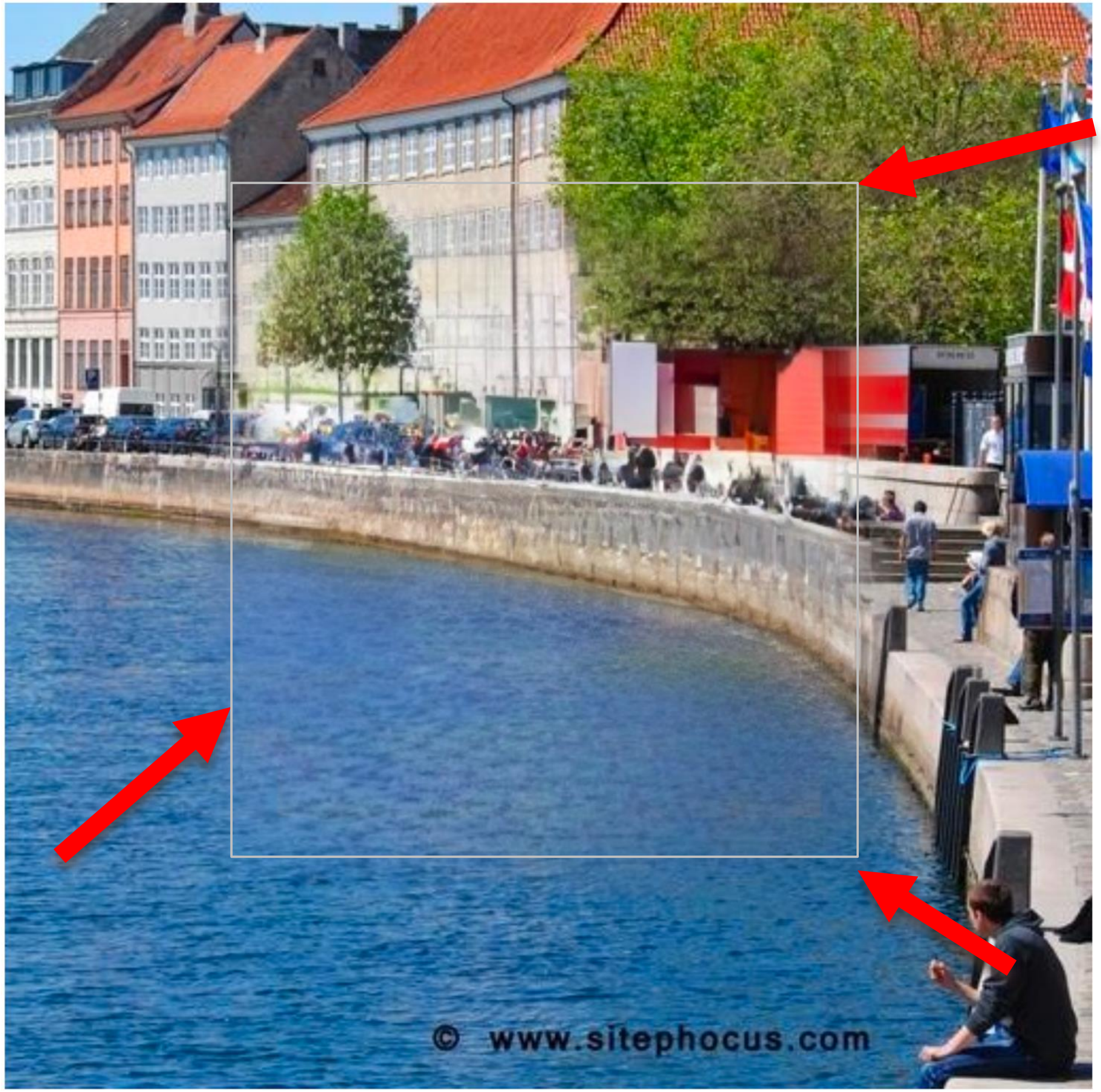
The Prevalence of Seam Artifacts

- Common in tasks that require composition, e.g. image inpainting, local editing
- A Longstanding Issue affects various models: GANs, diffusion models, few-step models, auto-regressive model
- Difficult to resolve and worsened by latent generative models.

Diffusion Inpainting Results



Few-step Inpainting results



Auto-regressive model (MaskGit)

Trade-offs in Text-to-Image Techniques

- **Alignment vs. Reconstruction**
- Technical Improvement focus more on text-to-image alignment, ignoring the harmonization issue
  - Classifier-free guidance
    - Color and saturation shifting
  - Latent-space generative models:
    - Texture distortion
    - Content Truncation at boundaries
  - Diffusion Inference
    - Smoothed textures

Content Truncation at Boundaries

GenAI inpainting results

What it should look like

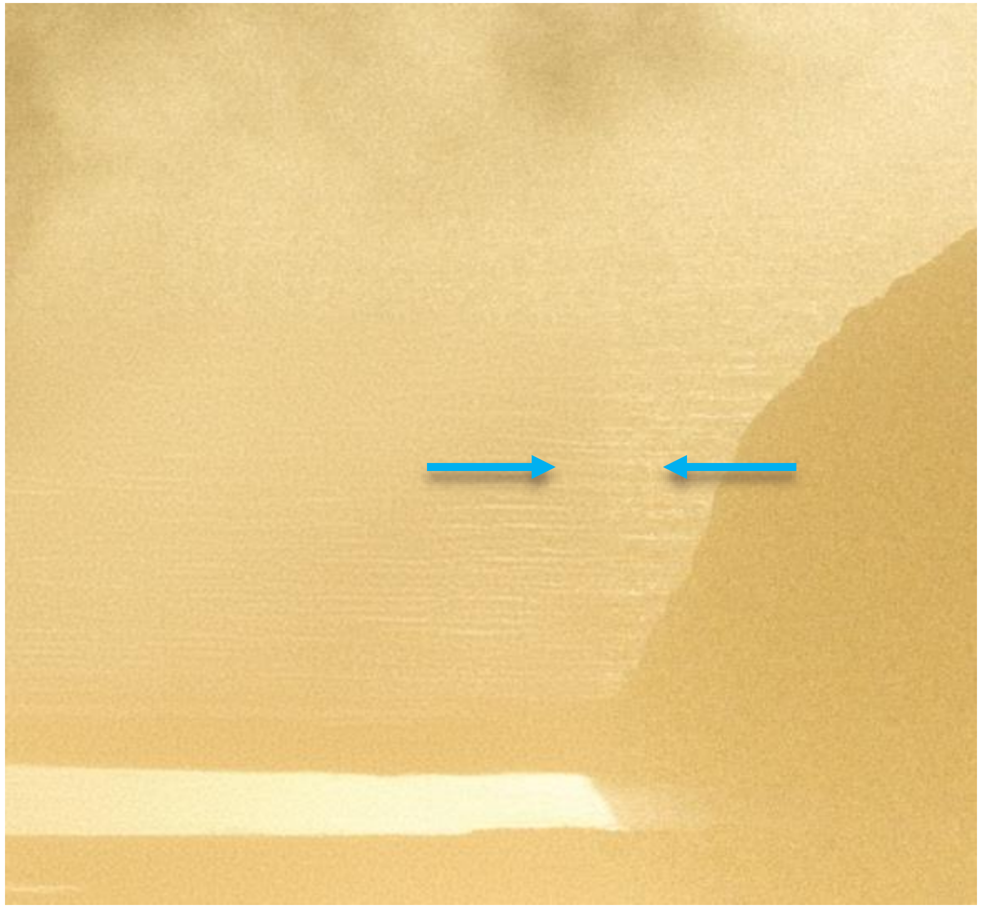
Adobe

Color Shifting

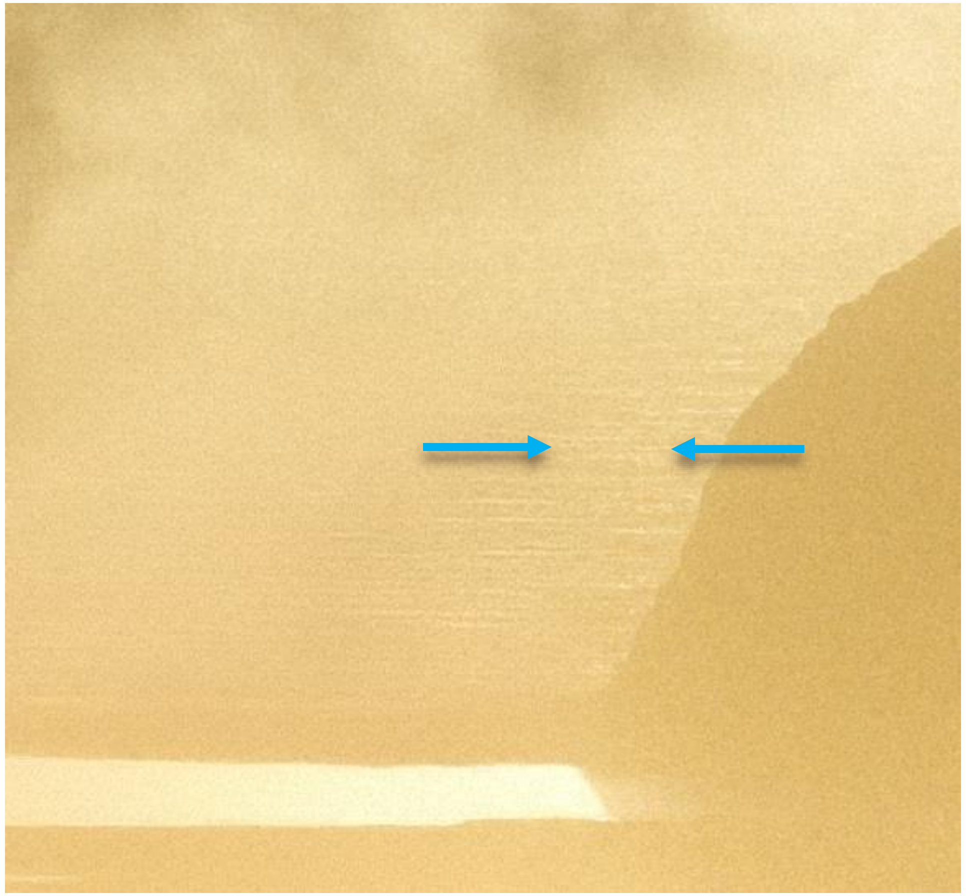
GenAI inpainting results

After applying our refiner

Noise Pattern Mismatch + Color Mismatch

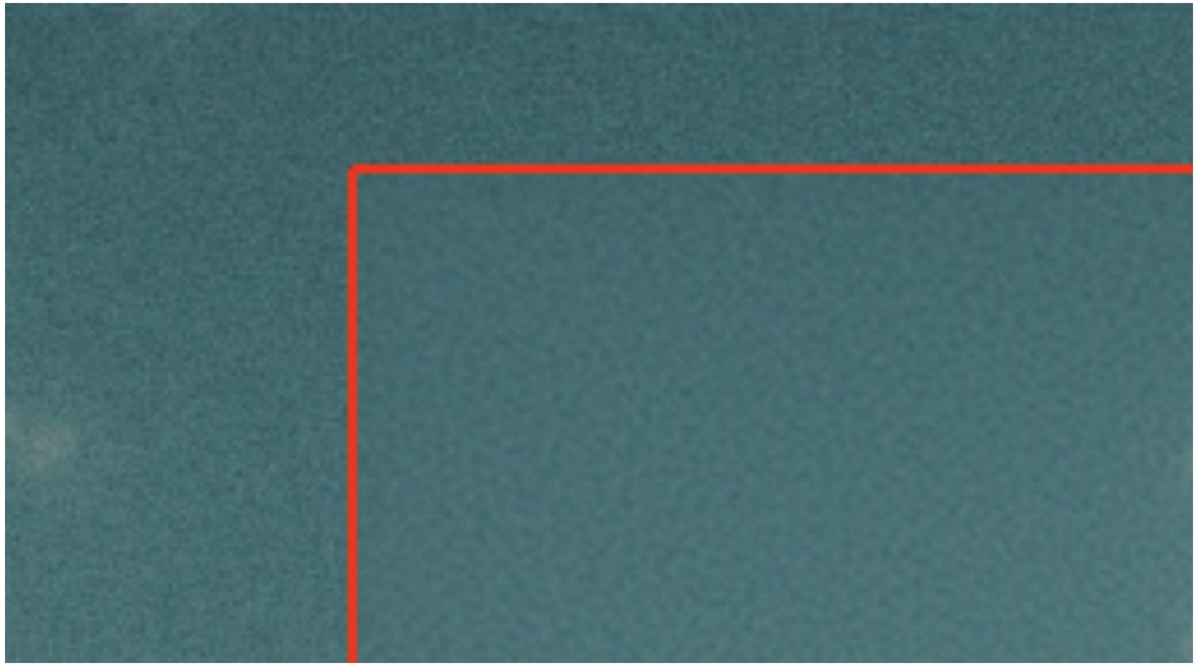


GenAI inpainting results



After applying our refiner

Noise Pattern Mismatch + Blurriness



GenAI inpainting results

After applying our corrector

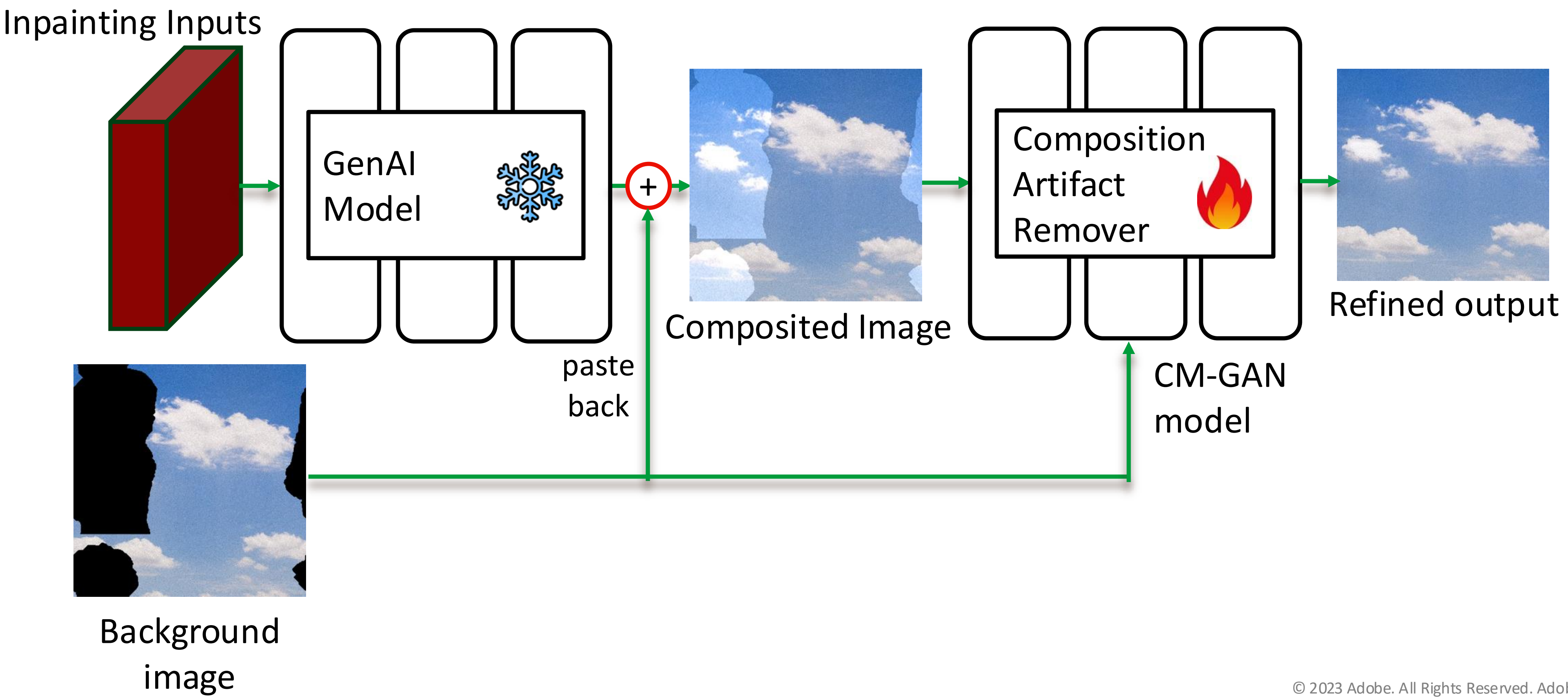
Adobe

Challenges in Combating Seam

- **How to support as many projects as possible**
  - Developing a pipeline that is agnostic to latent spaces and models choices.
- **Complex Artifact Mixtures**
  - The training need to replicate a mix of artifacts—color, textures, blurriness, sensor noise, and content mismatches.
- **Seams are subtle** even for human eyes (but it bothers expert users)
  - How to enhances model sensitivity to seam.

A Composition Artifact Refiner

- **Input**: 1) a composited image with artifacts 2) the composition mask
- **Output**: the refined image with artifacts fixed
- **Flexibility:** the pixel-space design makes the refiner agonistic to latent space and base model choices



A Paired-data Generation Pipeline

- Generates Clean-Artifact pairs on-the-fly during training
- Implemented a mixture of artifacts
  - Content truncation at boundaries
  - Uniform/Non-uniform/gradient color mismatches
  - Sensor noise pattern mismatch
  - Blurriness mismatch
  - JPEG artifacts mismatches

Noise/color/ blurriness mismatch

Clean Image

Content Truncation

Clean Image

Simulating Content Truncation

- On the boundary of the clean image, we randomly apply another inpainting model (CMGAN) during training on the input image, and paste back the background pixel for input image, so that input image contains cut-off seam, but output image is clean.

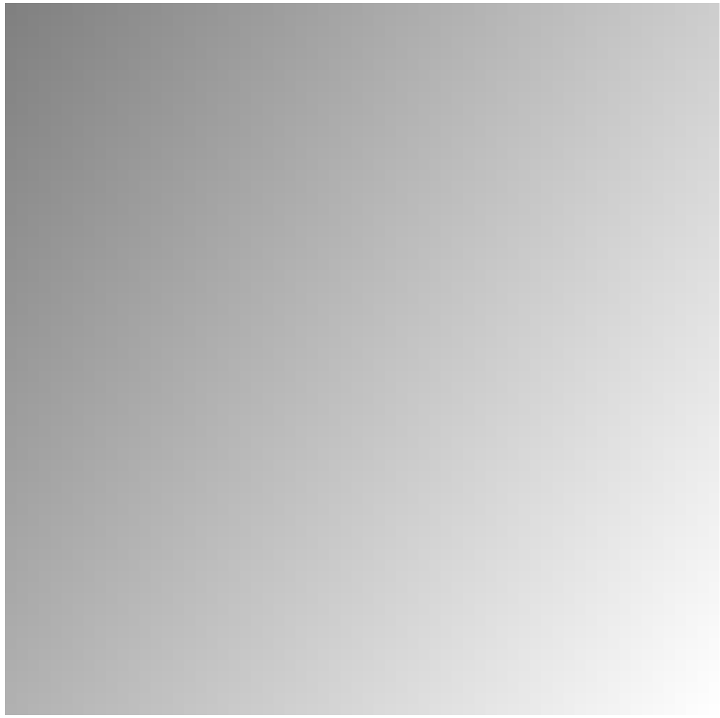
Content Truncation

Clean Image

Simulating Gradient Color Variant

1. uniform color augmentation
2. linear color augmentation

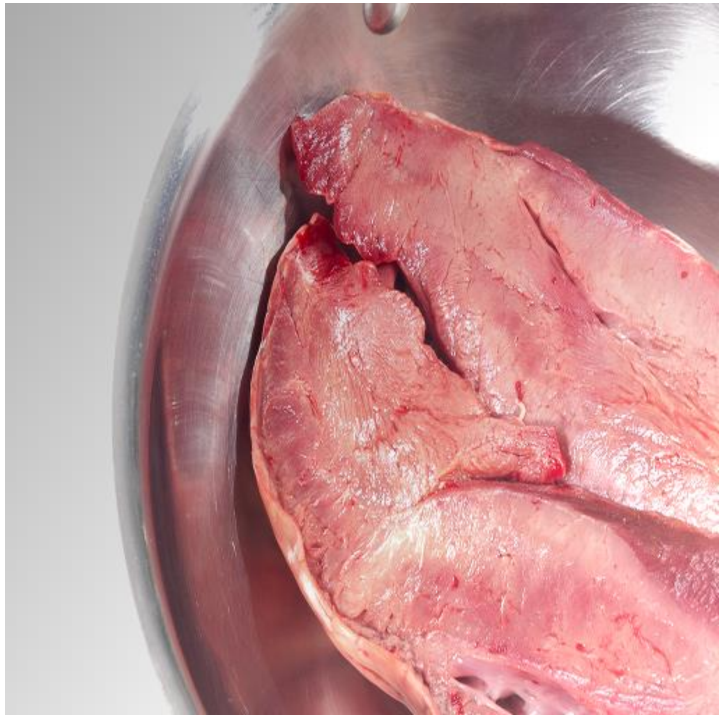
1. Generate two random color variant (hue, contrast, saturation augmentation on image)
2. Generate a random alpha-blending mask, which can be uniform, linear or non-local



Random Alpha Blending Mask

Color Jittering 1

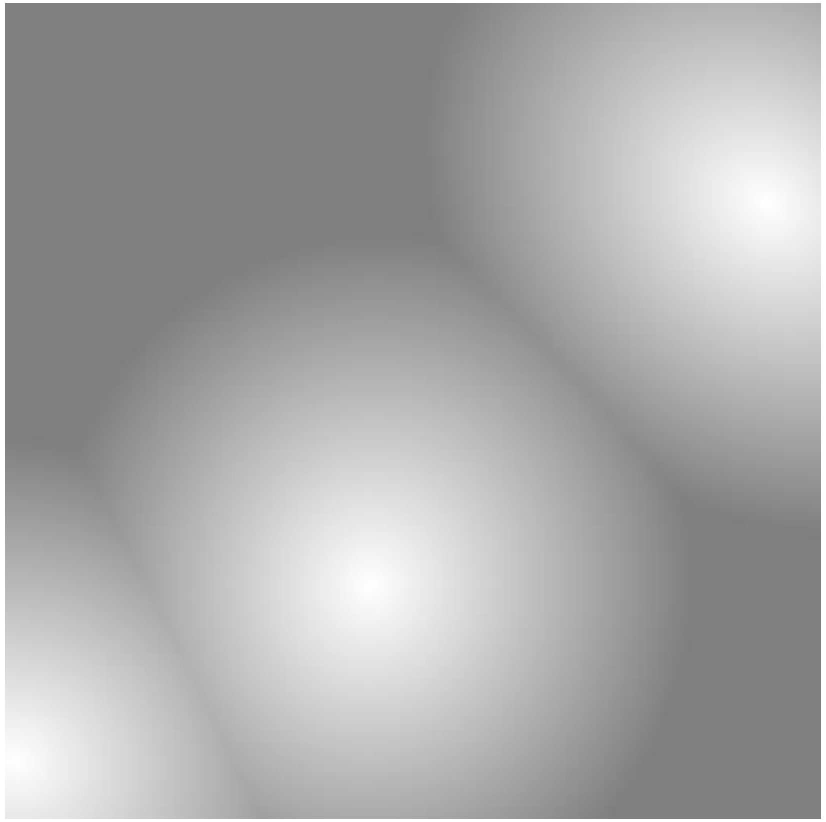
Color Jittering 2



Gradient Color Jittering Results

Adobe

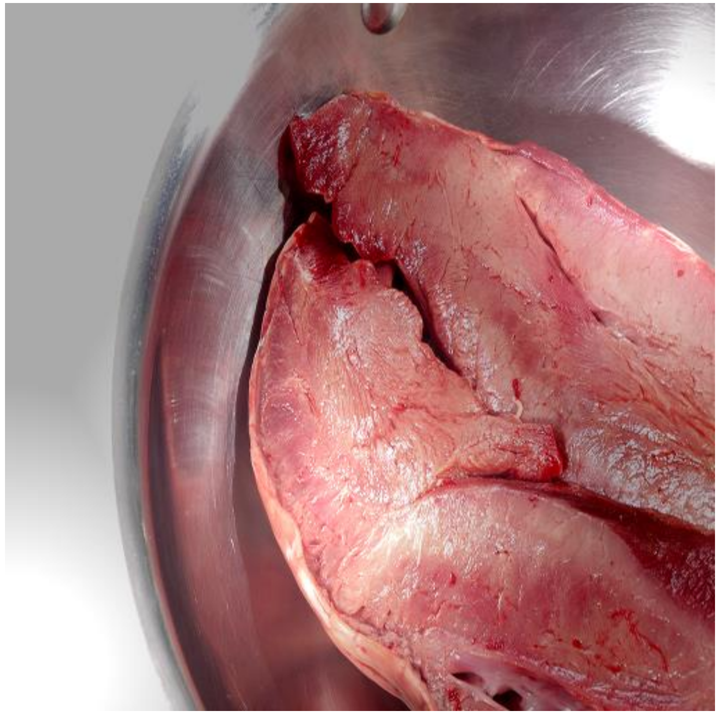
Simulating Non-uniform Color Variant



Random Alpha Blending Mask

Color Jittering 1

Color Jittering 2

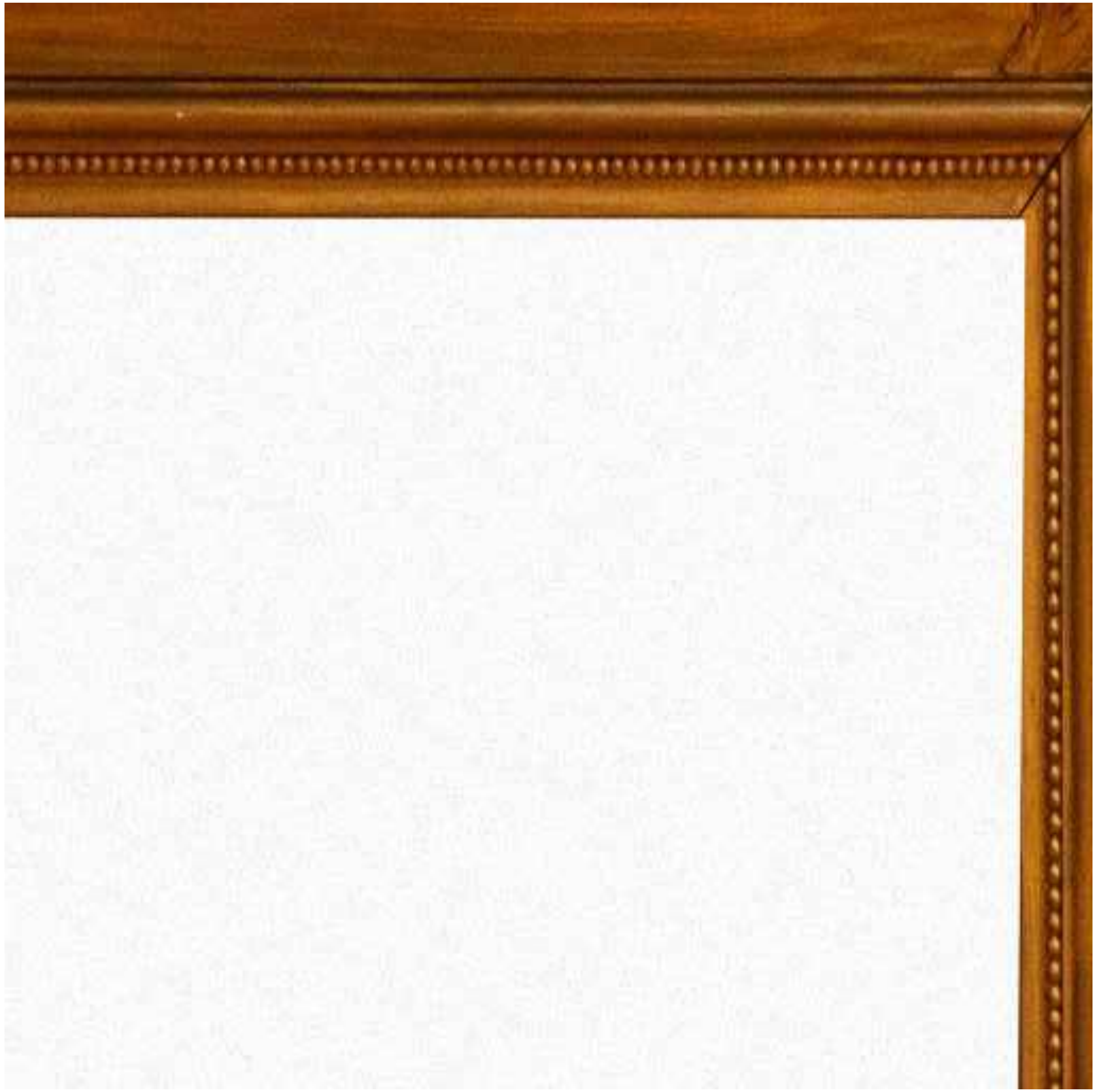


Gradient Color Jittering Results

Simulating the Realistic Sensor Noise Patterns

raw image (no noise) as the clean image

or



noisy image as the clean image

Adobe

Simulating Blurriness and Noise Pattern Mismatch

blurriness inside the hole

Image with artifacts

Clean Image

Simulating JPEG Artifacts Mismatch

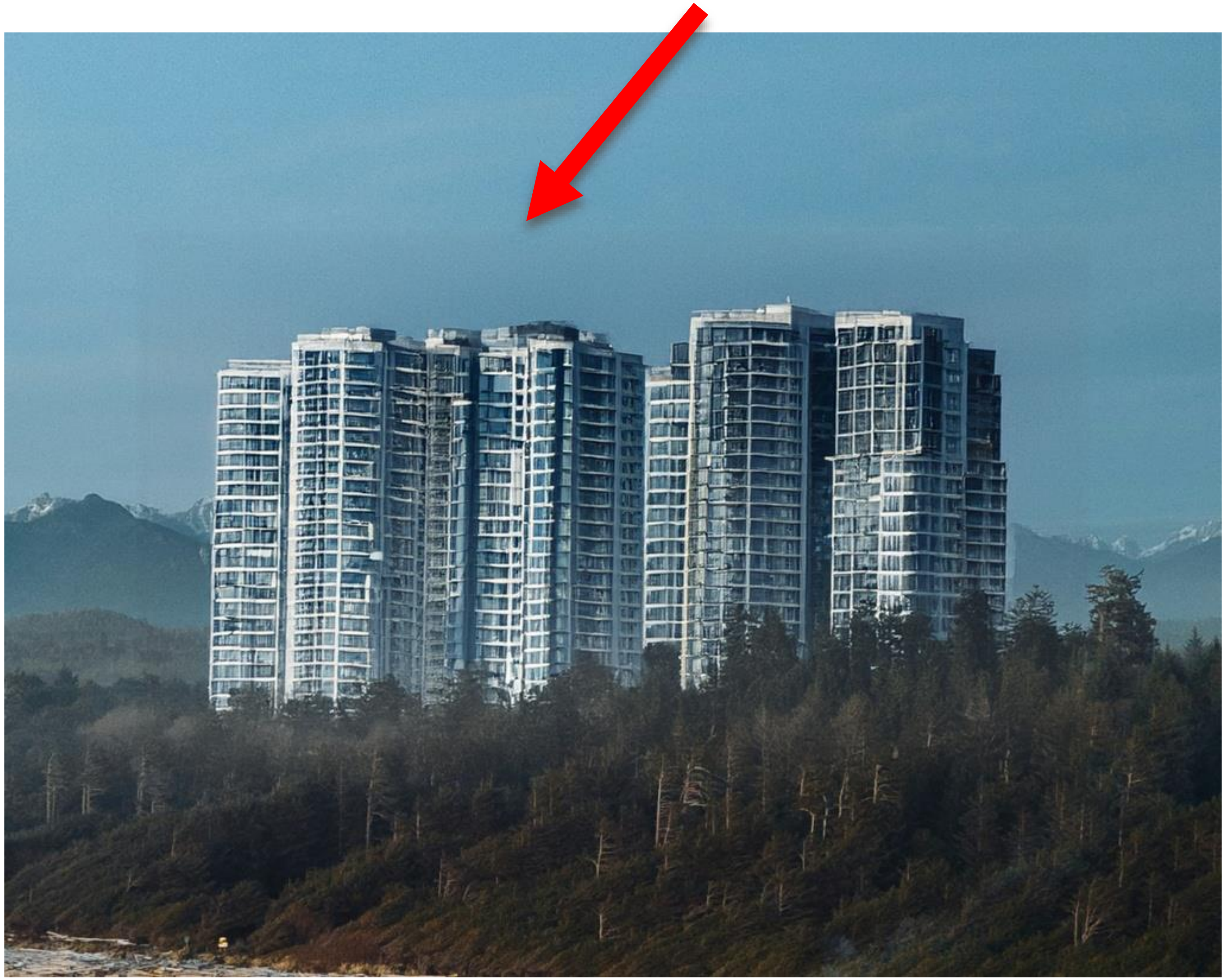
Image with Artifacts

Clean Image

Adobe

Naive Training Cannot Fix Subtle Color/Hue Difference!

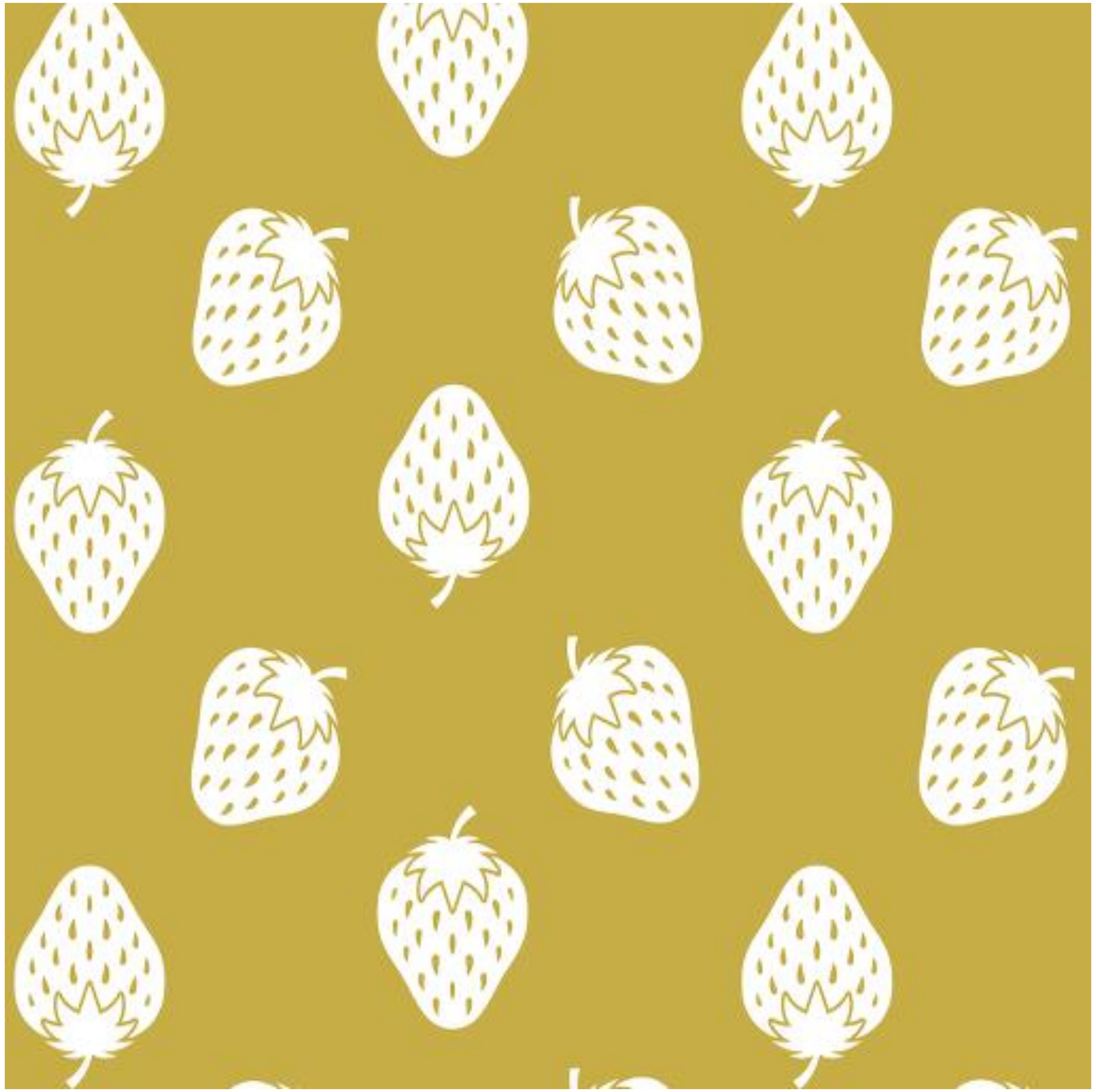
- Actually the tiny seam are extremely hard to correct for the refiner
- The perceptual loss and GAN loss is not sensitive to subtle color or hue difference
  - The color-shift is hard to eliminate for many models in the past due to that



Result before training with the invented loss

Adobe

A Real Example

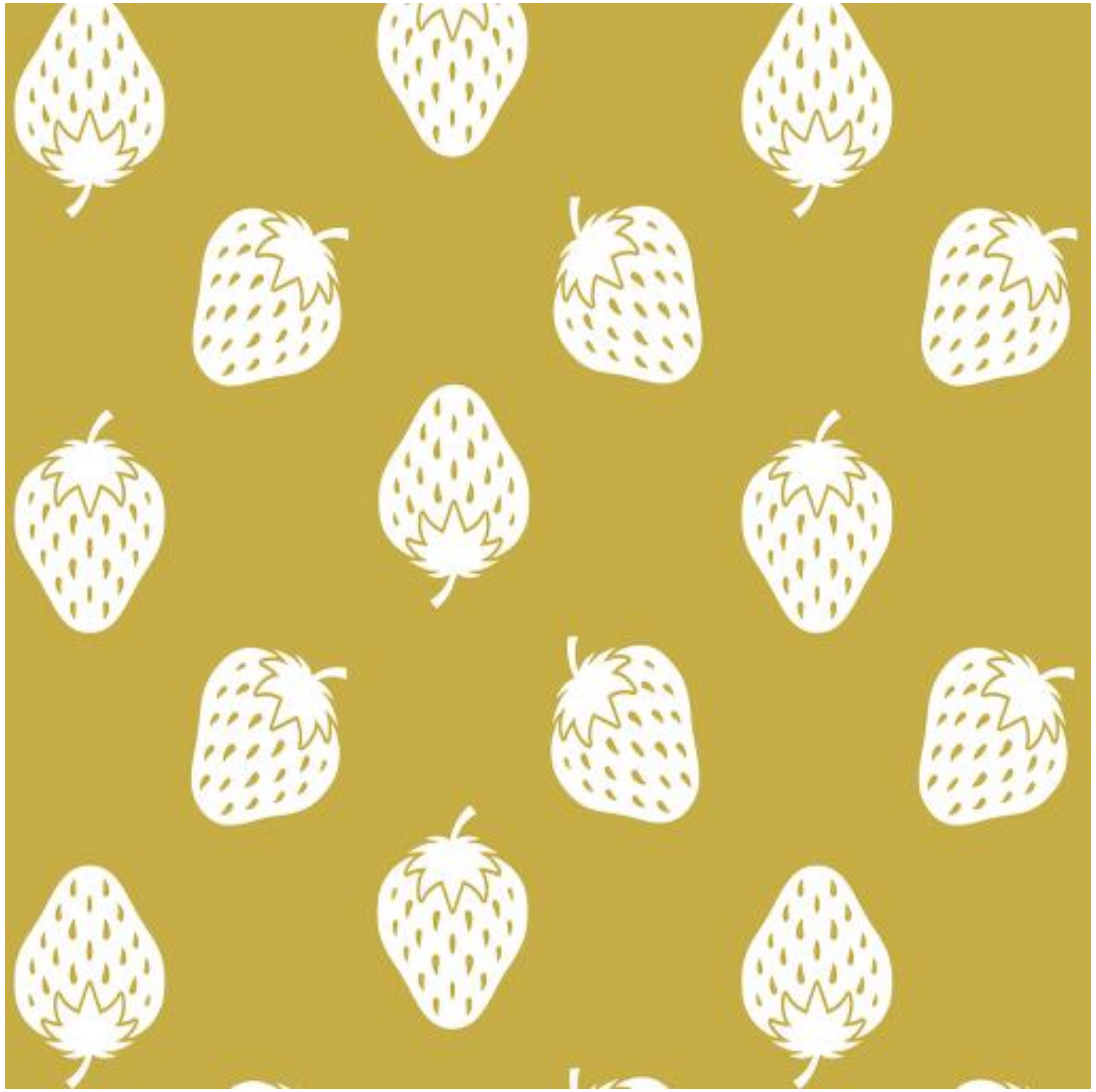


GT

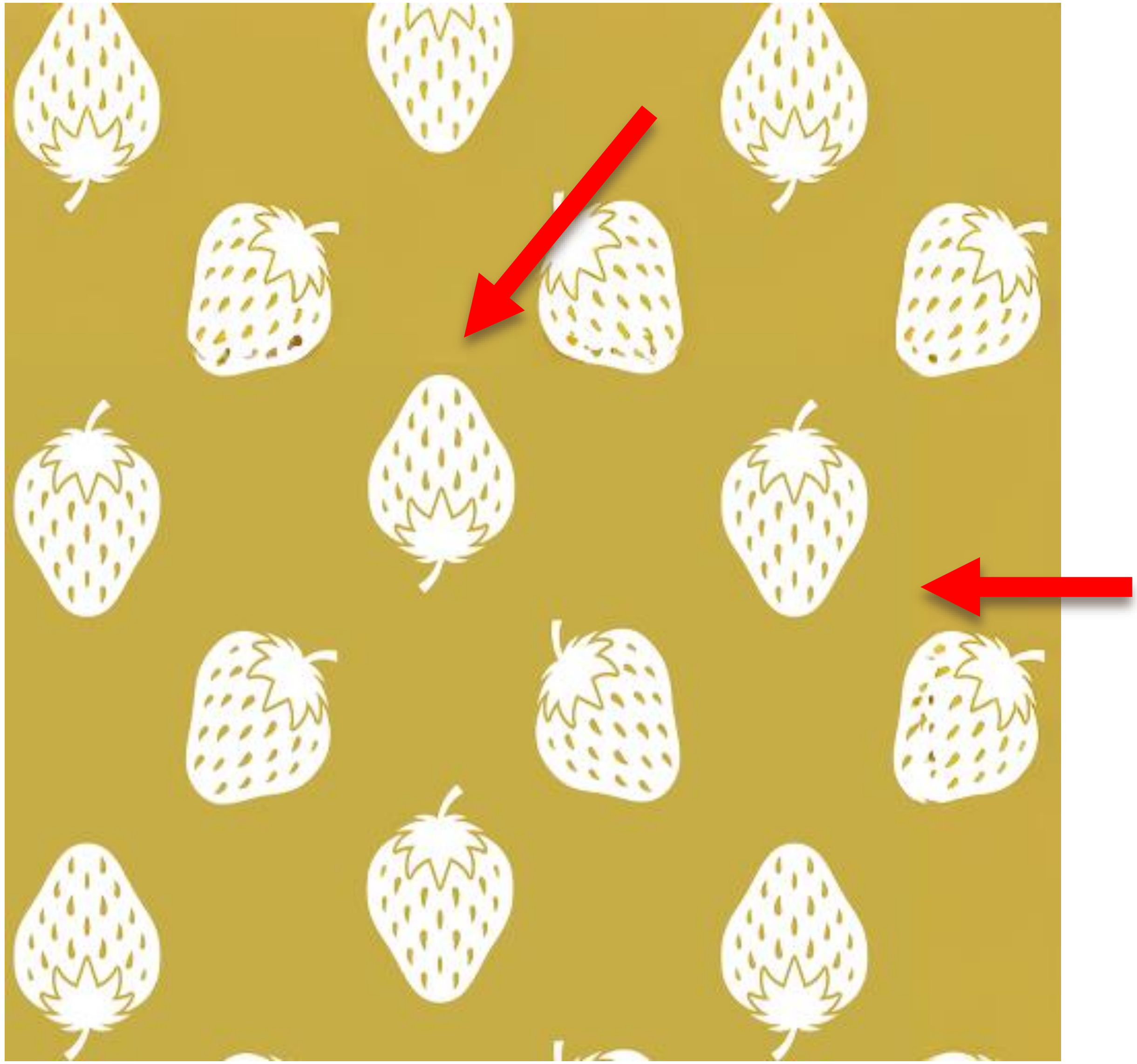
The refiner’s output with seam

Adobe

A Real Example



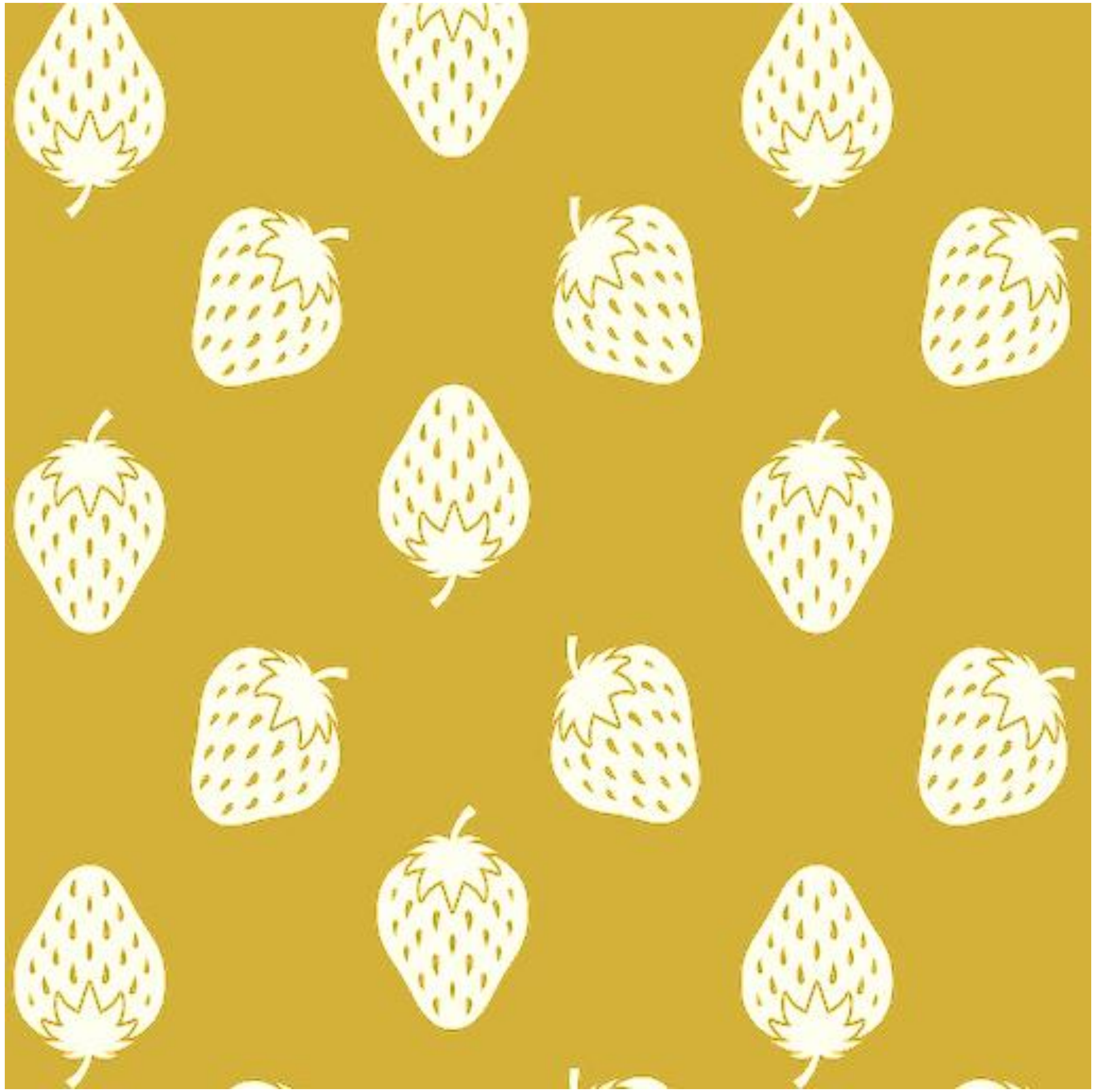
GT



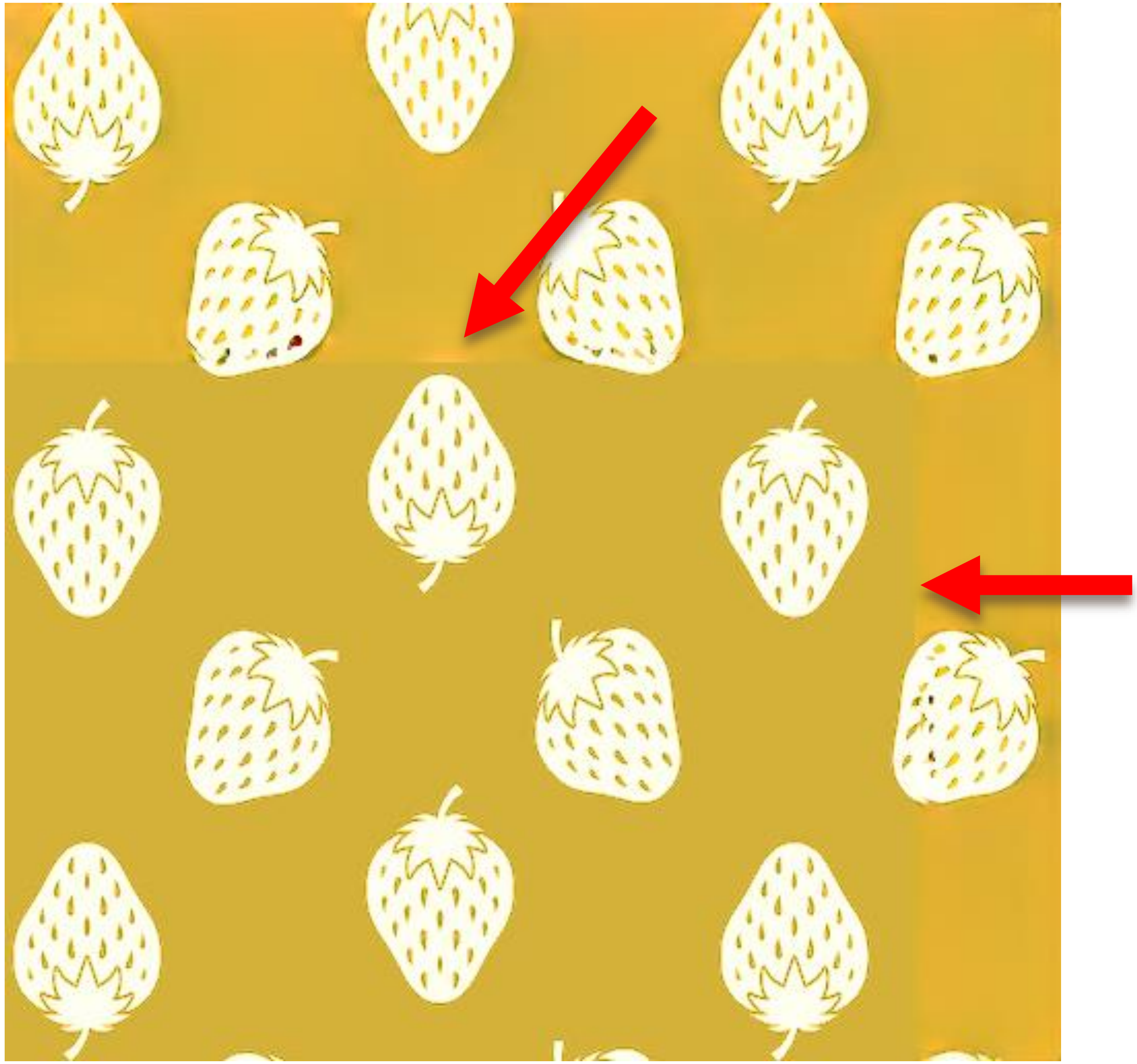
The refiner's output with seam

Adobe

Dynamic Color-Space Enhancement



GT



The refiner's output with seam

Adobe

The discriminative pixel space for improving harmonization

1. Given GT and output, generate an image amplified color artifacts.

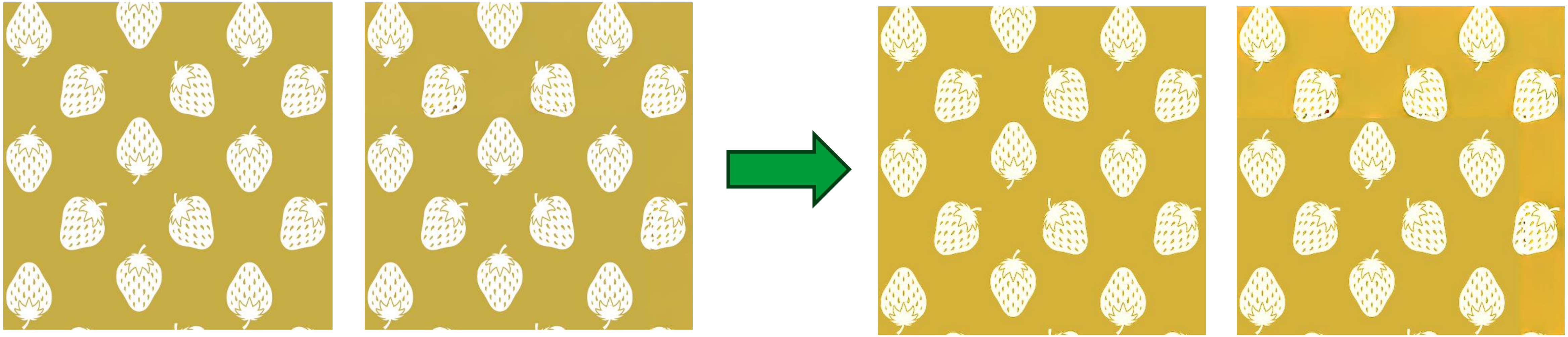
   amplified_output = gt + 20 * (output – gt)

2. Fit a polynomial function M() that maps the pixel value from output to amplified output.

   For background pixel, the mapping keeps the original color.

   For generated pixels, the mapping will amplify the color artifacts.

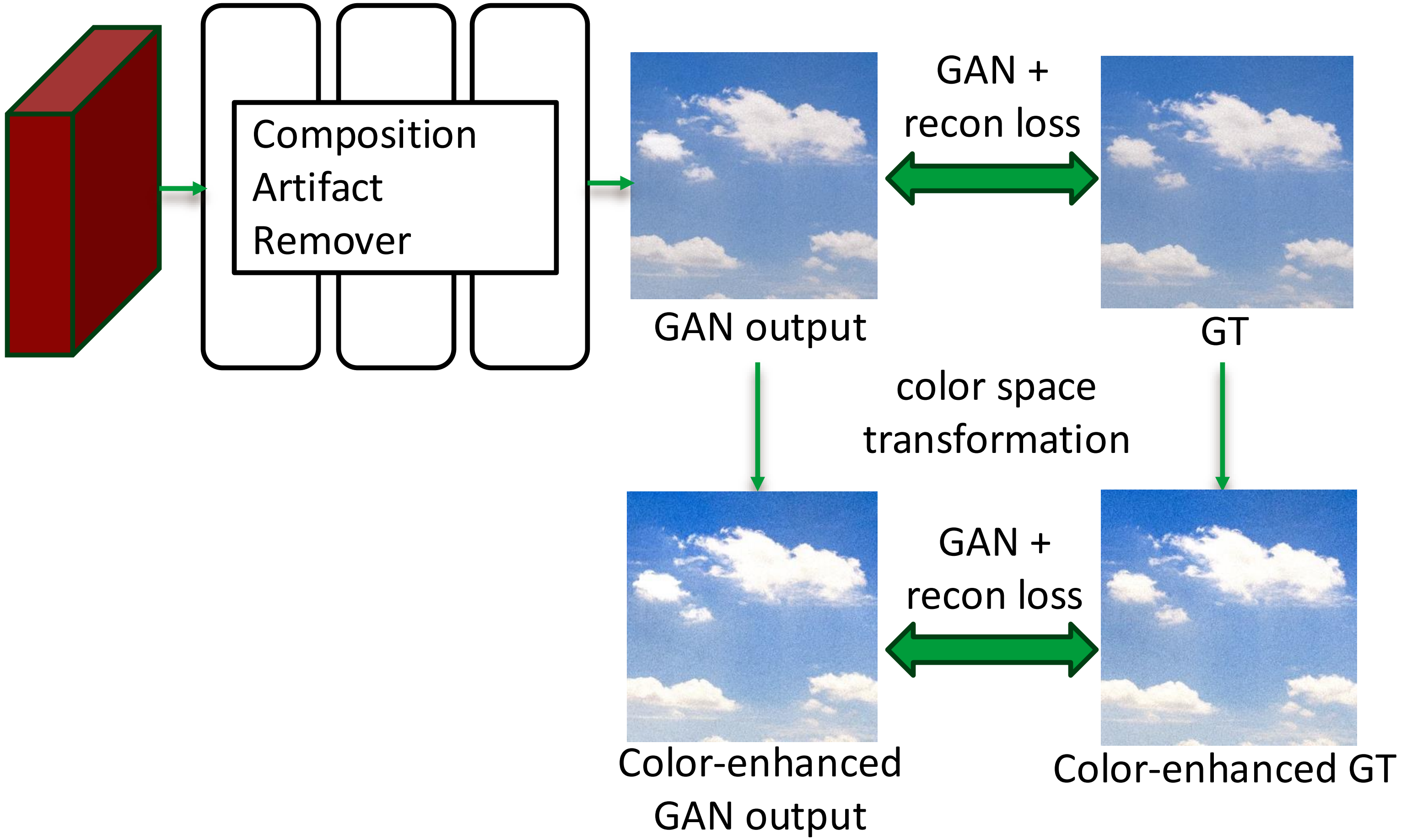
3. Compute L1 + perceptual loss between M(gt) and M(output).



gt   output   Enhanced gt   Enhanced output

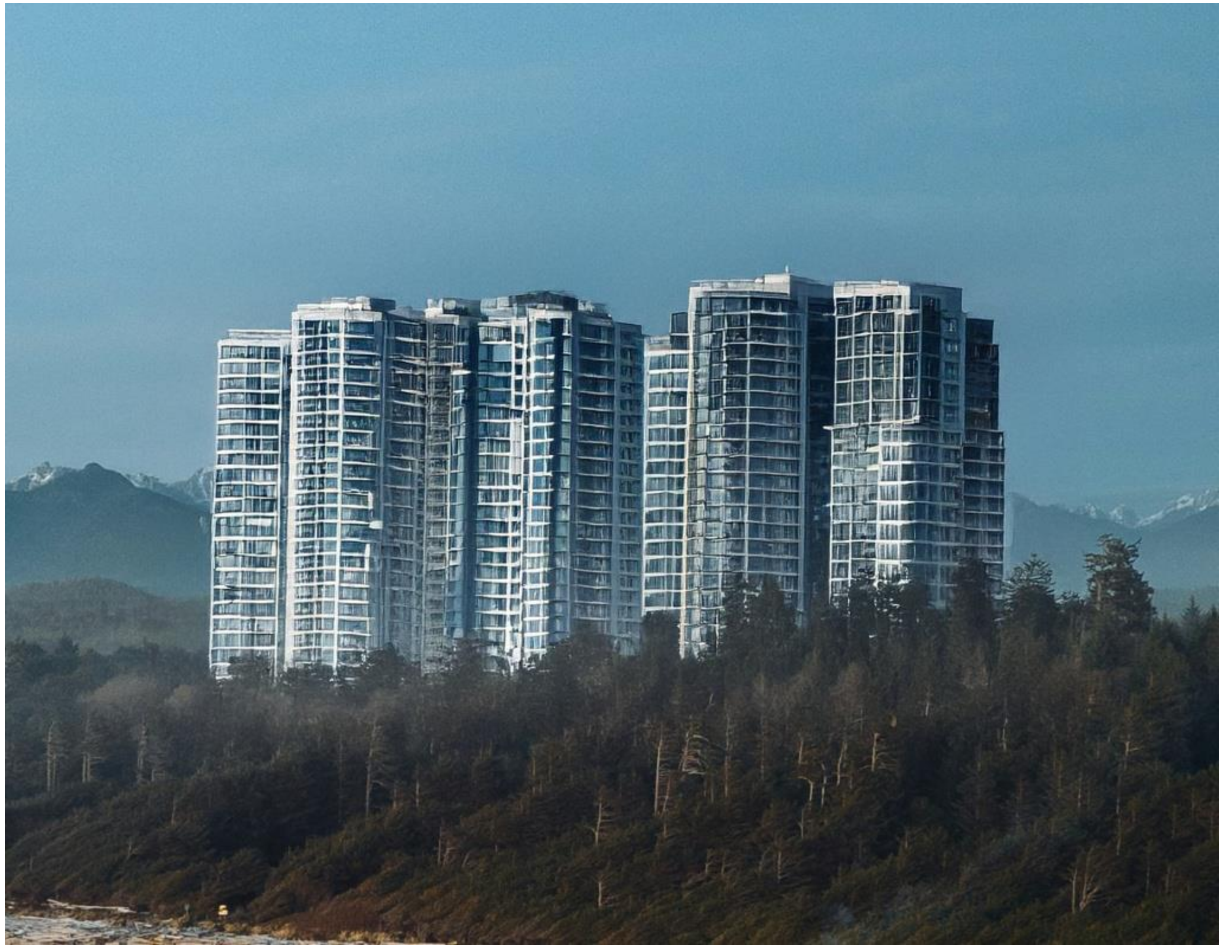
The Discriminative Pixel-Space Loss

- Applying perceptual and L1 loss on original and enhanced color space.



Comparison

w/o the discriminative pixel-space loss



w/ the discriminative pixel-space loss

Other Model Details

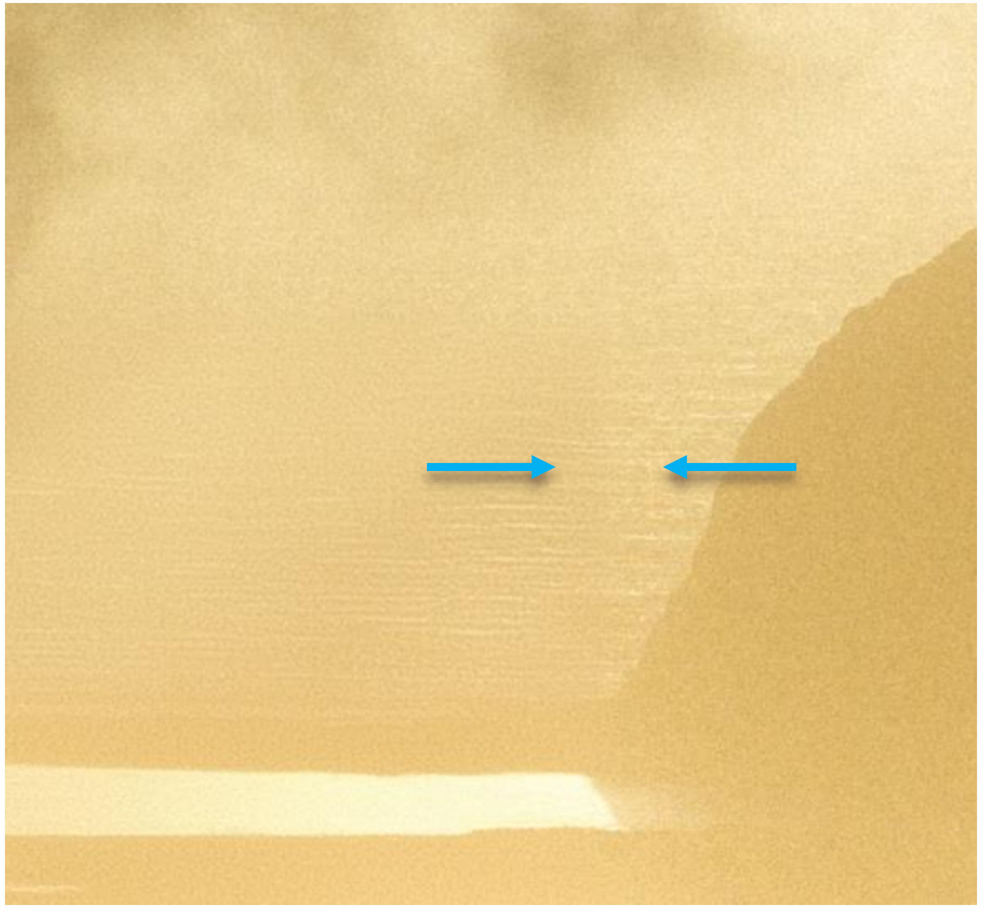
- We use the CMGAN model as the backbone for training / inference efficiency (4 nodes for training).
- Following the GigaGAN paper, we additionally add random noise to the input image to stabilize the training.
- The discriminator adds high-frequency component from clean image as condition.
- An inference trick: running the decoder N times, and use a heuristic to select from the best results.

Results on Inpainting

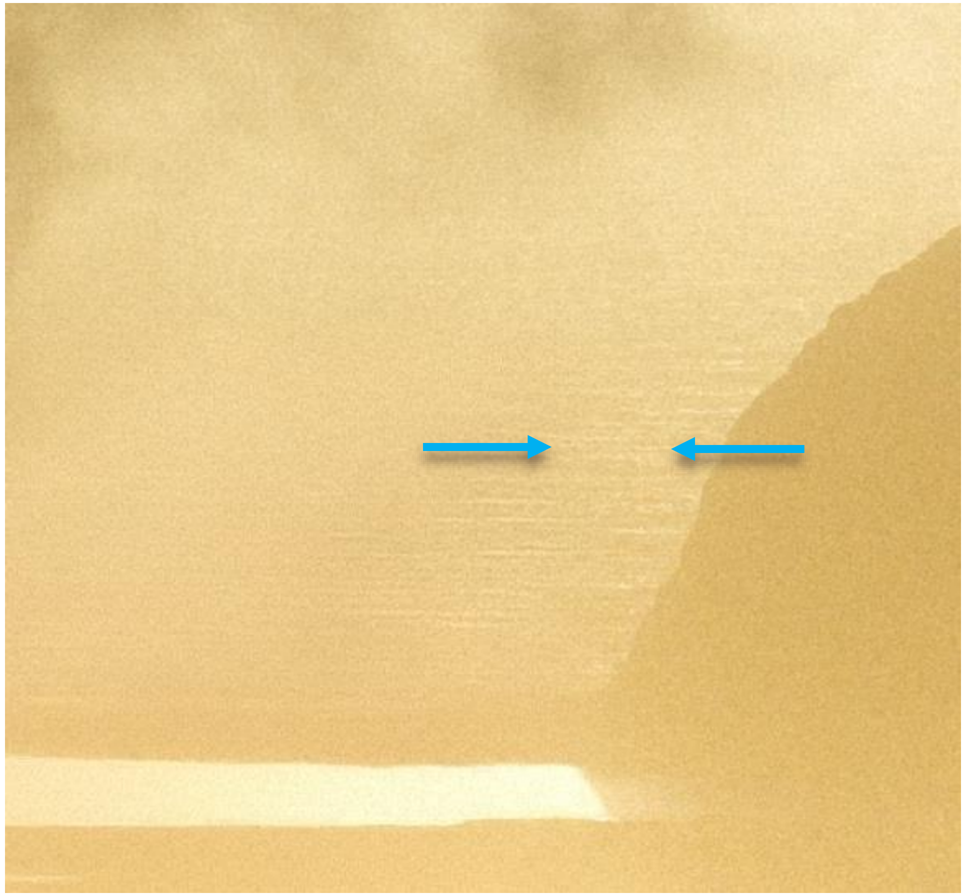
GenAI inpainting results

After seam removal

Results on Inpainting



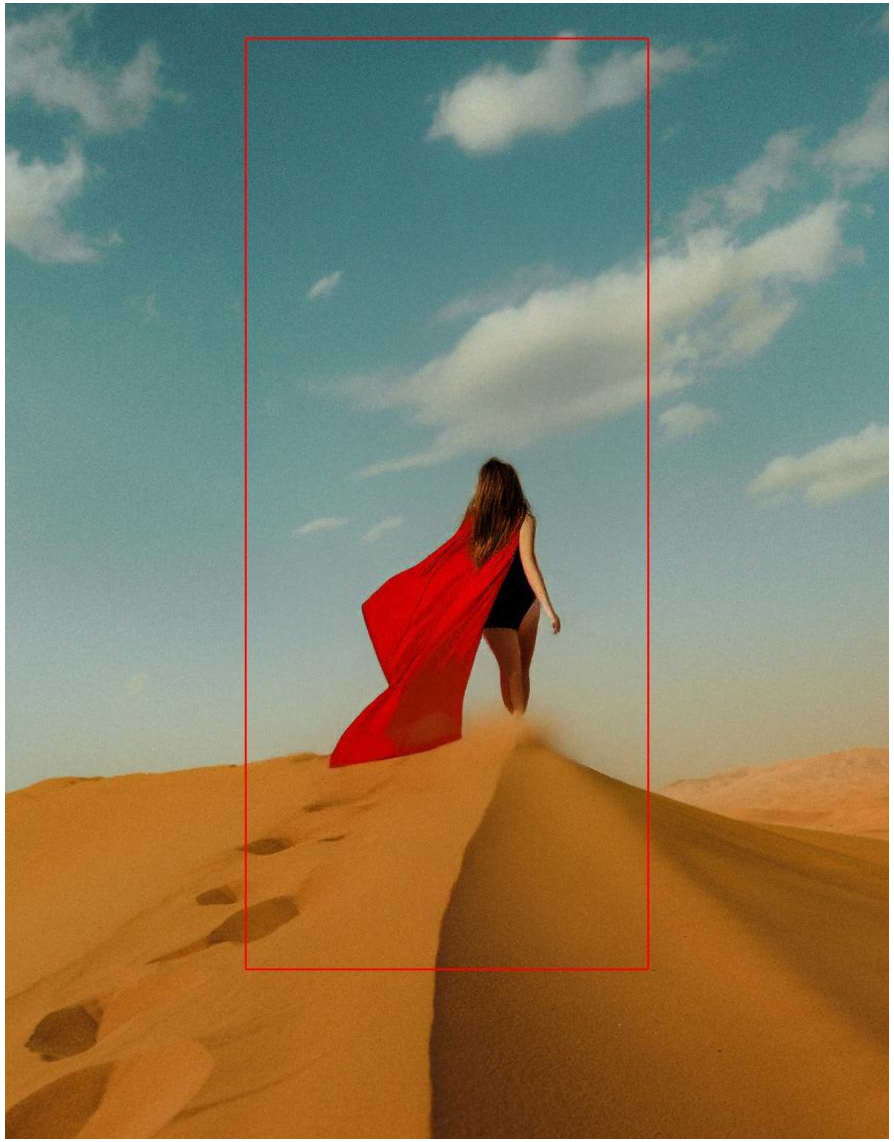
GenAI inpainting results

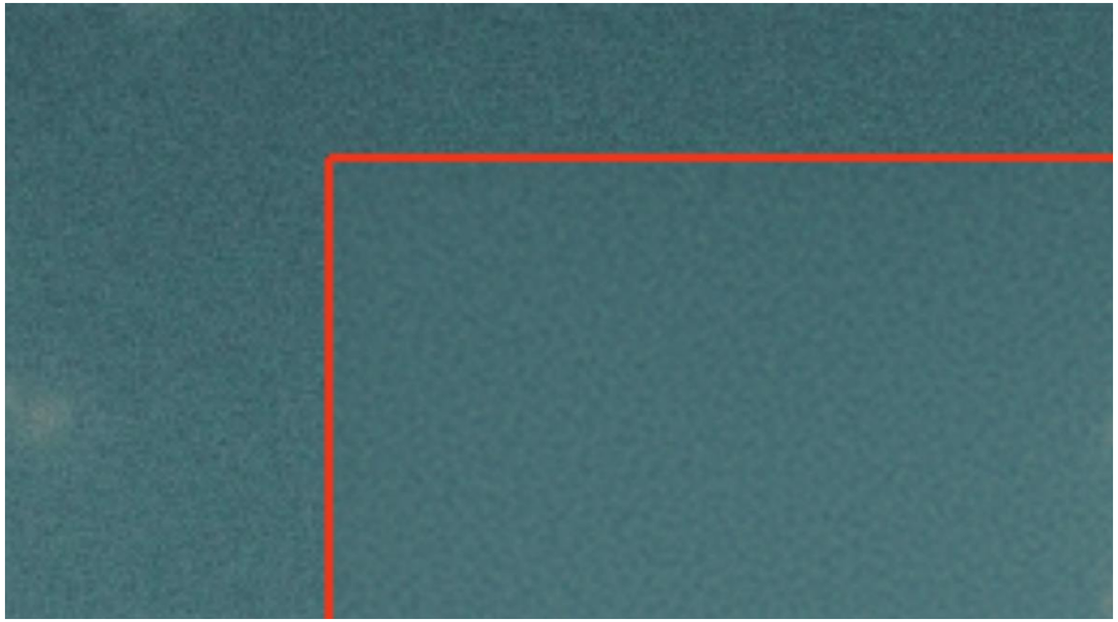


after applying our corrector

Results on Inpainting

- Better noise pattern matching

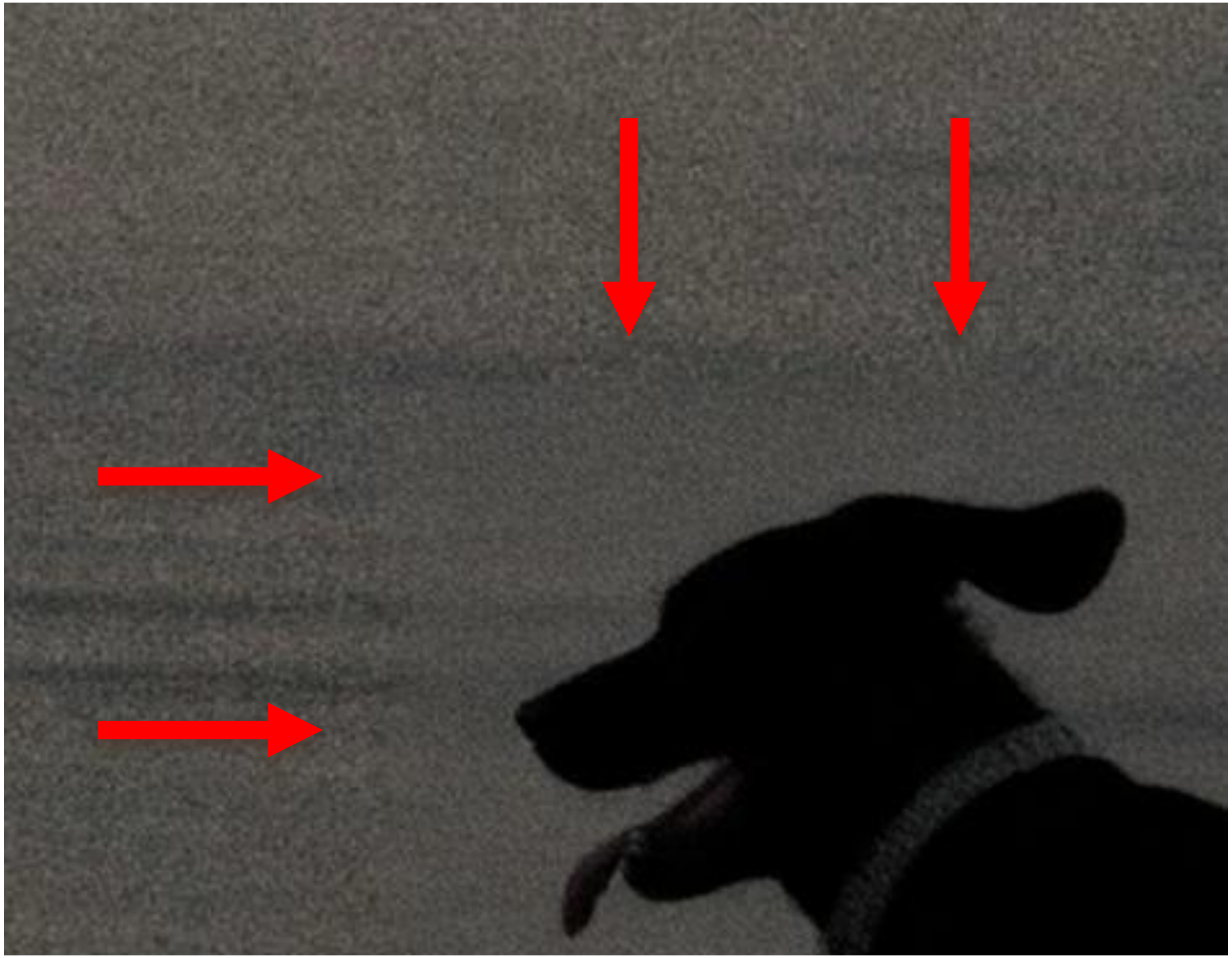




Original output + pixel pasting

Final output

Inpainting results



Inpainting results w/ PixelPerfect