Style and Content, Texture Synthesis

Jun-Yan Zhu

16-726, Spring 2023

Many slides are borrowed from Alyosha Efros and Bill Freeman

Collection Style Transfer

Photograph ©Alexei Efros

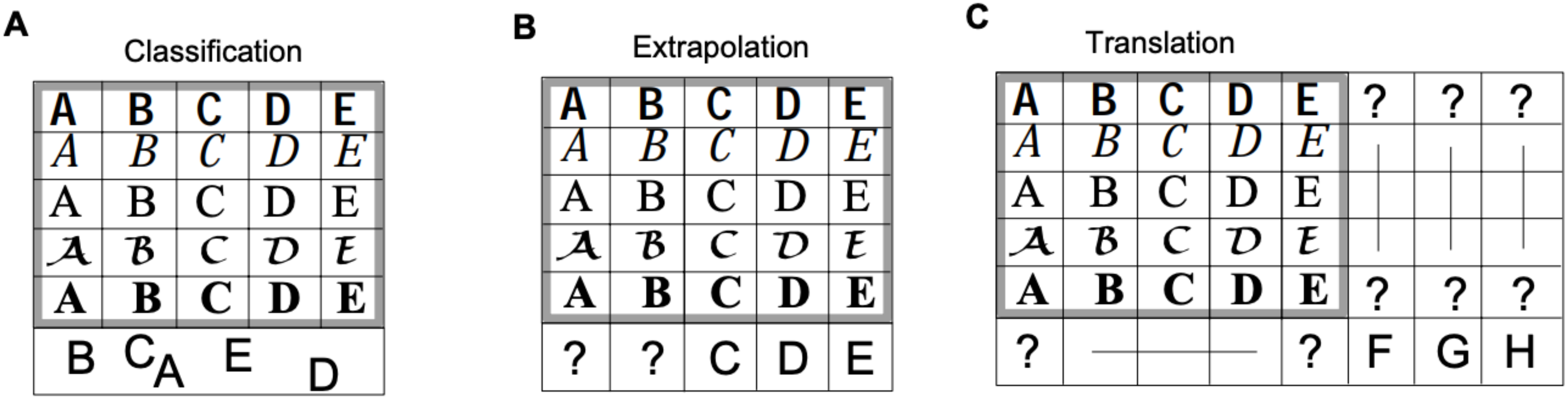
Monet

Van Gogh

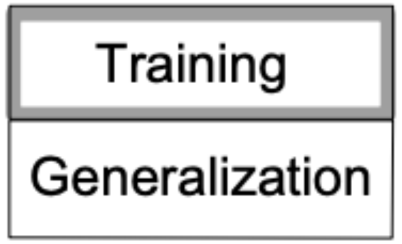
Cezanne

Ukiyo-e

Style and Content Separation



Domain Adaptation   Paired Image-to-Image Translation   Unpaired Image-to-Image Translation



Separating Style and Content
[Tenenbaum and Freeman 1996]

$$y_k^{sc} = \sum_{i=1}^{I} \sum_{j=1}^{J} w_{ijk} a_i^s b_j^c .$$

Style and Content

Adversarial loss

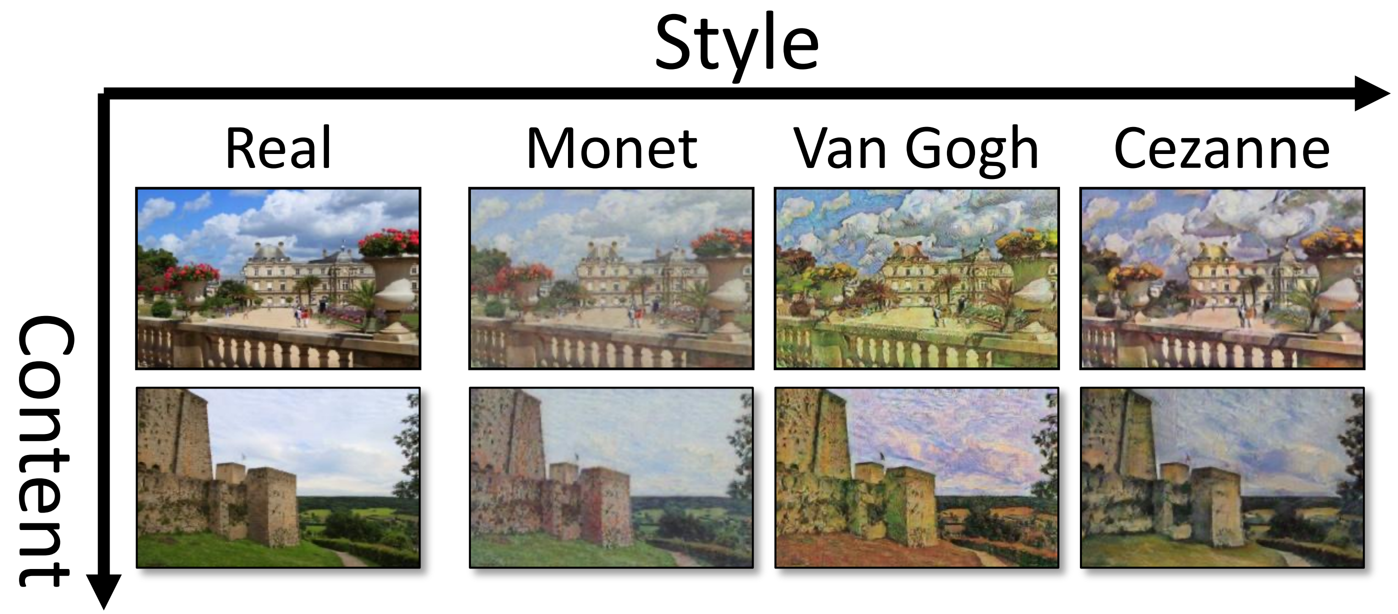
$$\mathbb{E}_x \log(1 - D_Y(G(x))) + \mathbb{E}_y \log D_Y(y)$$

$p(x) \rightarrow p(y)$ change **style**

Cycle-consistency loss

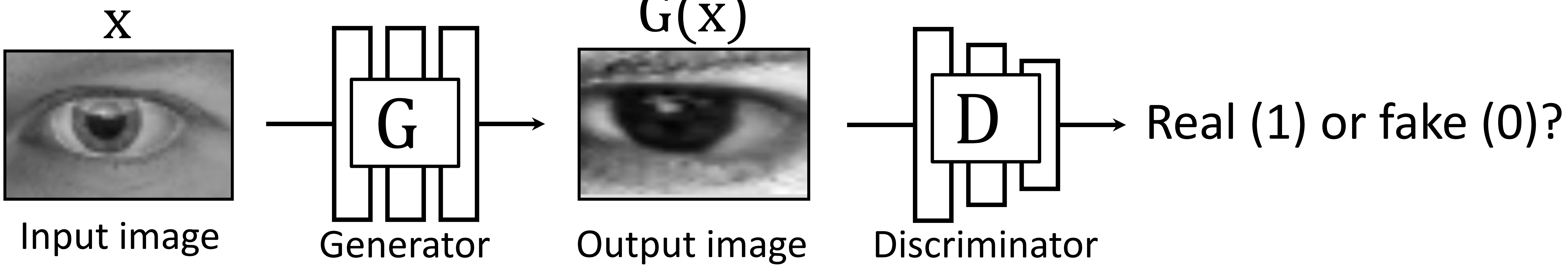
$$\mathbb{E}_x ||F(G(x)) - x||_1$$

Bidirectional: preserve **content**



Separating Style and Content
[Tenenbaum and Freeman 1996]

Style and Content

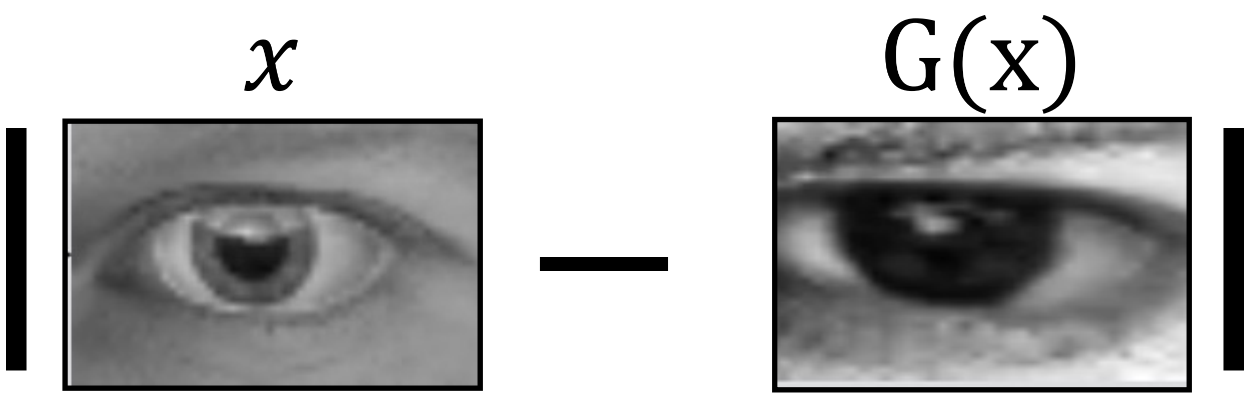


Adversarial loss (change style)

$$\mathbb{E}_x \log(1 - D_Y(G(x))) + \mathbb{E}_y \log D_Y(y)$$

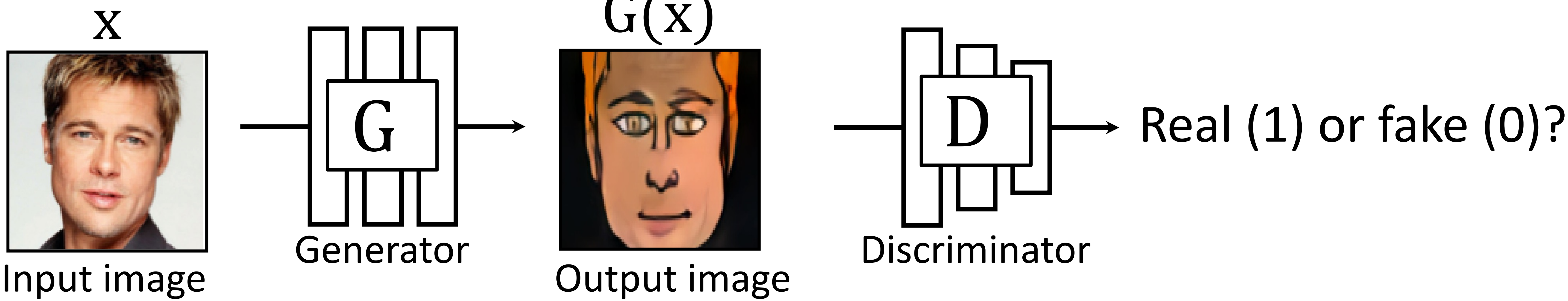
L1 loss (preserve content in pixel space)

$$\mathbb{E}_x ||G(x) - x||_1$$



SimGAN [Shrivastava et al., 2017]

Style and Content

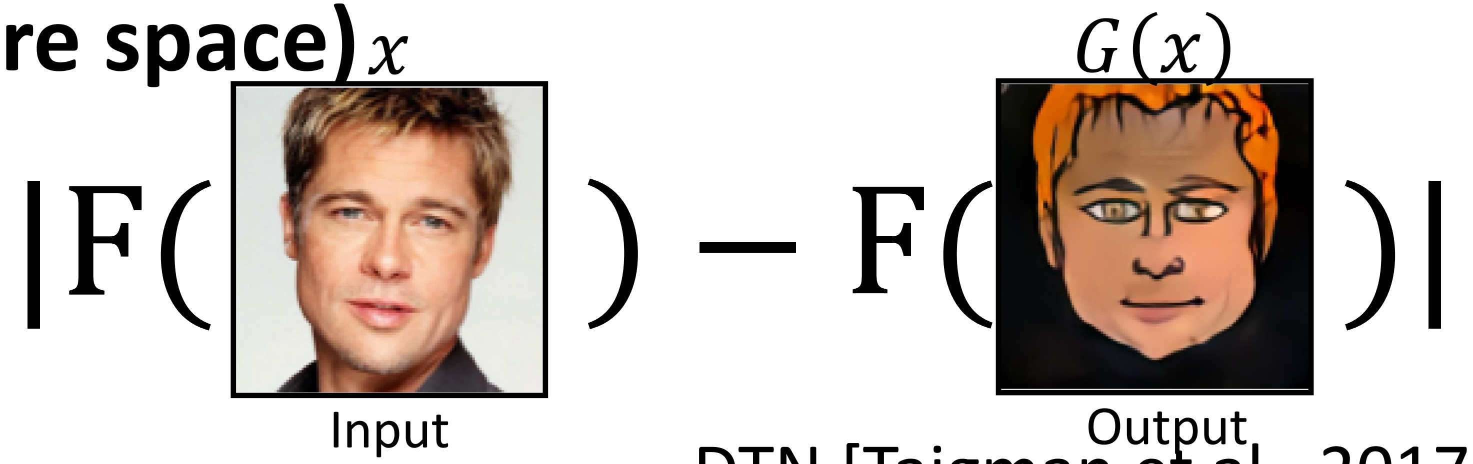


Adversarial loss (change style)

$$\mathbb{E}_x \log(1 - D_Y(G(x))) + \mathbb{E}_y \log D_Y(y)$$

Feature loss (Preserve content in feature space)

$$\mathbb{E}_x ||F(G(x)) - F(x)||$$



DTN [Taigman et al., 2017]

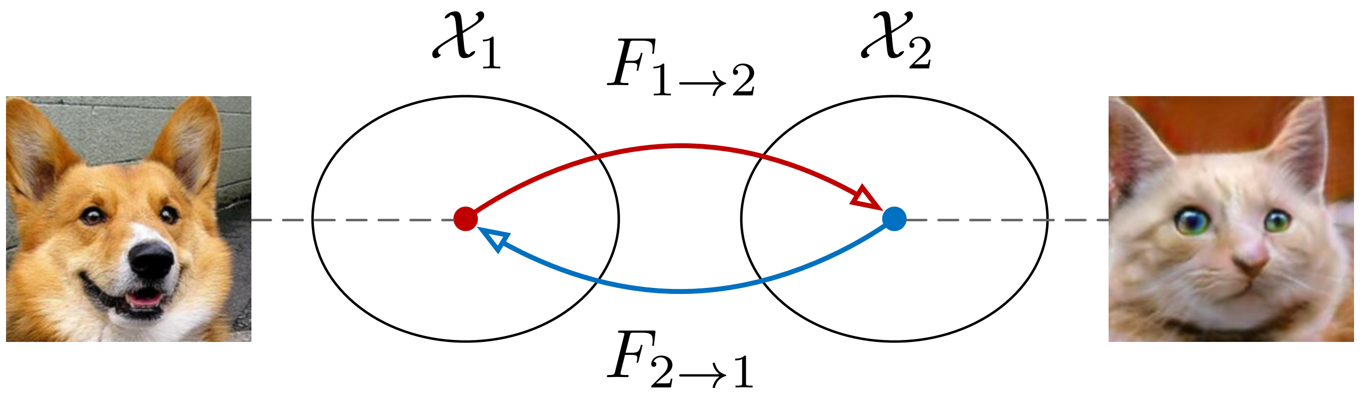
Style and Content

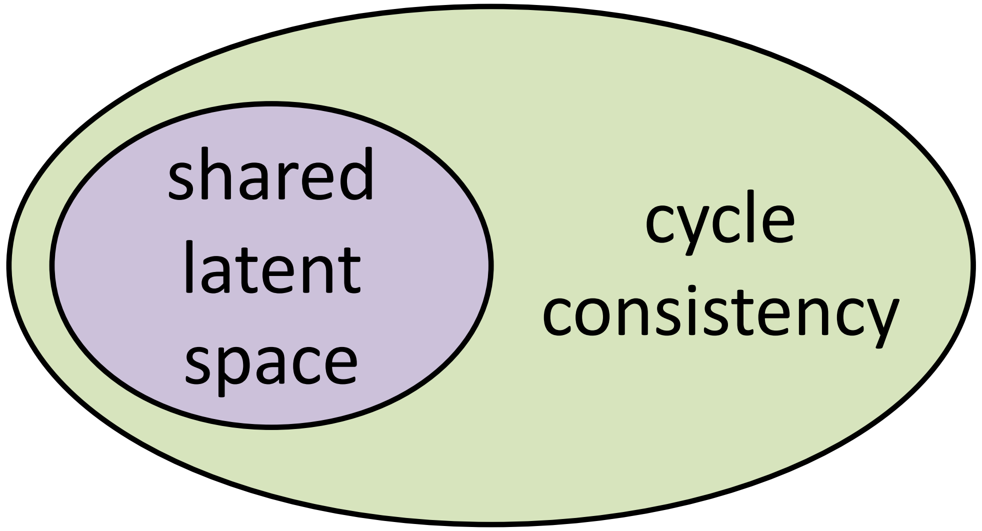
- <u>Style</u>: domain-specific features

  (horse vs. zebra)
- <u>Content</u>: features shared across two domains

CycleGAN and UNIT

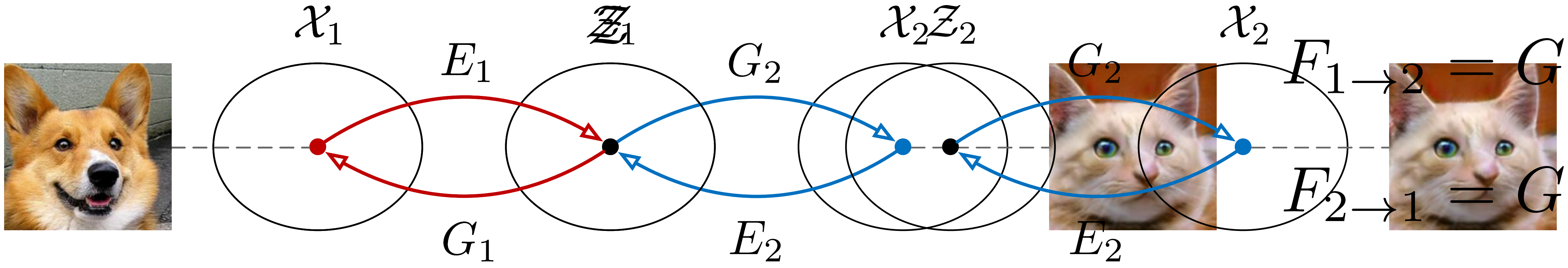
- CycleGAN (**cycle consistency**)





- UNIT (**shared latent space**) [Liu et al. 2017]

shared latent space $\Longrightarrow$ cycle consistency

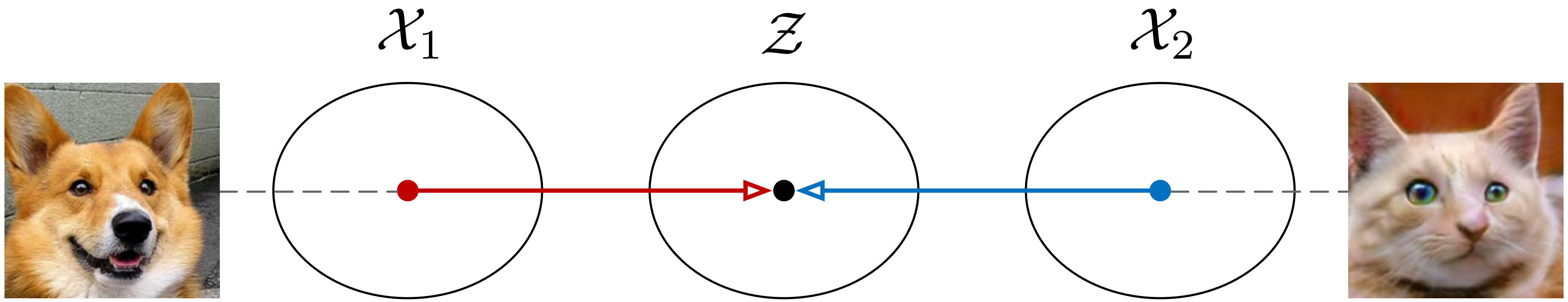


$$F_{1\to 2} = G_2 \circ E_1$$

$$F_{2\to 1} = G_1 \circ E_2$$

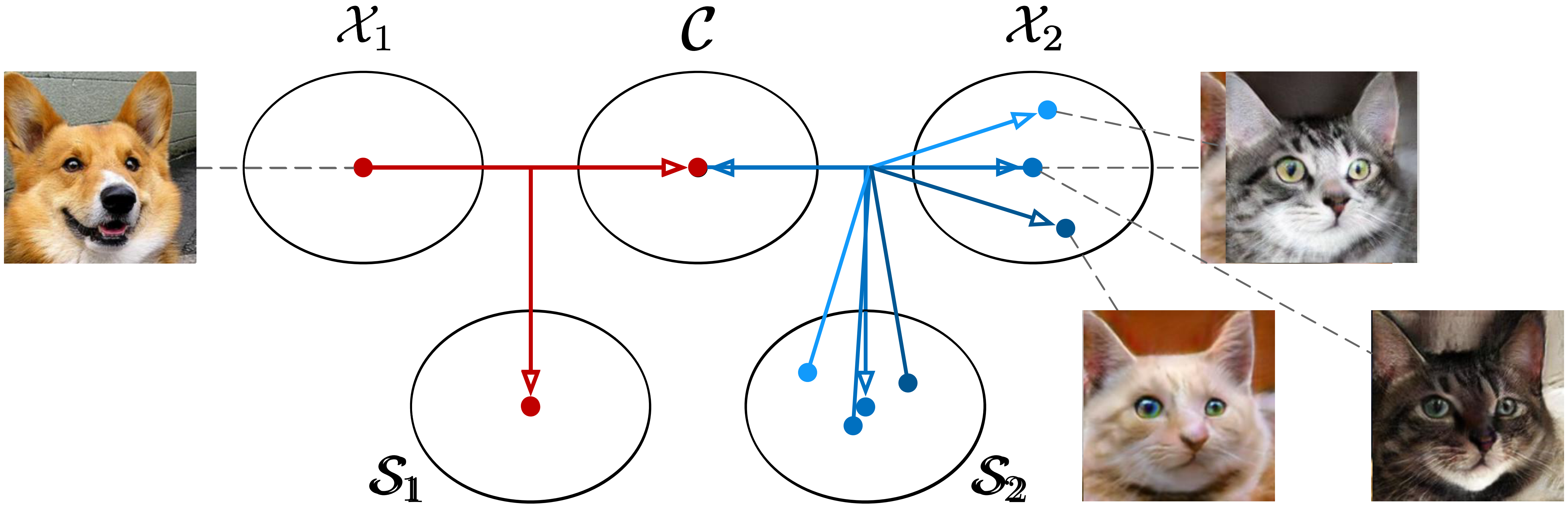
Disentangling the Latent Space

- UNIT
  - A single **shared, domain-invariant** latent space $\mathcal{Z}$



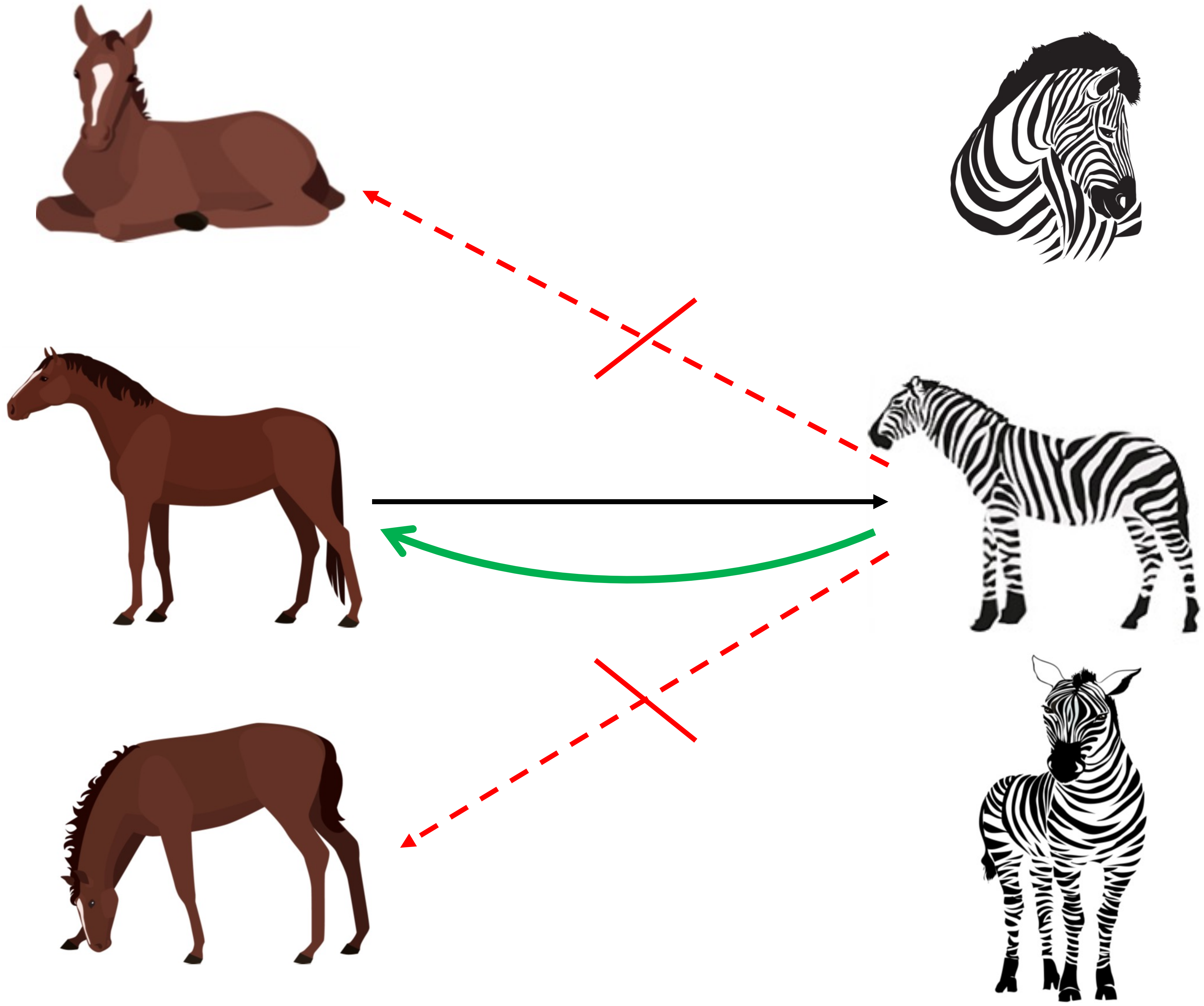
Disentangling the Latent Space

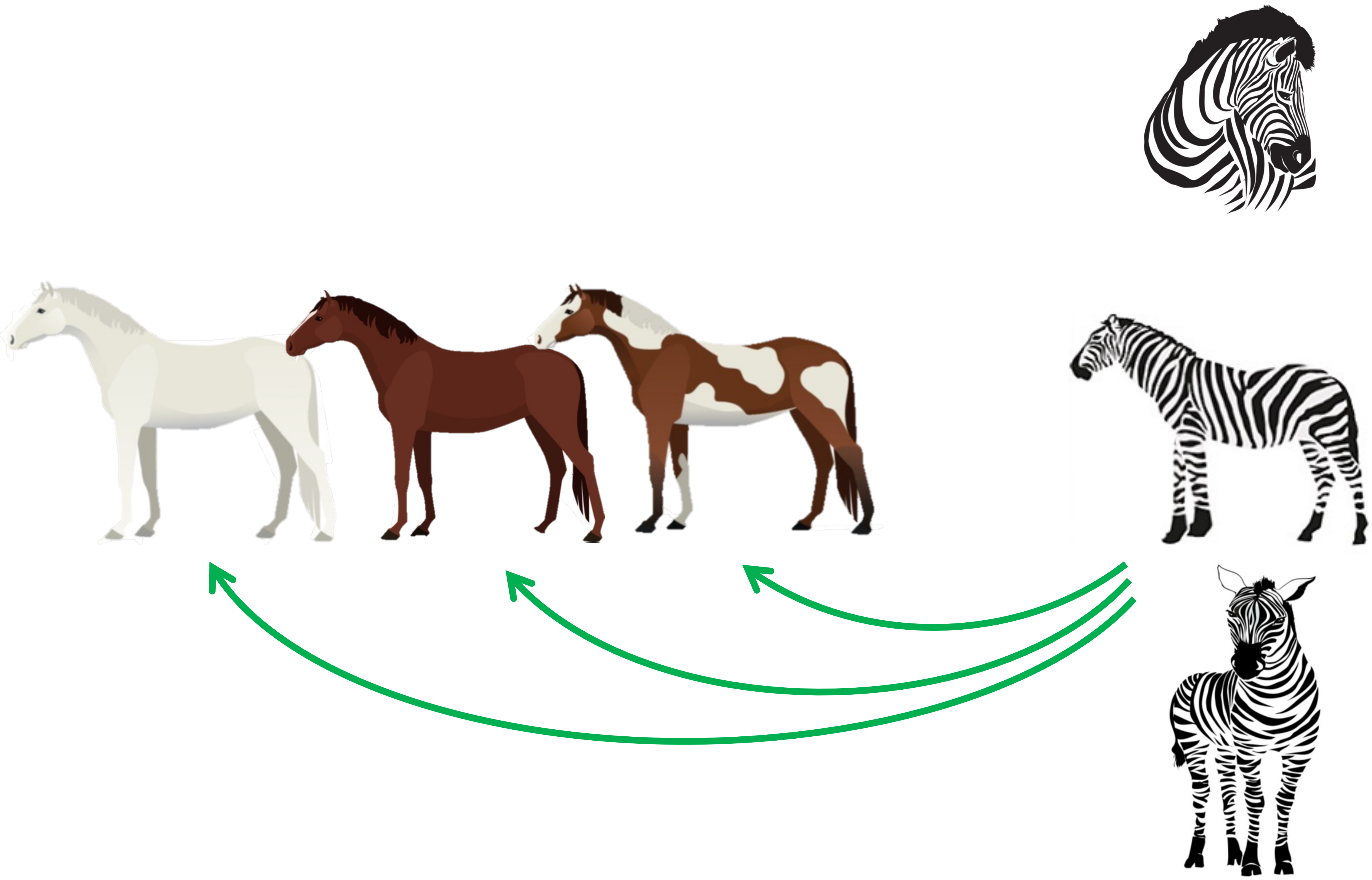
- Multimodal UNIT (MUNIT)
  - A **content** space $\mathcal{C}$ that is **shared, domain-invariant**
  - Two **style** spaces $\mathcal{S}_1, \mathcal{S}_2$ that are **unshared, domain-specific**

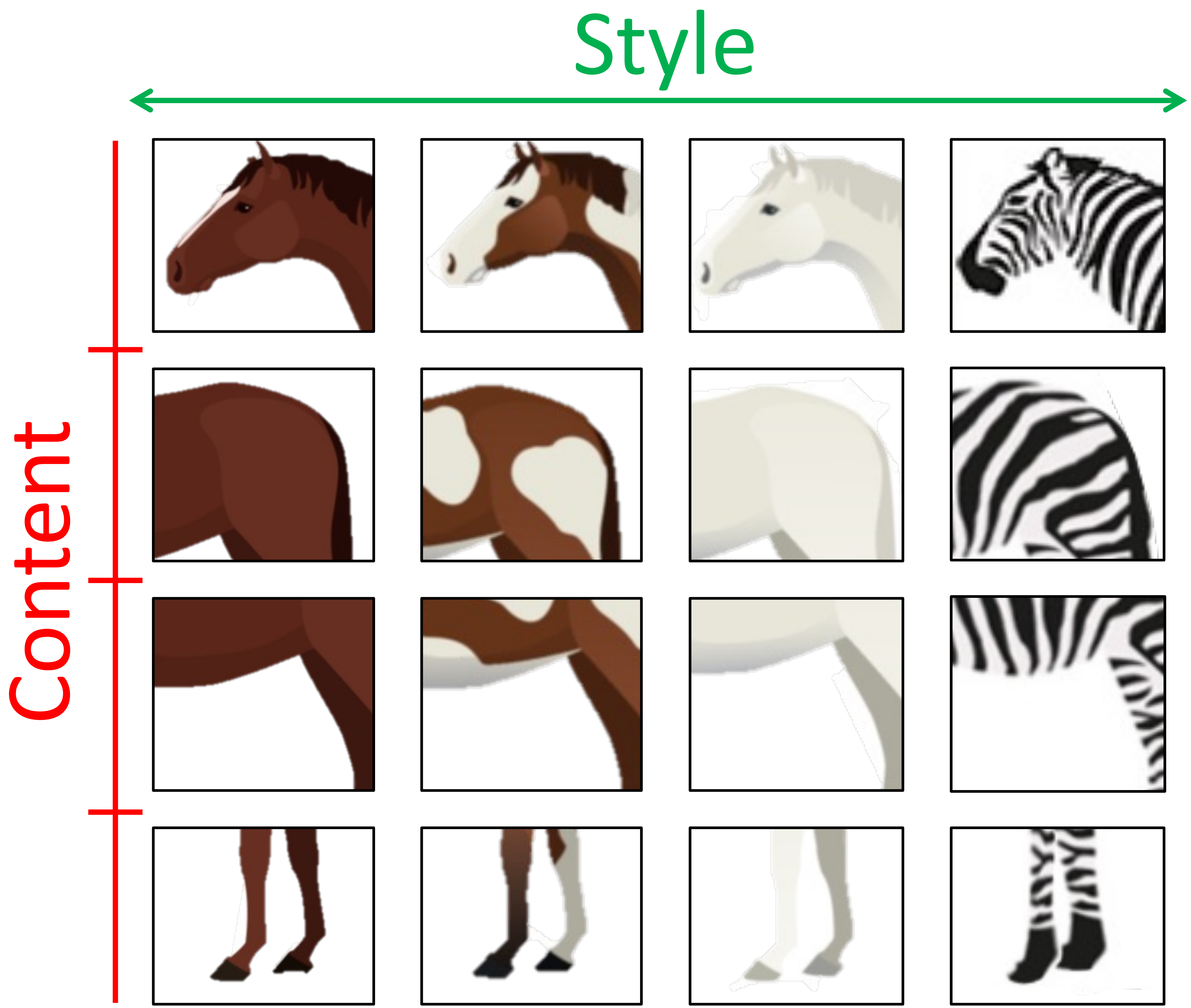


Style and Content

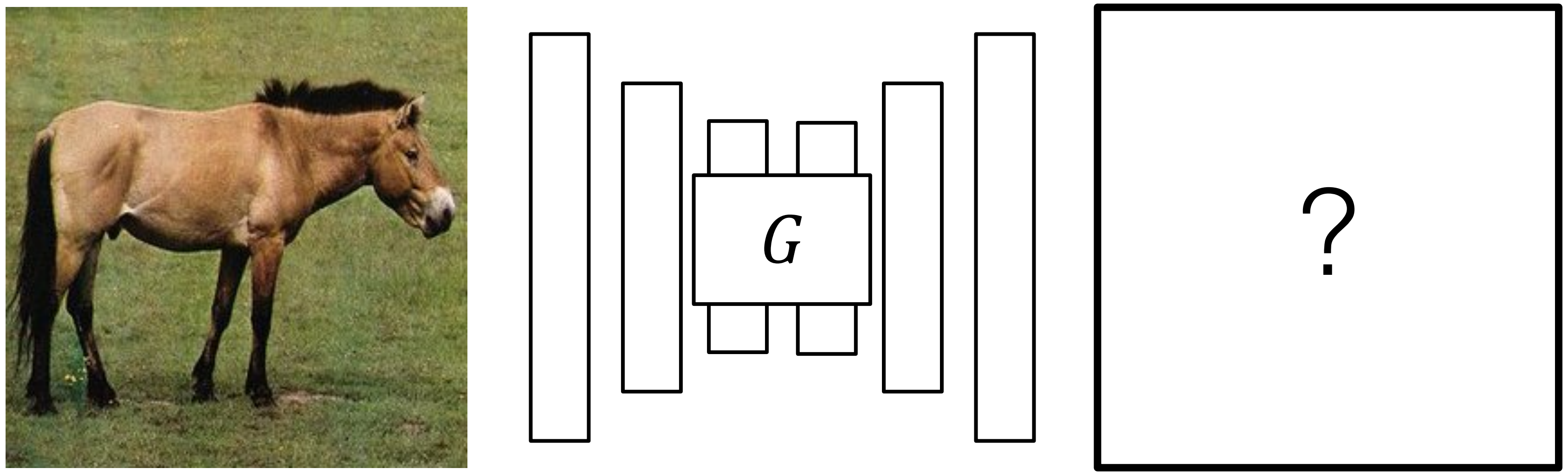
- <u>Style</u>: variations within the same domain
  (different colors, textures, etc.)
- <u>Content</u>: features shared across two domains



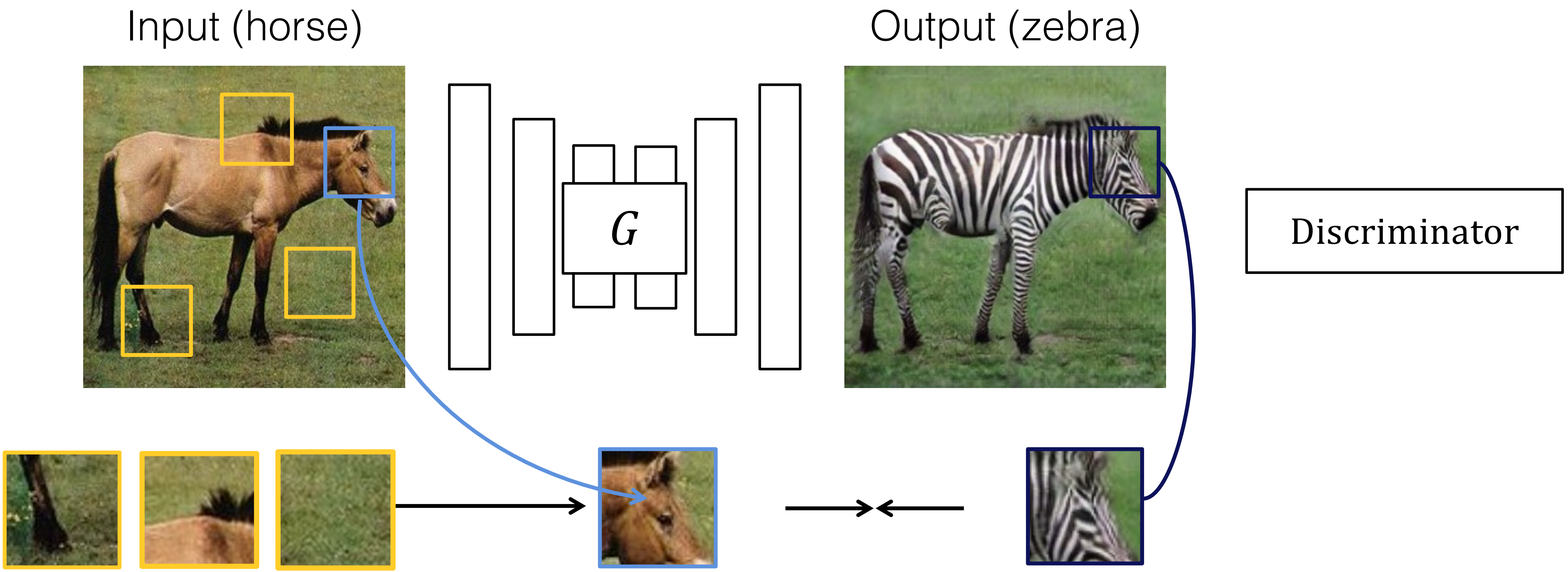



Style
Content

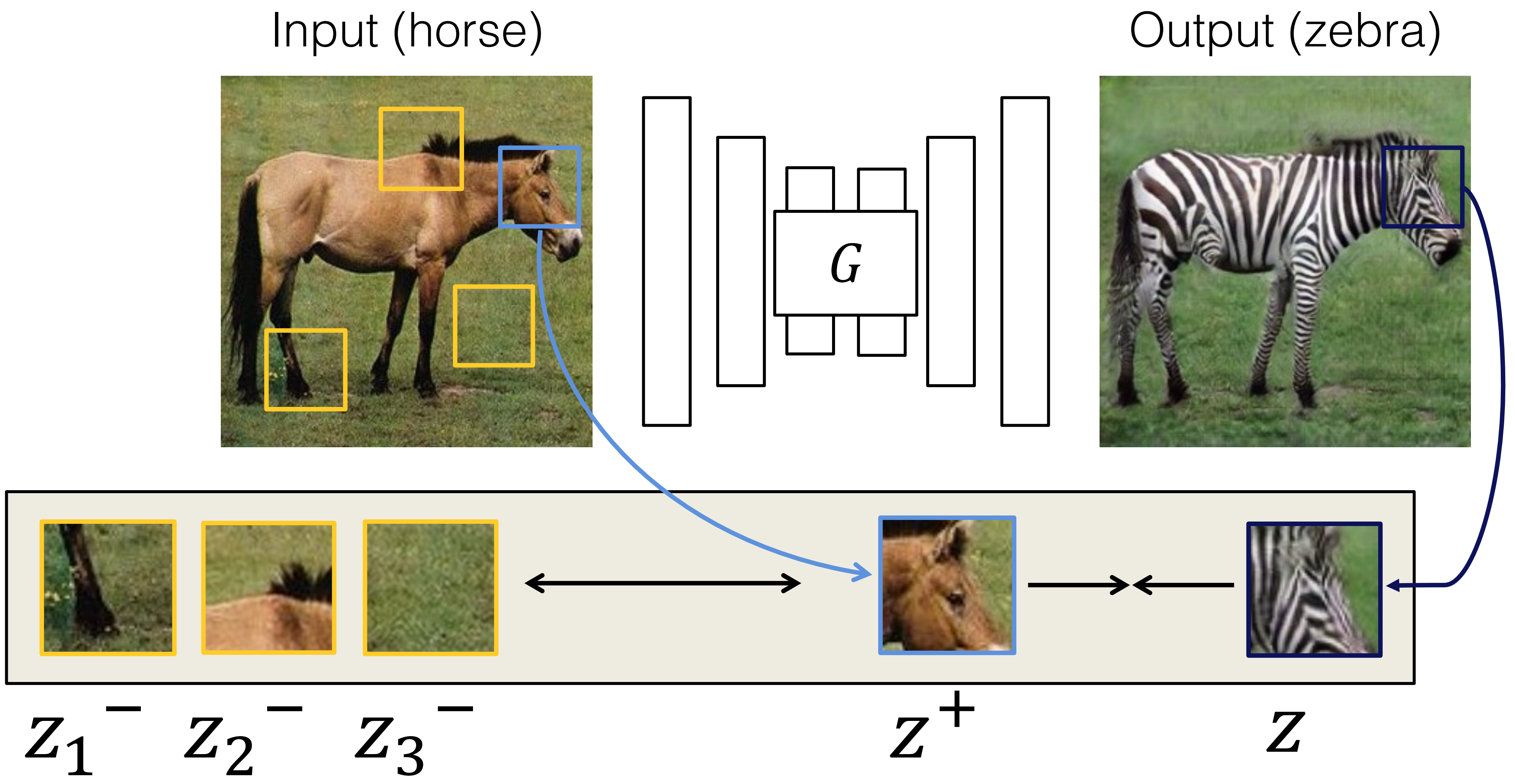
What makes for a good output?



Retaining input content

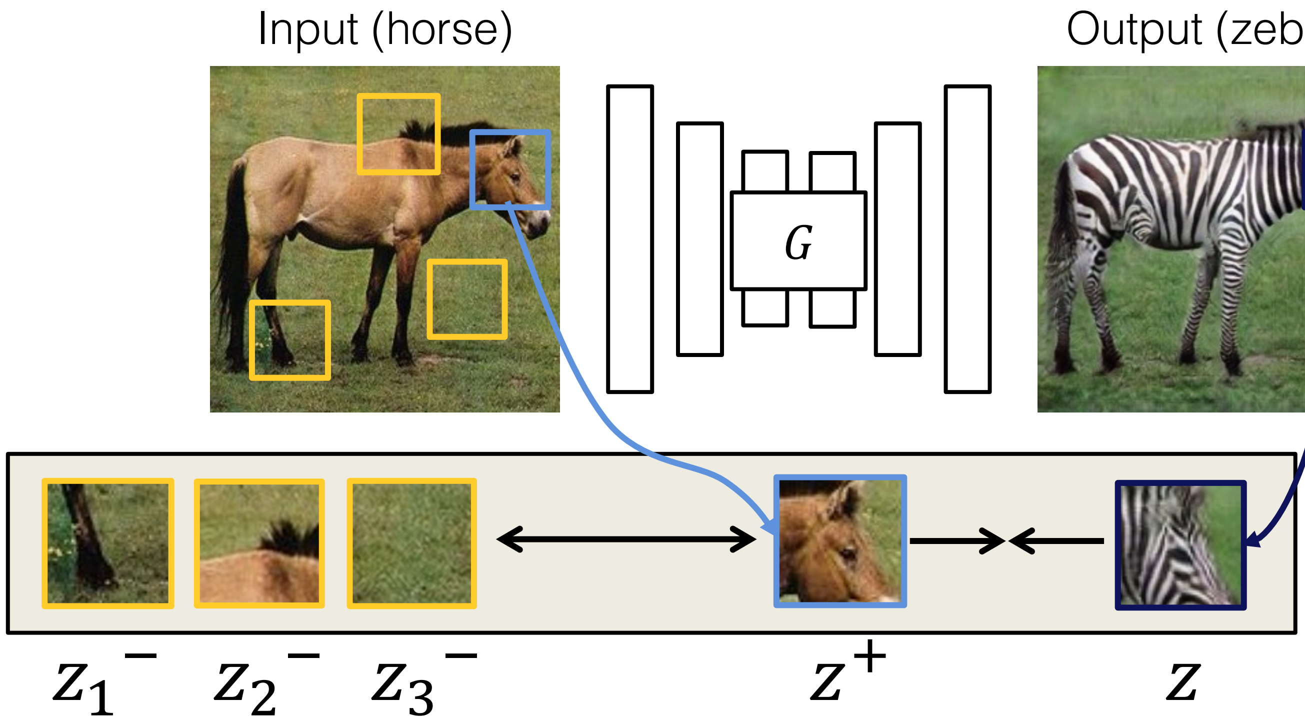


Retaining input **content**



Corresponding patches should have **high similarity**

Patch-based Contrastive Loss



$$\text{softmax}\begin{pmatrix} \uparrow z \cdot z^{+}/\tau \\ \downarrow z \cdot z_1^{-}/\tau \\ \downarrow z \cdot z_2^{-}/\tau \\ \vdots \\ \downarrow z \cdot z_N^{-}/\tau \end{pmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

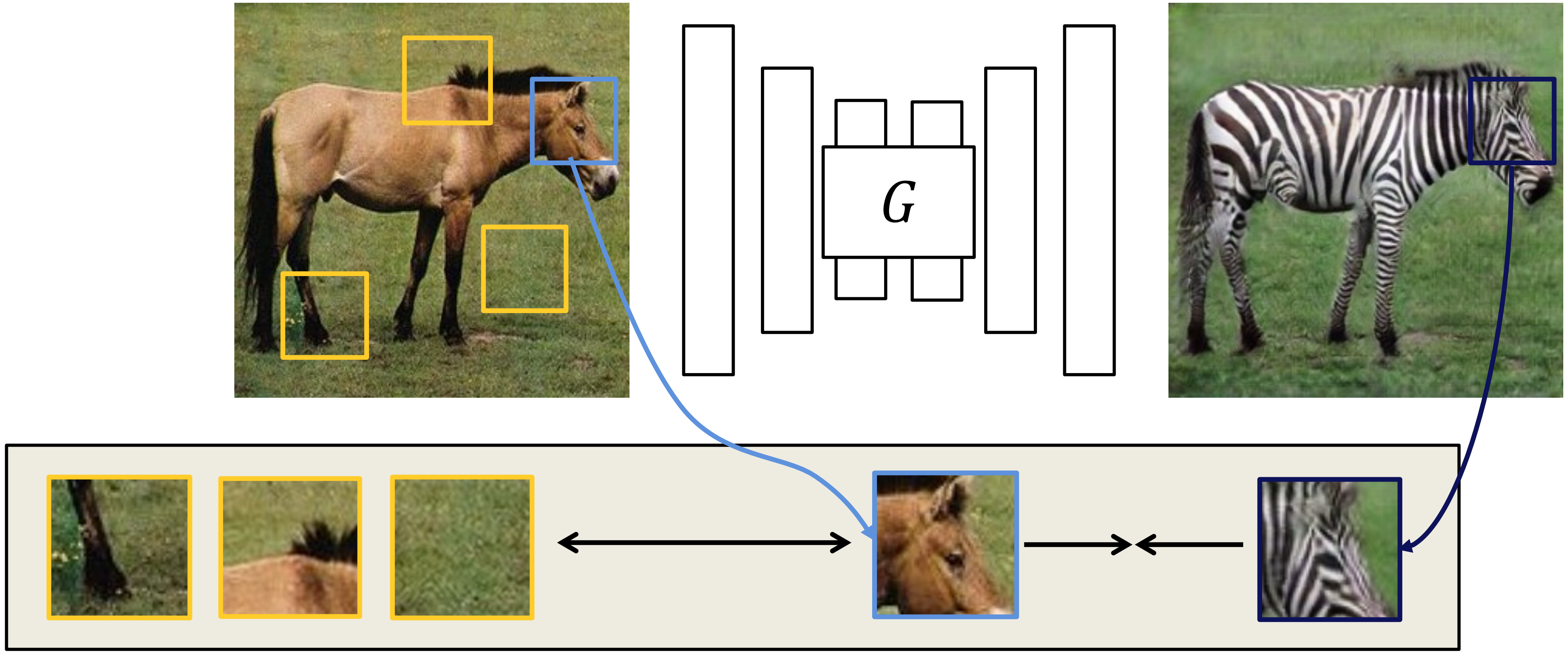
softmax (cosine similarities $/\tau$)

τ=0.07

- InfoNCE loss (Gutmann et al., AISTATS18 , van den Oord et al., 2018) used in MoCo and SimCLR
- To produce positive pairs:
  - Handcrafted data augmentation (MoCo, SimCLR, etc.)
  - Input and synthesized image (ours)

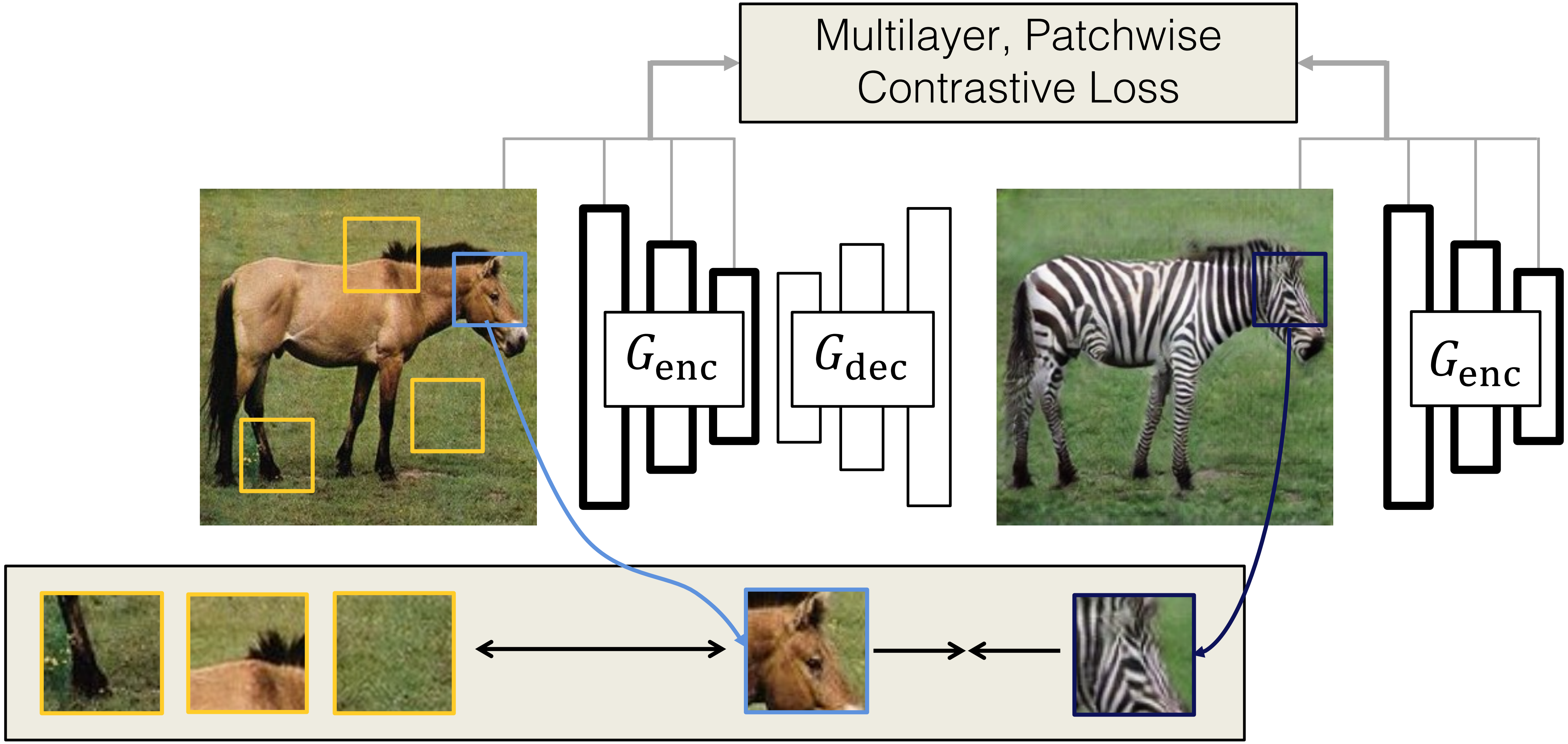
MoCo: He et al., CVPR20, SimCLR: Chen et al., ICML20

Patchwise contrastive loss



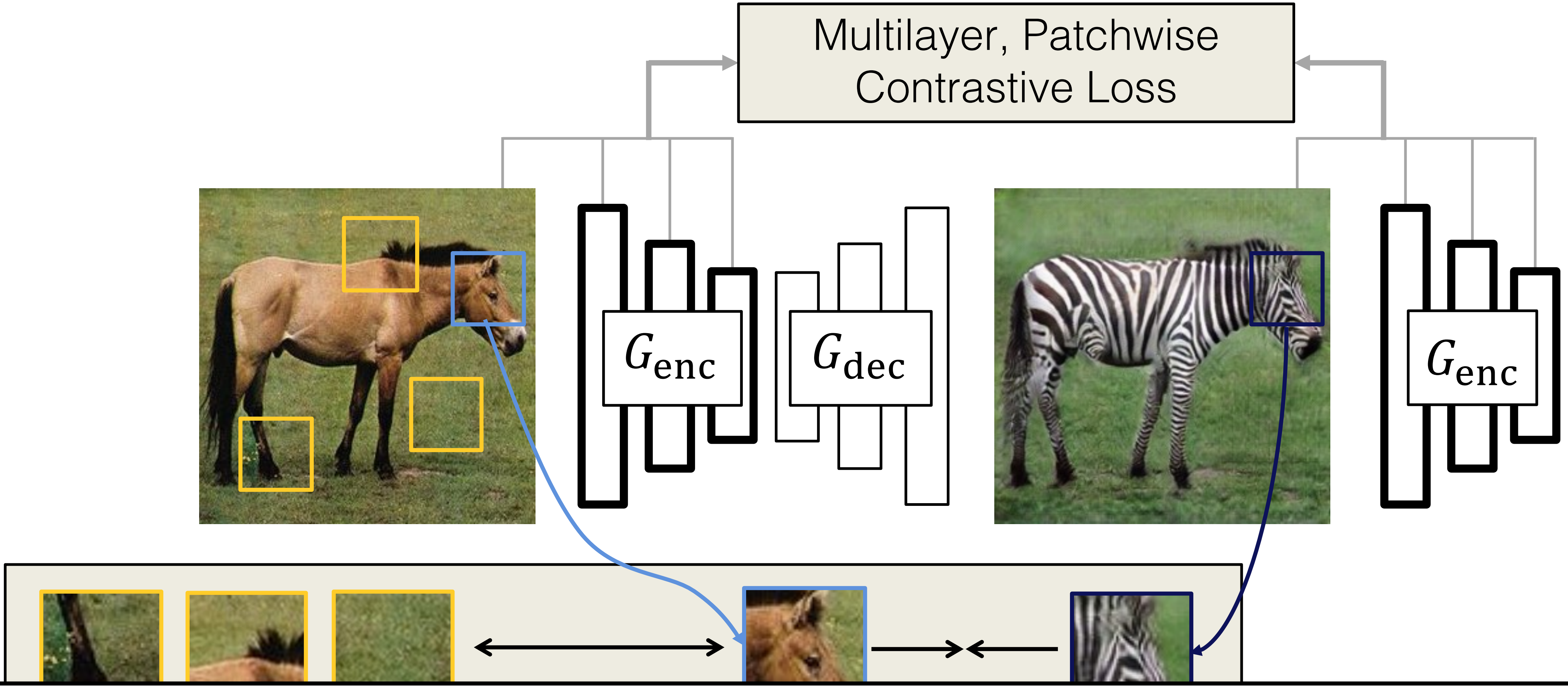
[Park et al., 2020]

Patchwise contrastive loss



[Park et al., 2020]

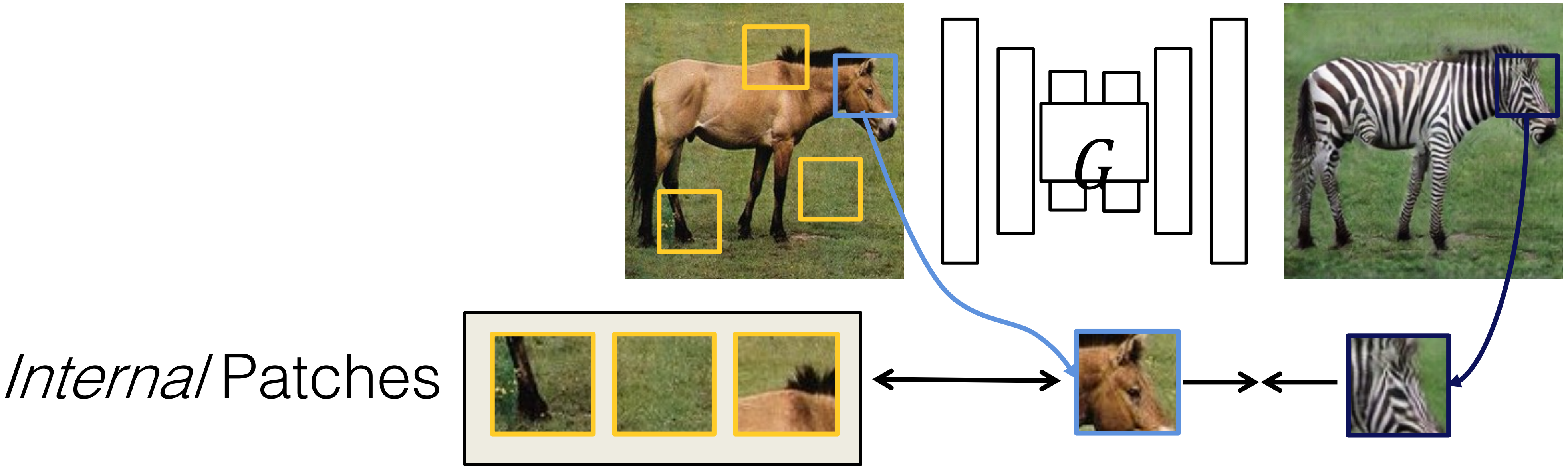
Patchwise contrastive loss



+ No fixed similarity metric (e.g., L1 or perceptual loss)
+ One-sided (no inverse mapping needed)

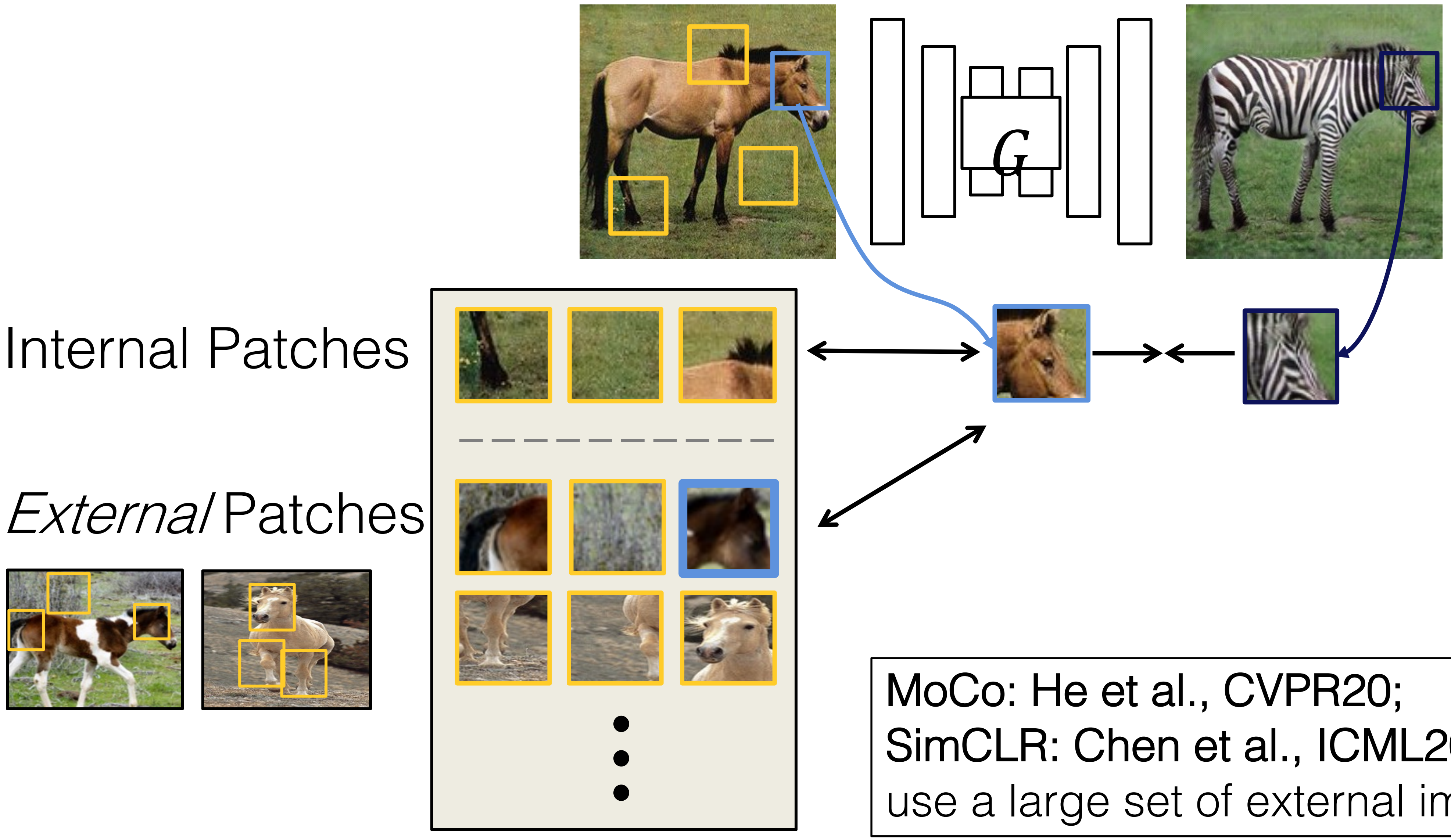
[Park et al., 2020]

Internal vs External Patches



[Park et al., 2020]

Internal vs External Patches

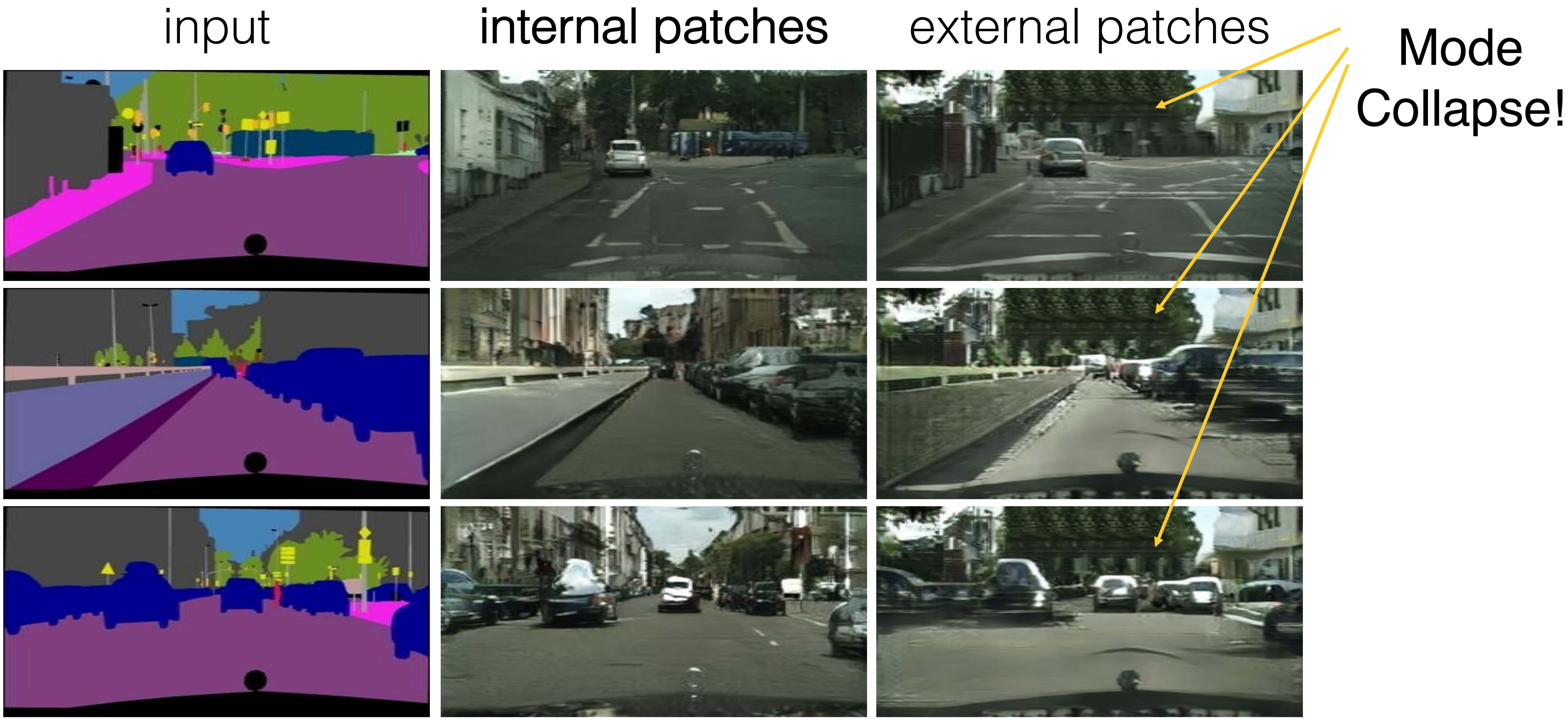


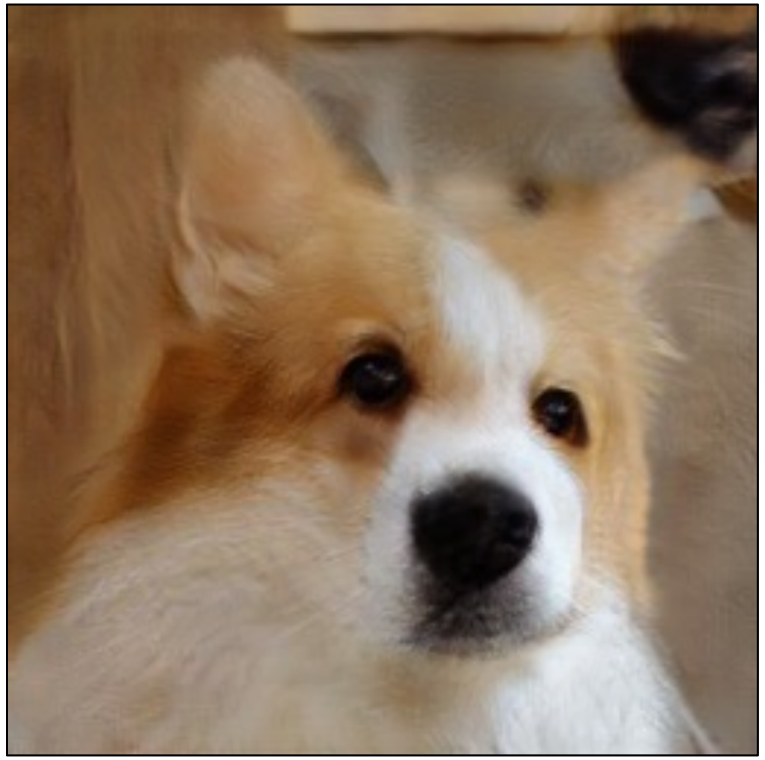
MoCo: He et al., CVPR20;
SimCLR: Chen et al., ICML20
use a large set of external images as negative samples

External patches make things worse

[Park et al., 2020]

Internal vs External Patches





Cat

Yosemite Summer

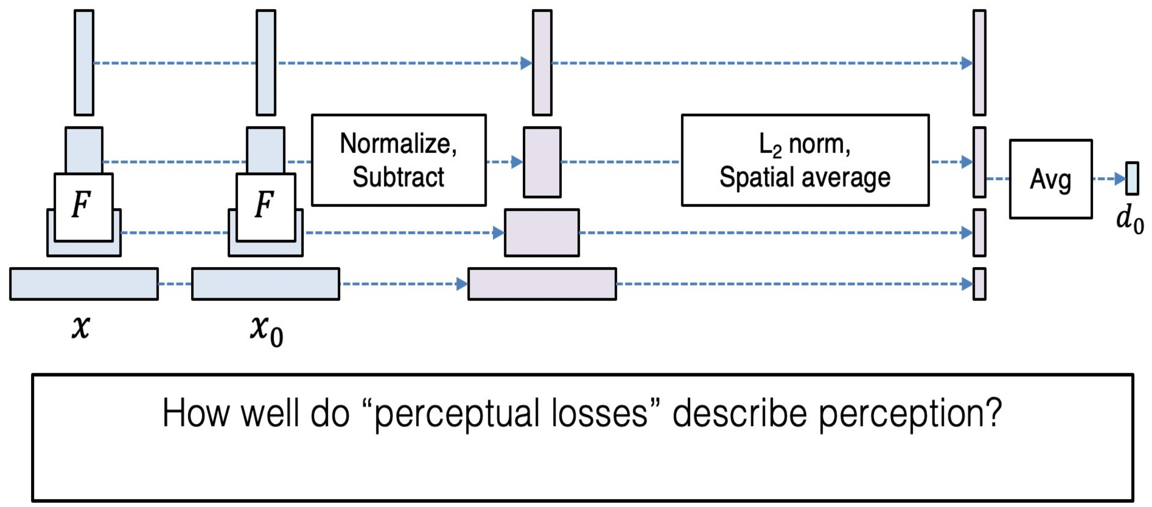
Apple

Paris

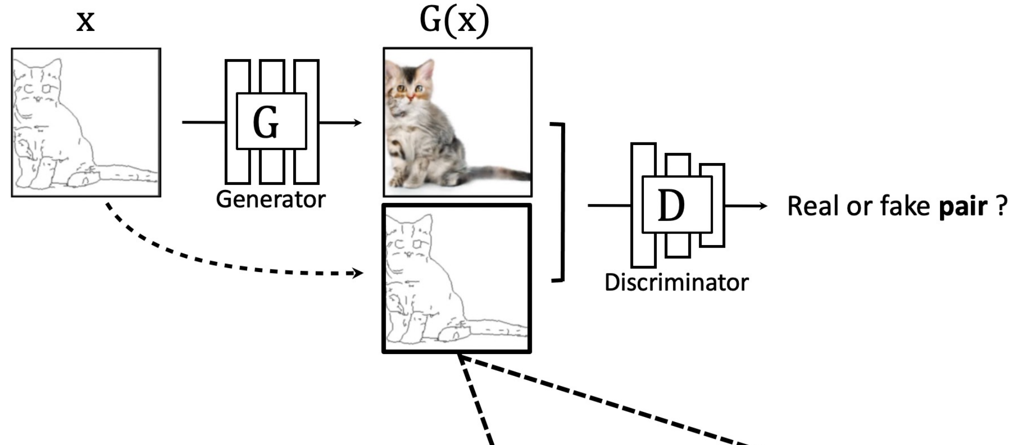
GTA

Summary

Perceptual/Feature Loss



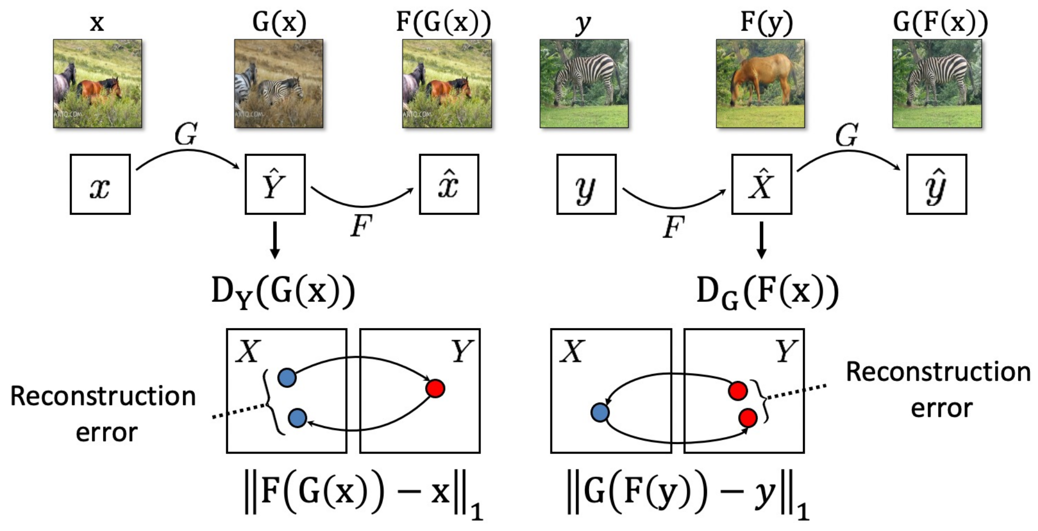
(Conditional) GAN Loss



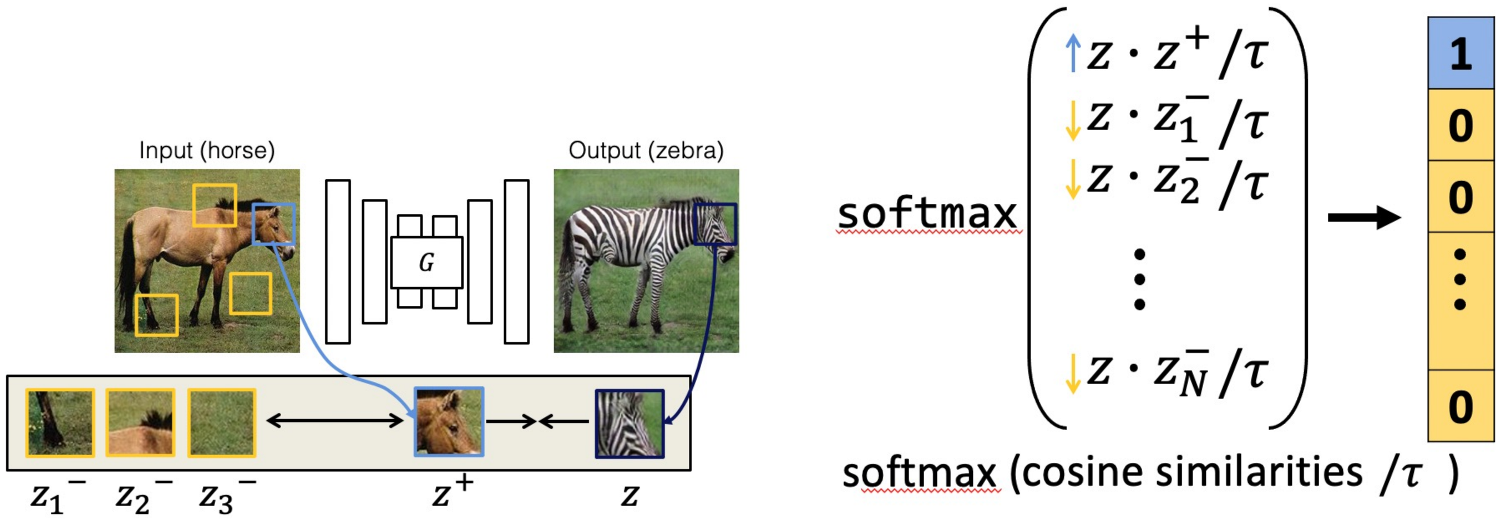
Learning objective

$$\min_G \max_D \mathbb{E}_x[\log(1-D(\boxed{x}, G(x))] + \mathbb{E}_{x,y}[\log D(\boxed{x}, y)]$$

Cycle-Consistency Loss



Patch-wise Contrastive Loss



Other loss functions: Style Loss [Gatys et al.], Contextual Loss [Mechrez et al.],
Domain-specific Loss (e.g., face), 3D-aware Loss (for geometric data)

Style and Content, Texture Synthesis

Jun-Yan Zhu

16-726, Spring 2023

Many slides are borrowed from Alyosha Efros and Bill Freeman

photo © [Gatys et al., 2016]

Texture

- Texture depicts spatially repeating patterns
- Many natural phenomena are textures

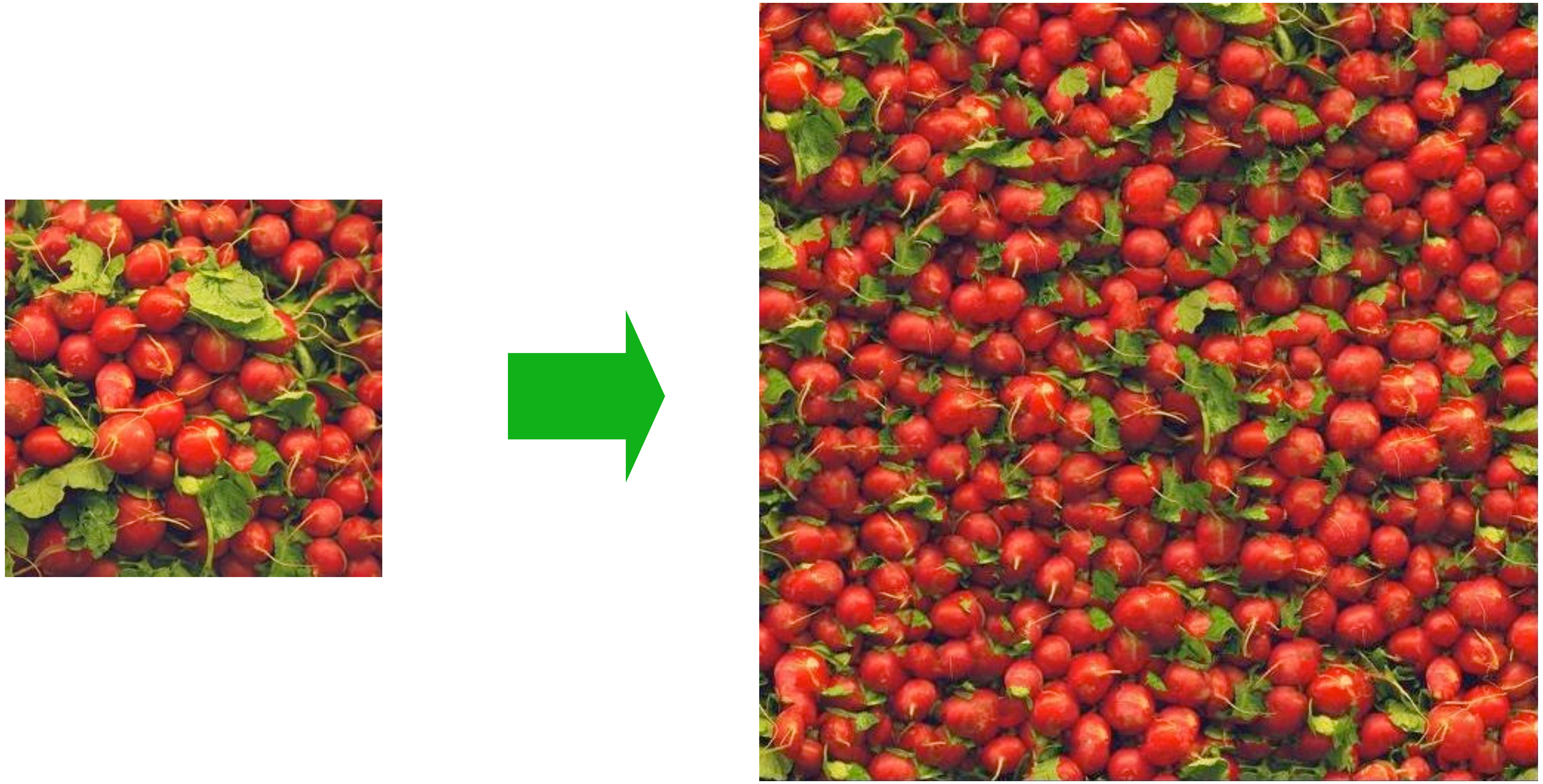
radishes

rocks

yogurt

Texture Synthesis

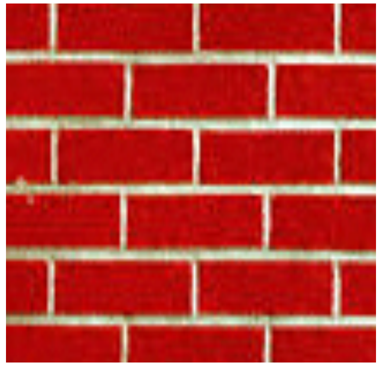
- Goal: create new samples of a given texture
- Applications: virtual environments, inpainting, texturing surfaces



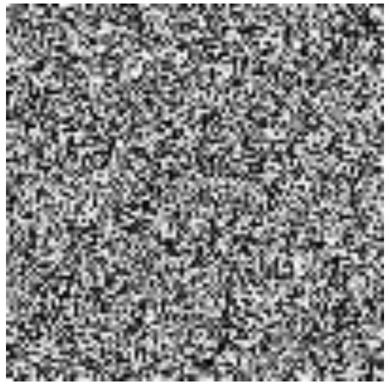
Non-parametric Texture Synthesis

The Challenge

- Need to model the whole spectrum: from repeated to stochastic texture



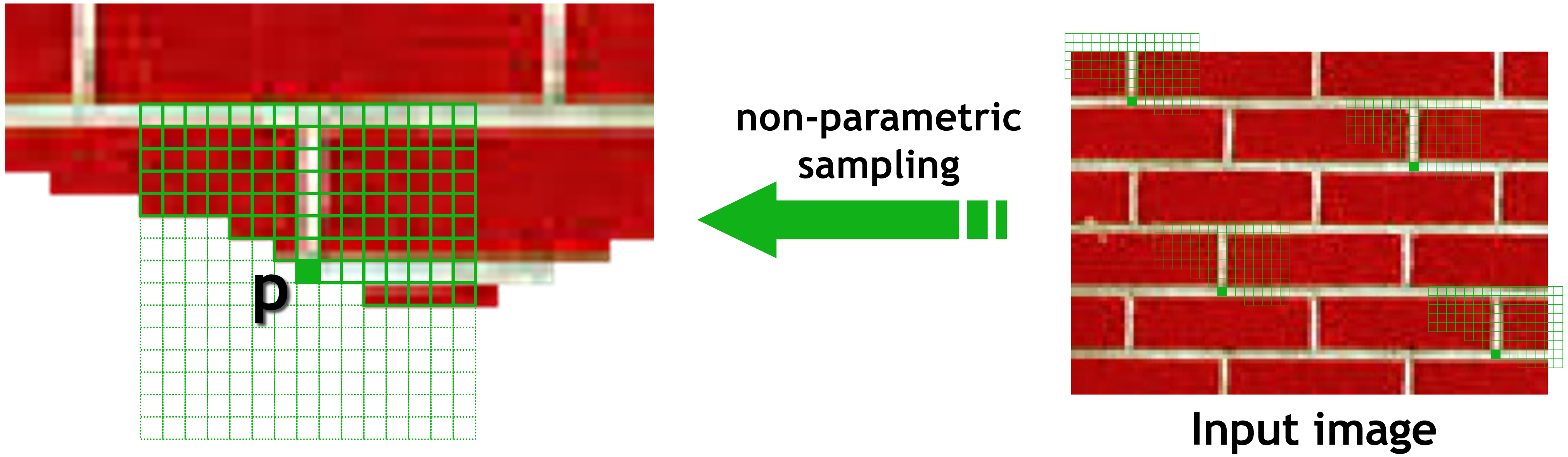
repeated



stochastic

Both?

Efros & Leung Algorithm



Synthesizing a pixel

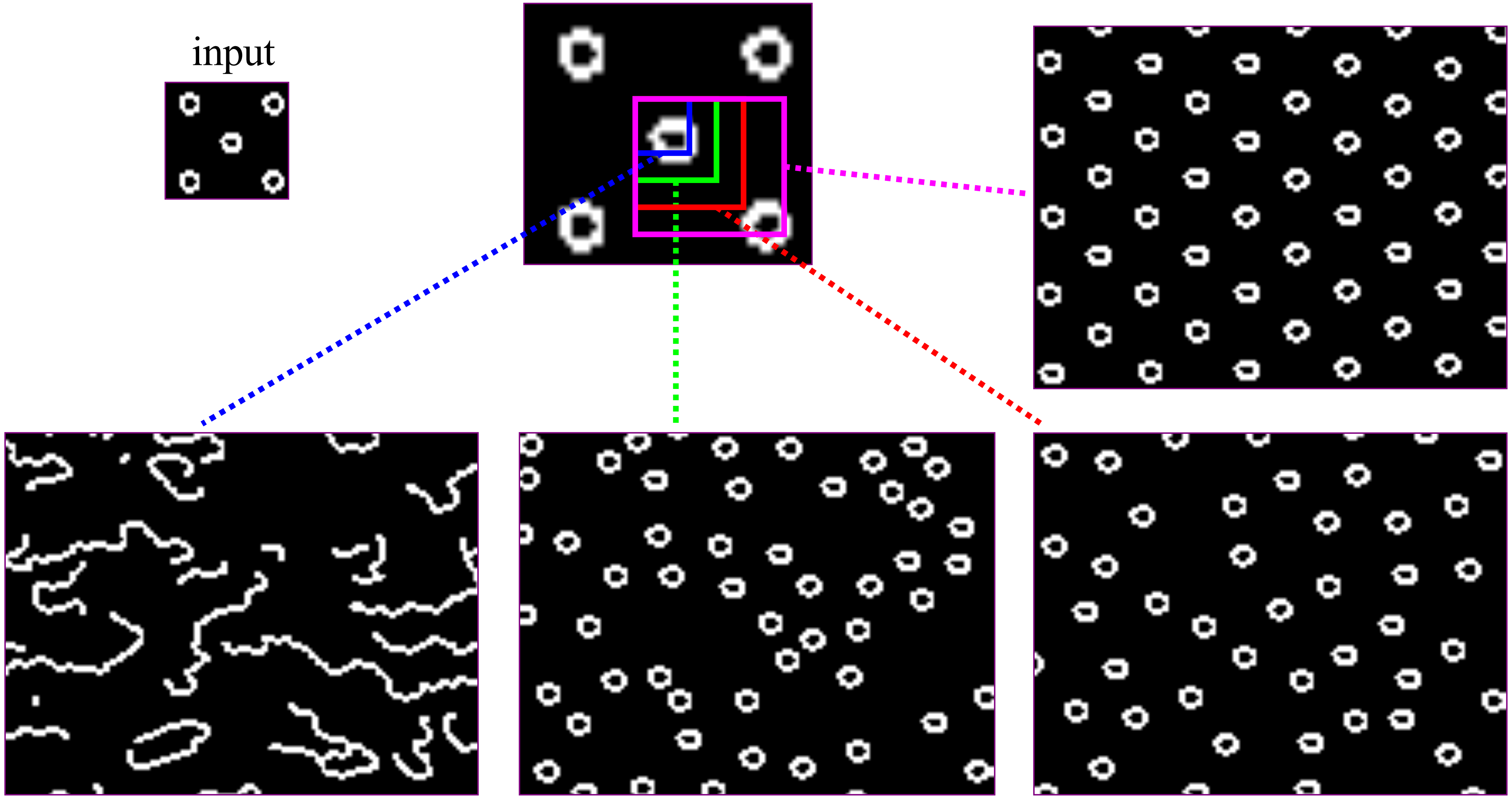
- Assuming Markov property, compute P(**p**|N(**p**))
  - Building explicit probability tables infeasible
  - Instead, we *search the input image* for all similar neighbourhoods — that's our pdf for **p**
  - To sample from this pdf, just pick one match at random

PixelCNN

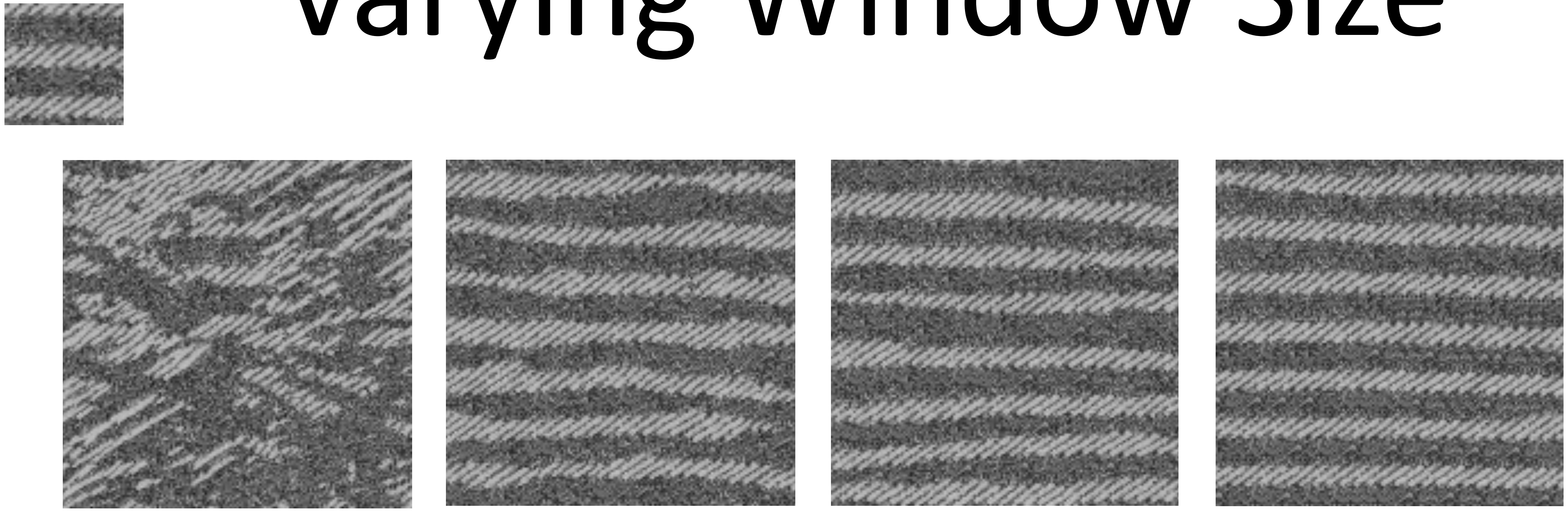
Some Details

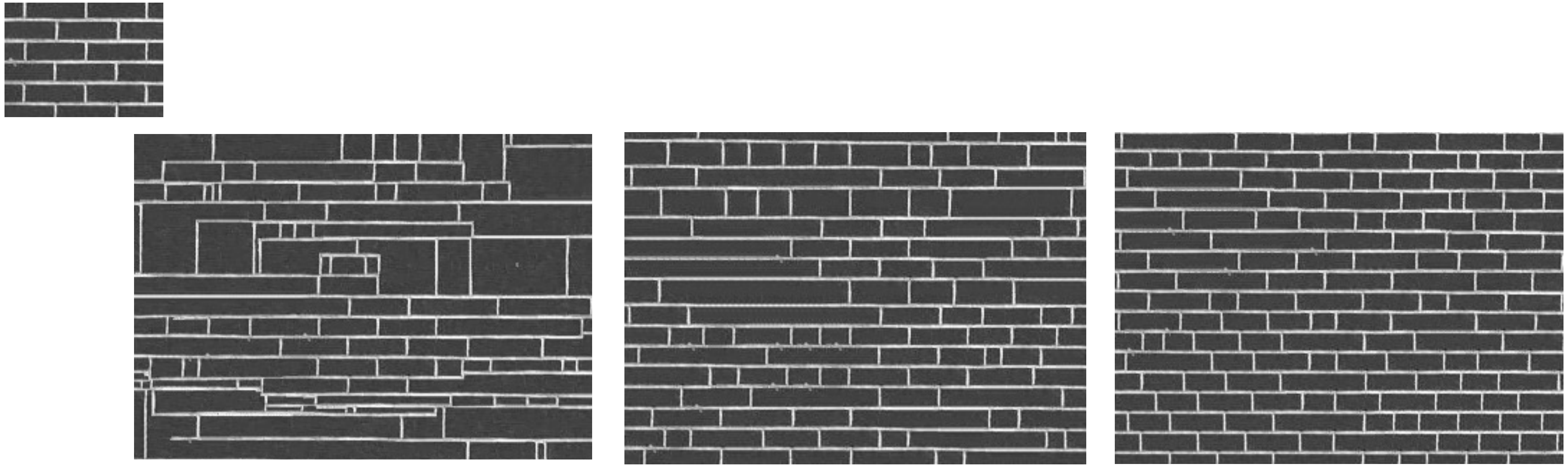
- Growing is in “onion skin” order
  - Within each “layer”, pixels with <u>most neighbors</u> are synthesized first
  - If no close match can be found, the pixel is not synthesized until the end
- Using *Gaussian-weighted* SSD is very important
  - to make sure the new pixel agrees with its <u>closest</u> neighbors
  - Approximates reduction to a smaller neighborhood window if data is too sparse

Neighborhood Window



Varying Window Size

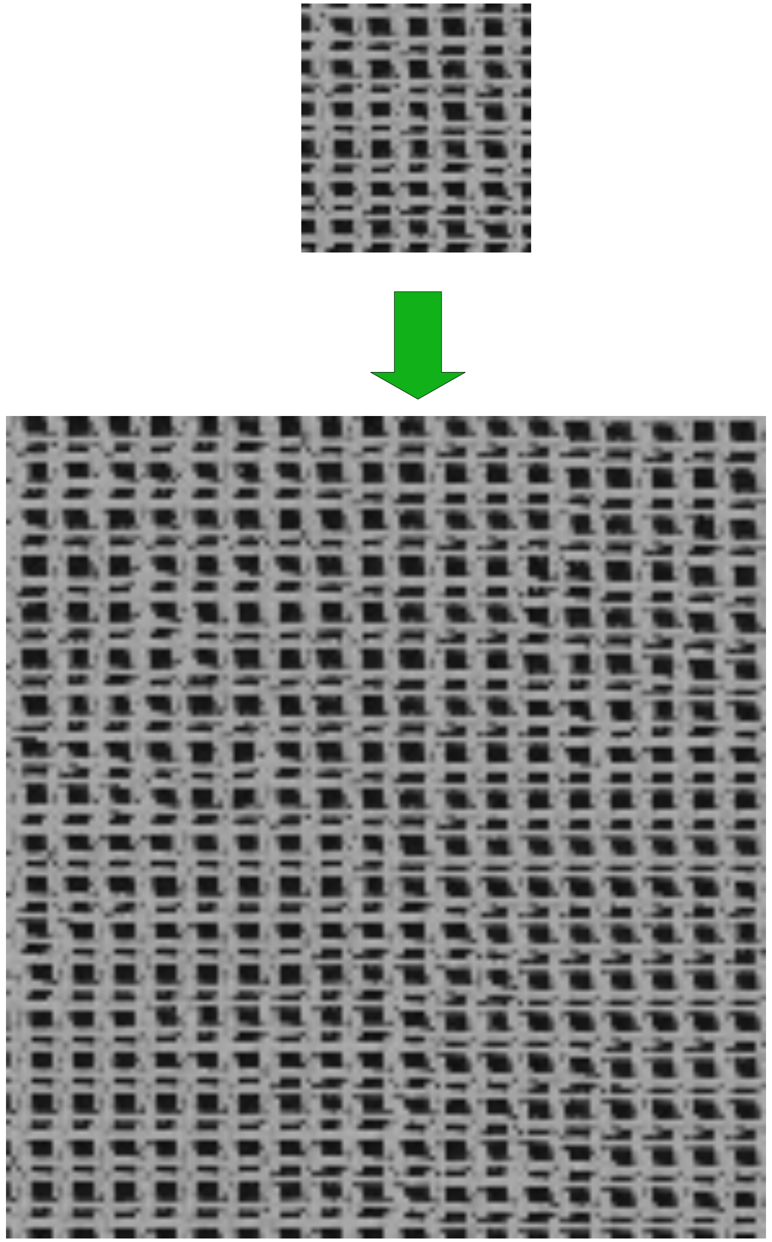




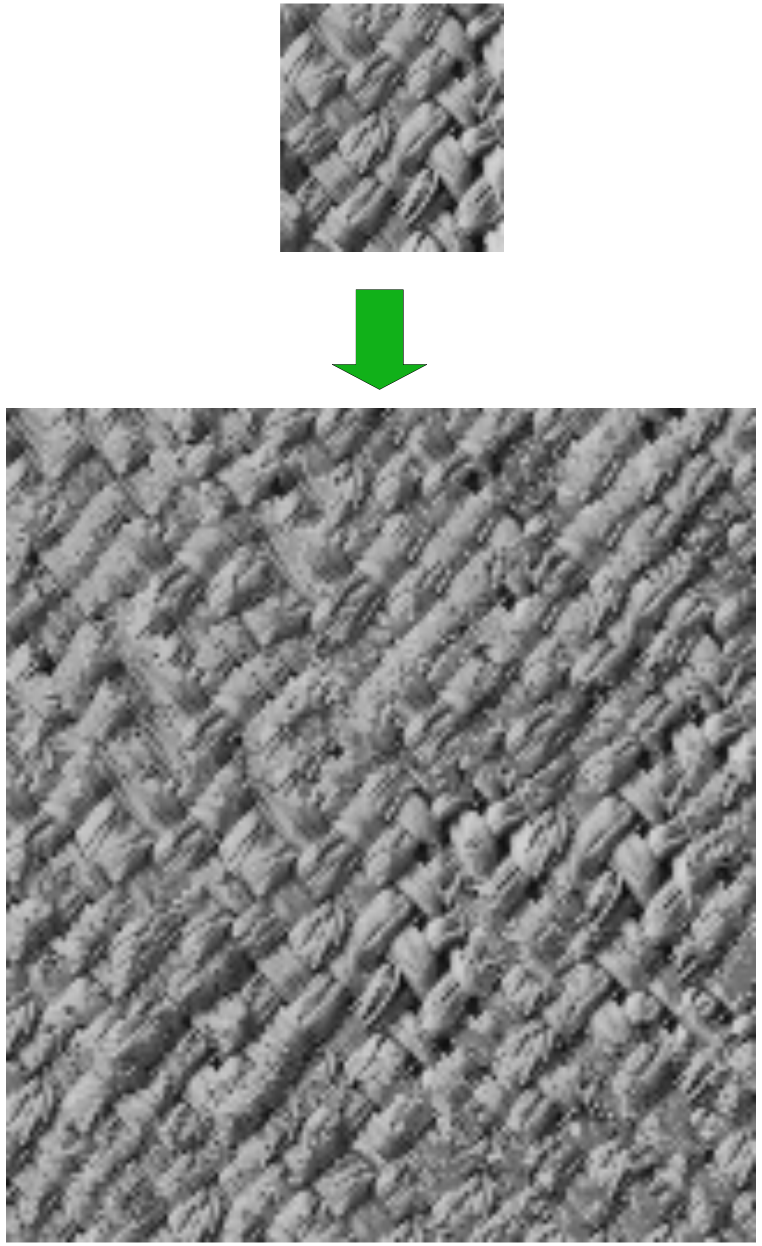
Increasing window size

Synthesis Results

french canvas



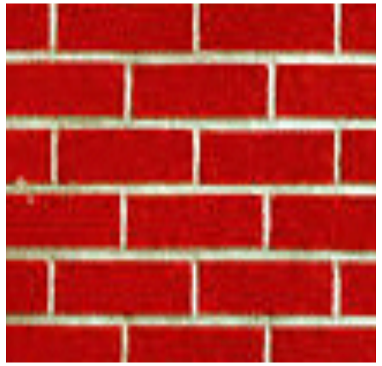
rafia weave



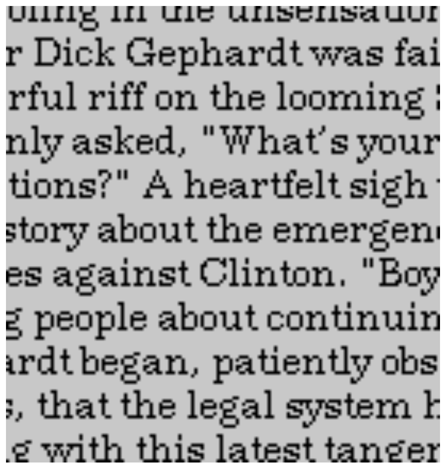
More Results

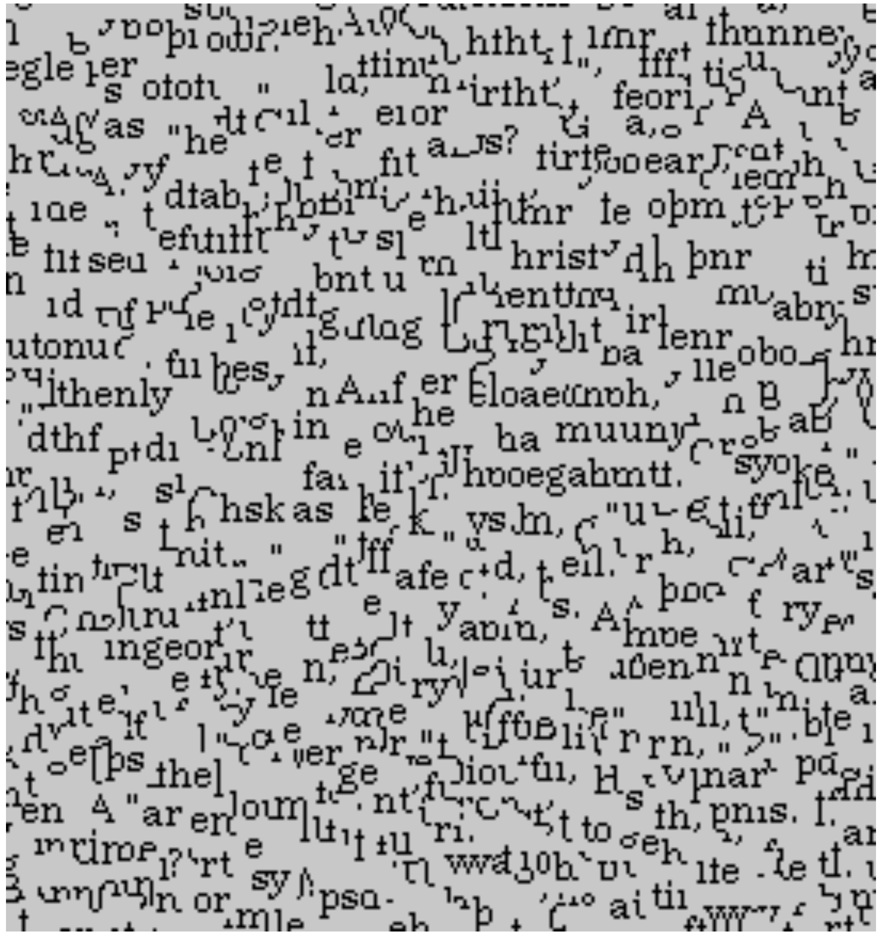
white bread

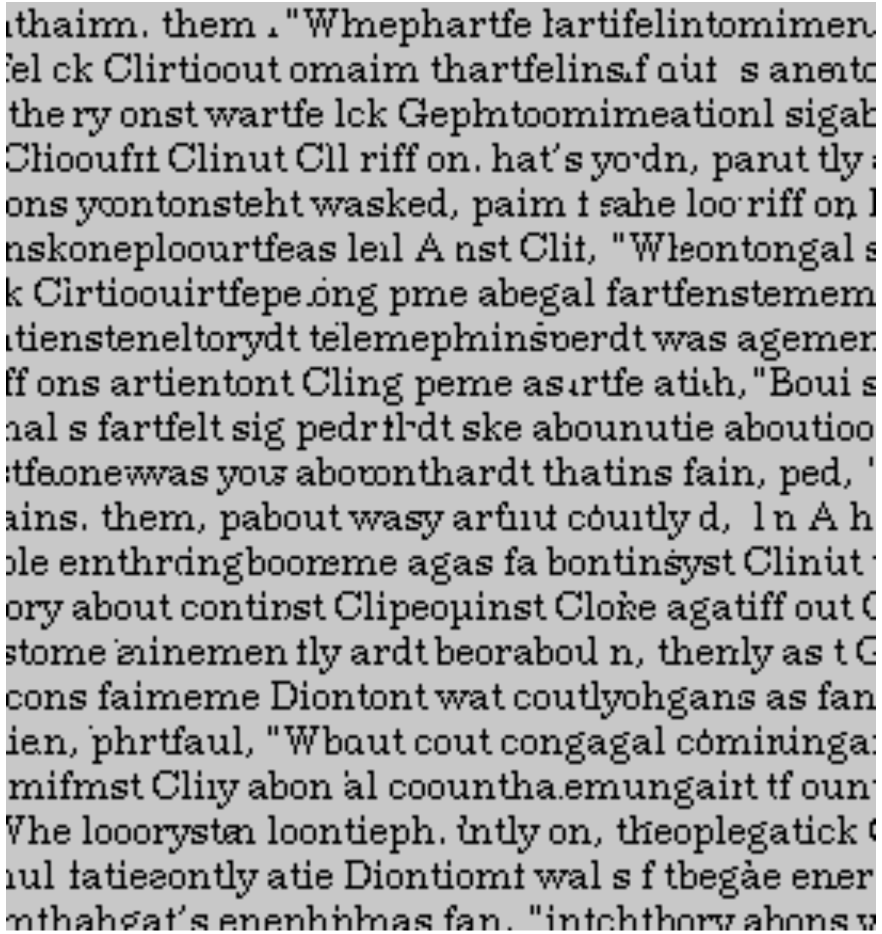
brick wall



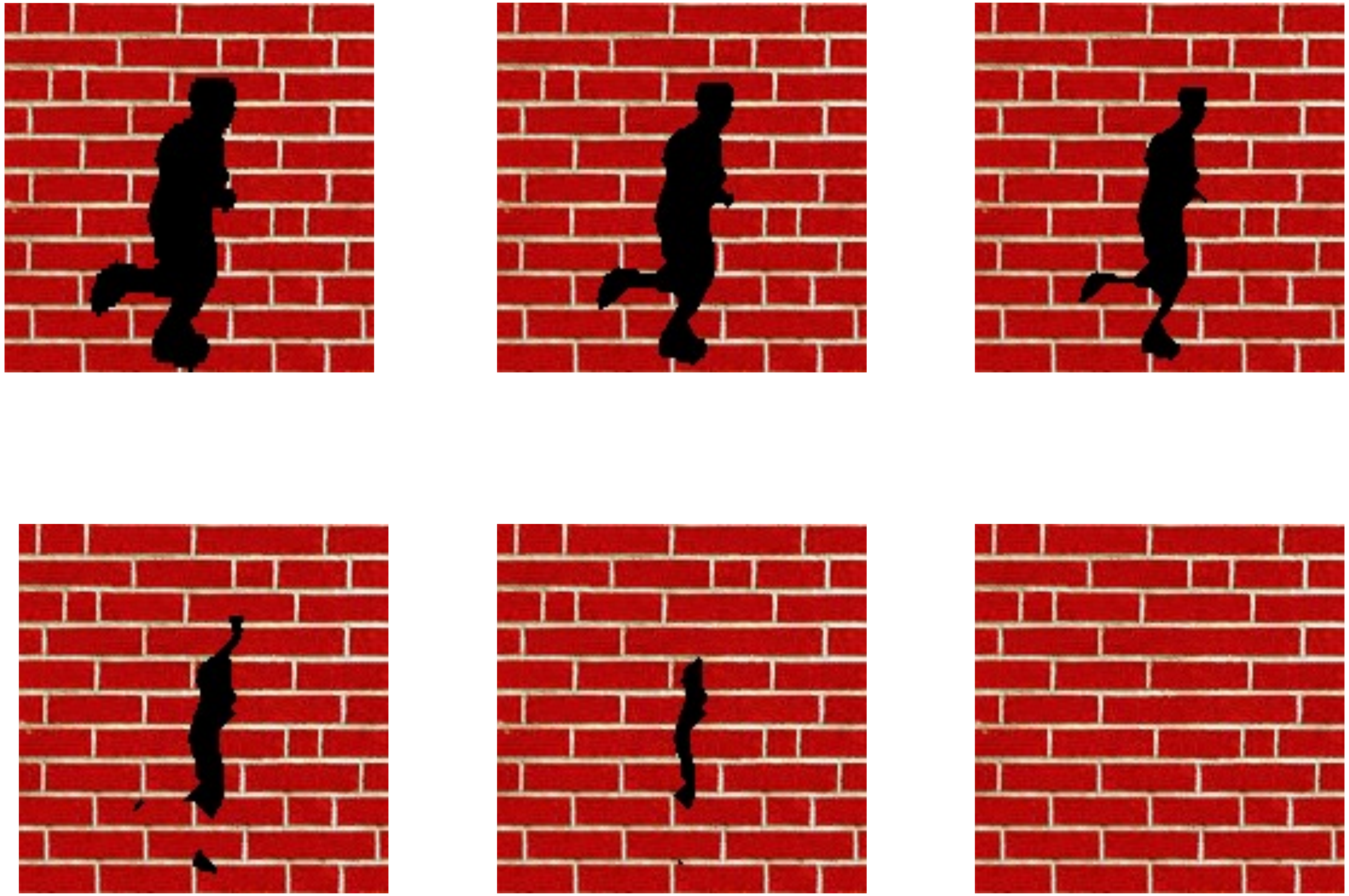
Homage to Shannon



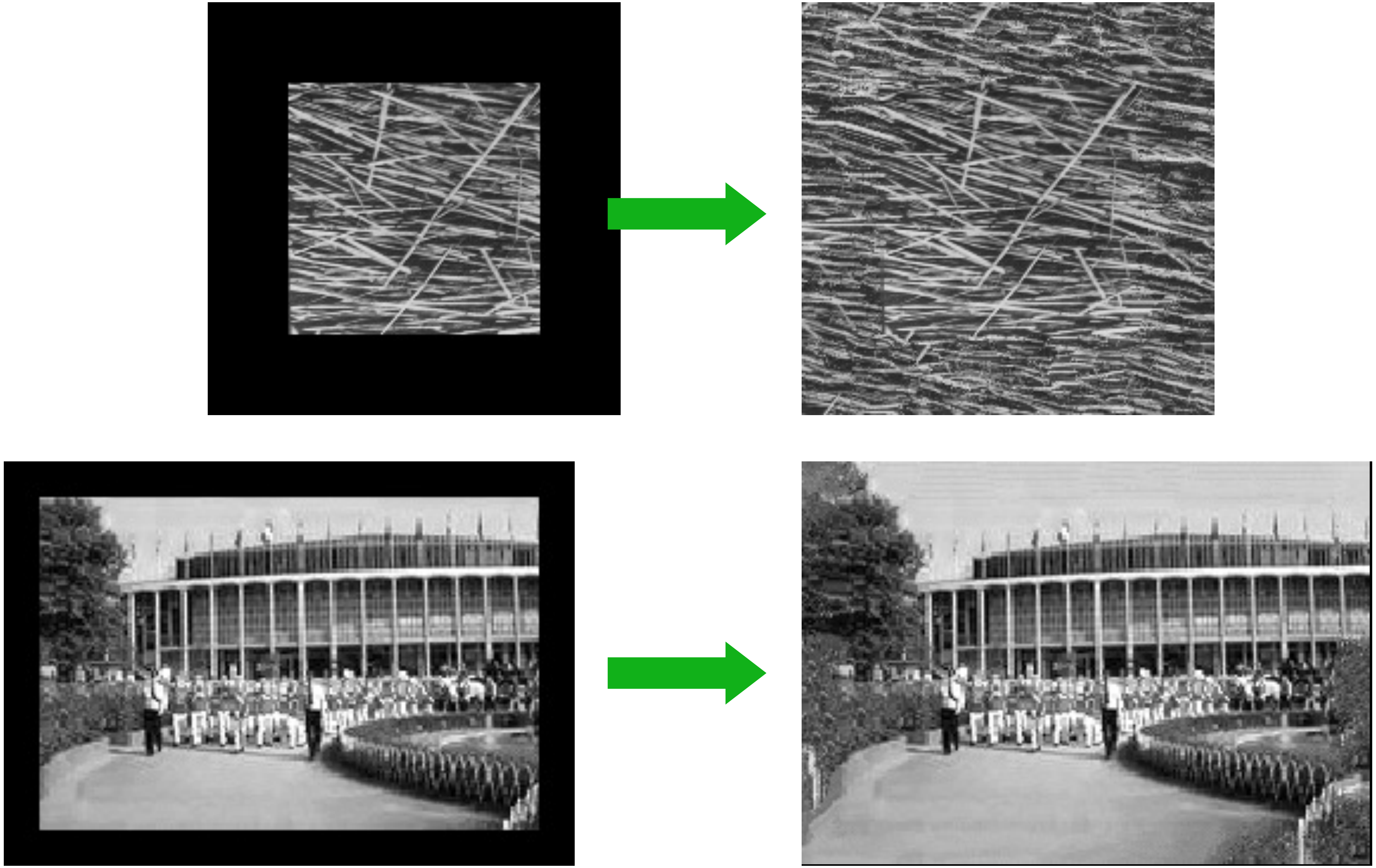




Hole Filling



Extrapolation



Summary

- The Efros & Leung algorithm

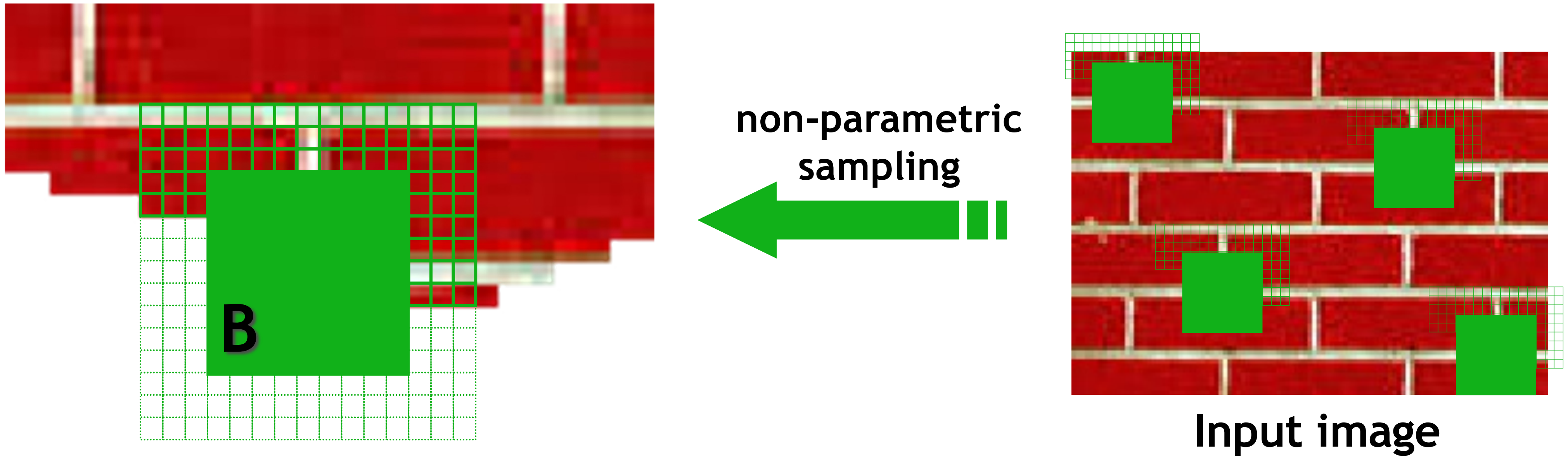
  \+ Very simple

  \+ Surprisingly good results

  \+ Synthesis is easier than analysis!

  -…but very slow

Image Quilting [Efros & Freeman]



Synthesizing a block

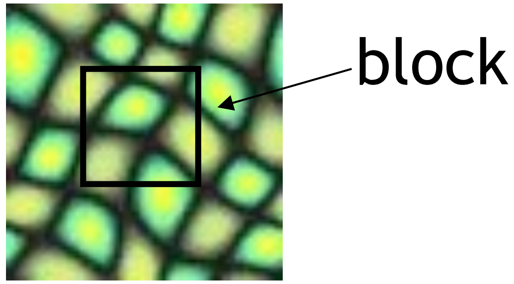
Input image

- Observation: neighbor pixels are highly correlated

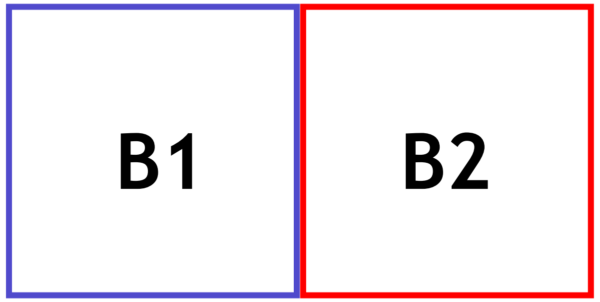
Idea: unit of synthesis = block

- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!

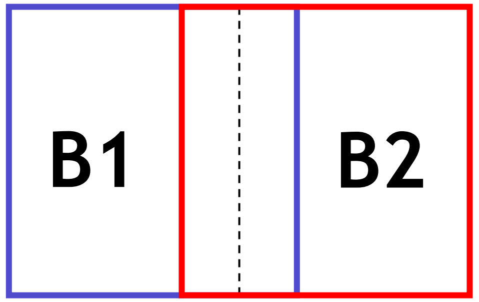
VQ-VAE2



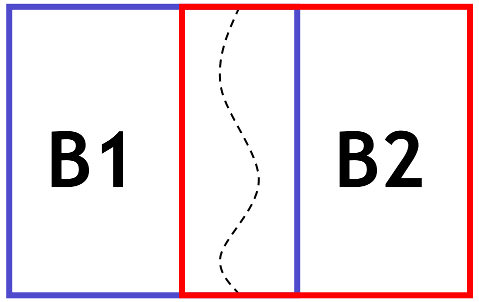
Input texture



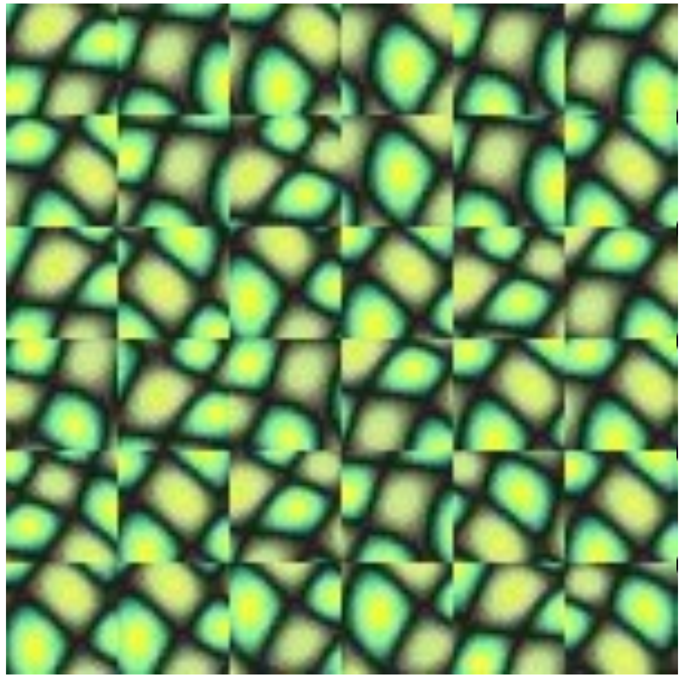
Random placement of blocks

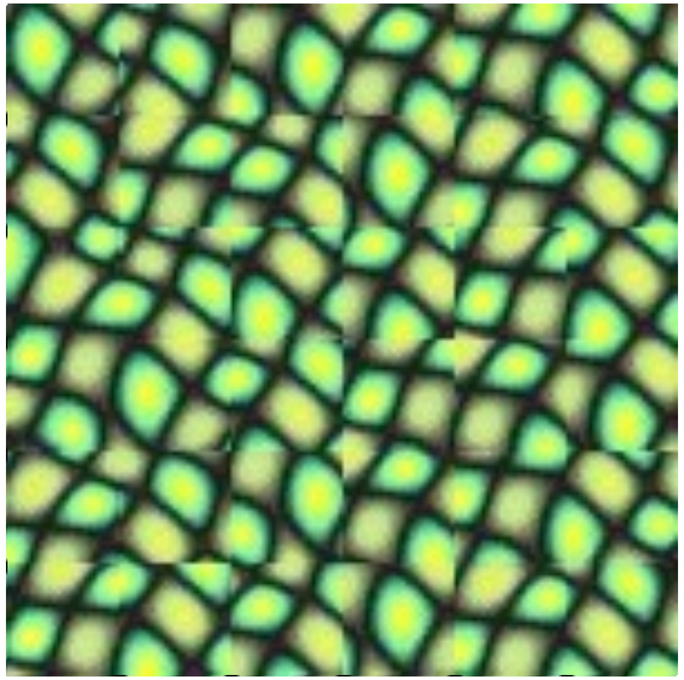


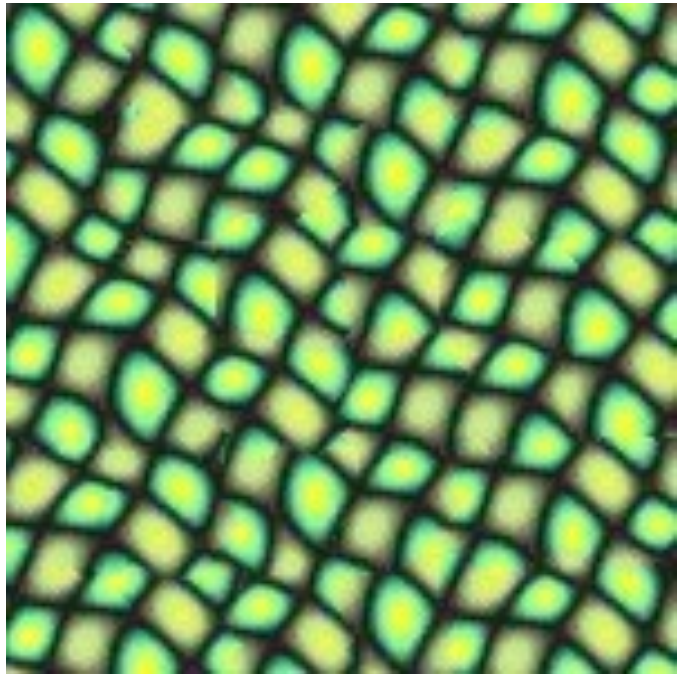
Neighboring blocks constrained by overlap



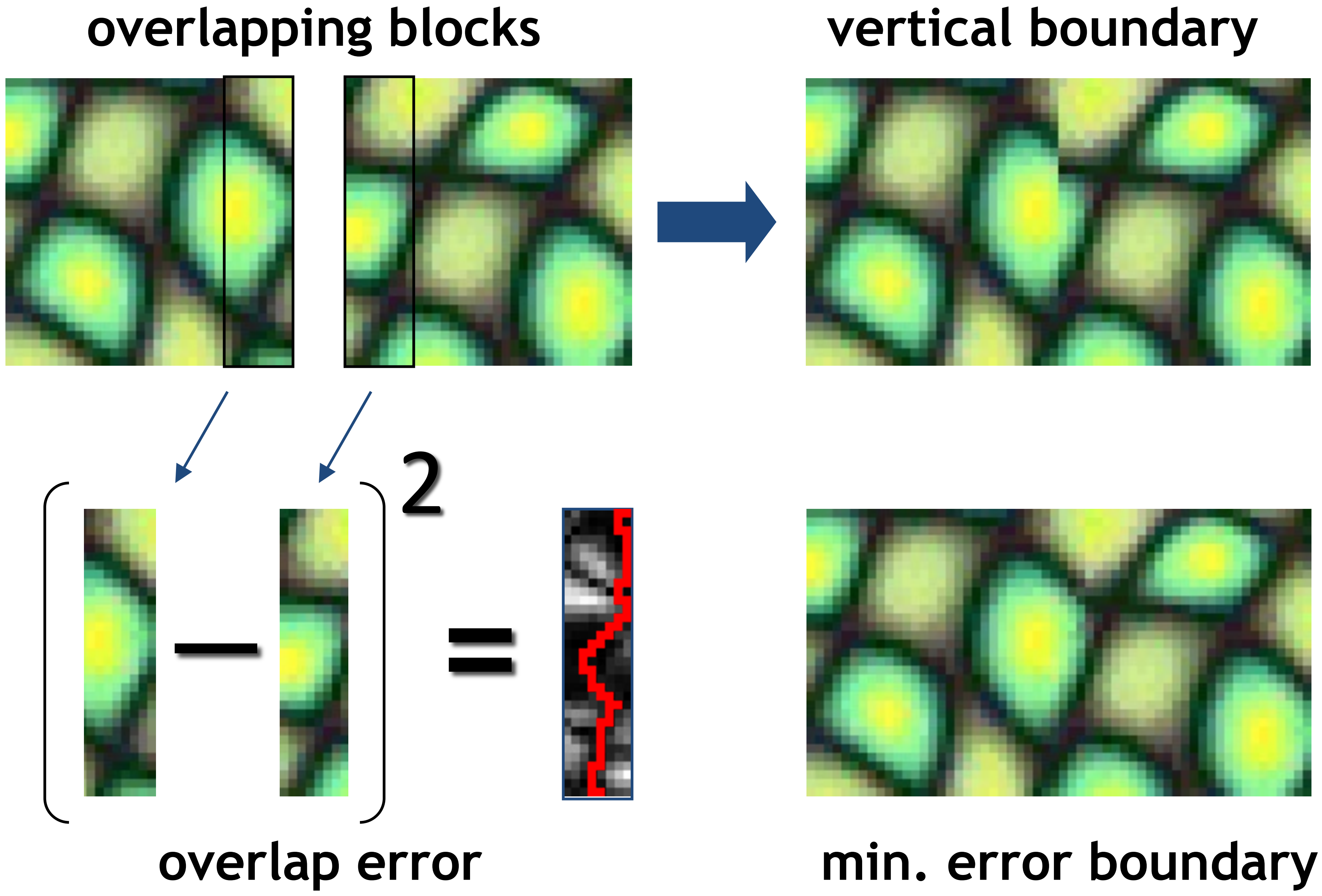
Minimal error boundary cut







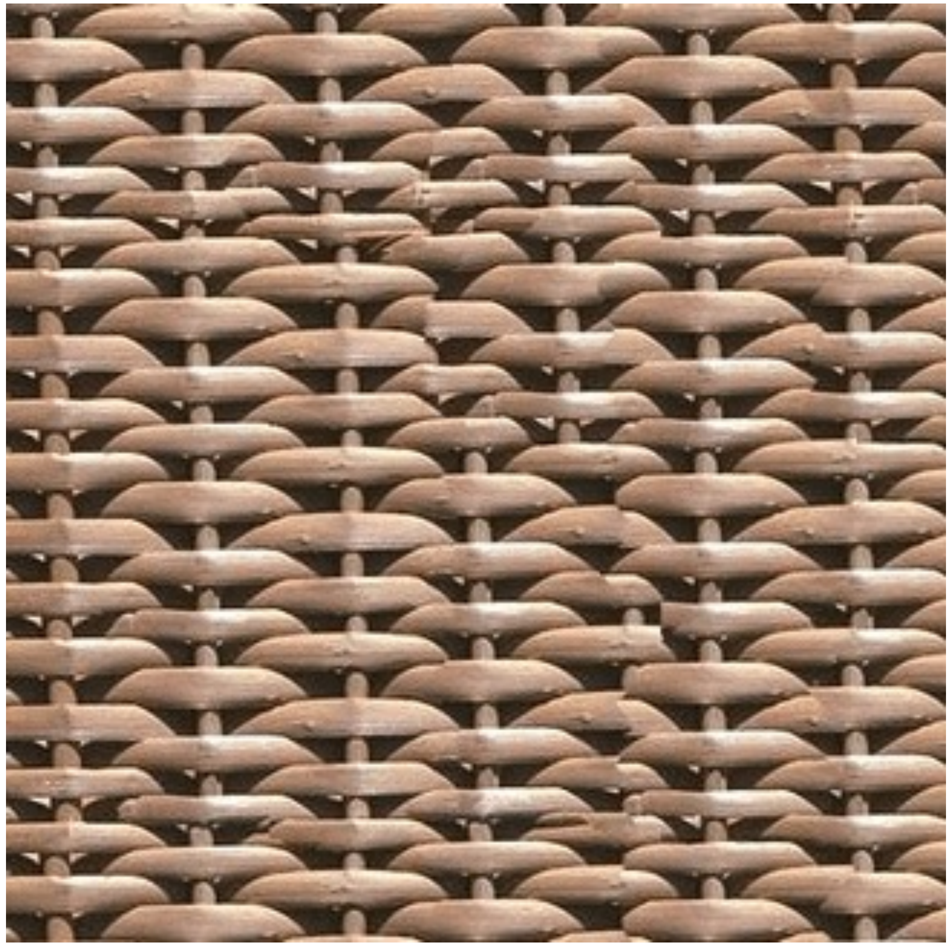
Minimal error boundary



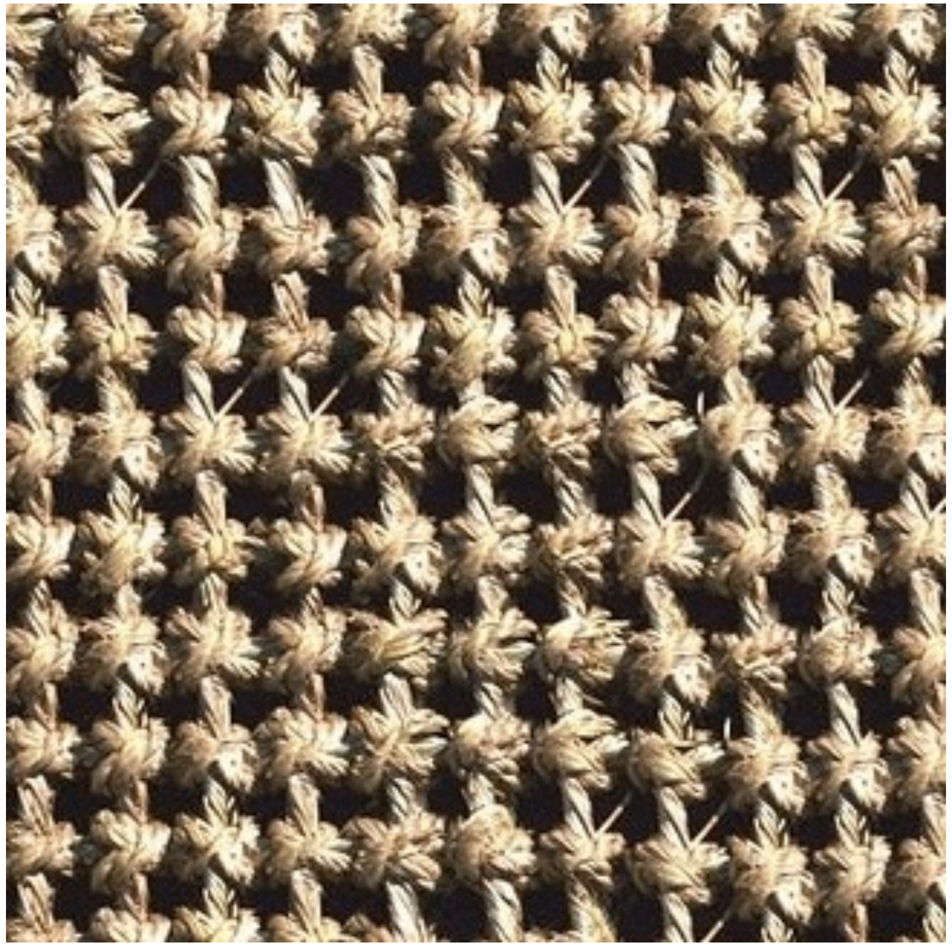
Our Philosophy

- The "Corrupt Professor's Algorithm":
  - Plagiarize as much of the source image as you can
  - Then try to cover up the evidence
- Rationale:
  - Texture blocks are by definition correct samples of texture so problem only connecting them together



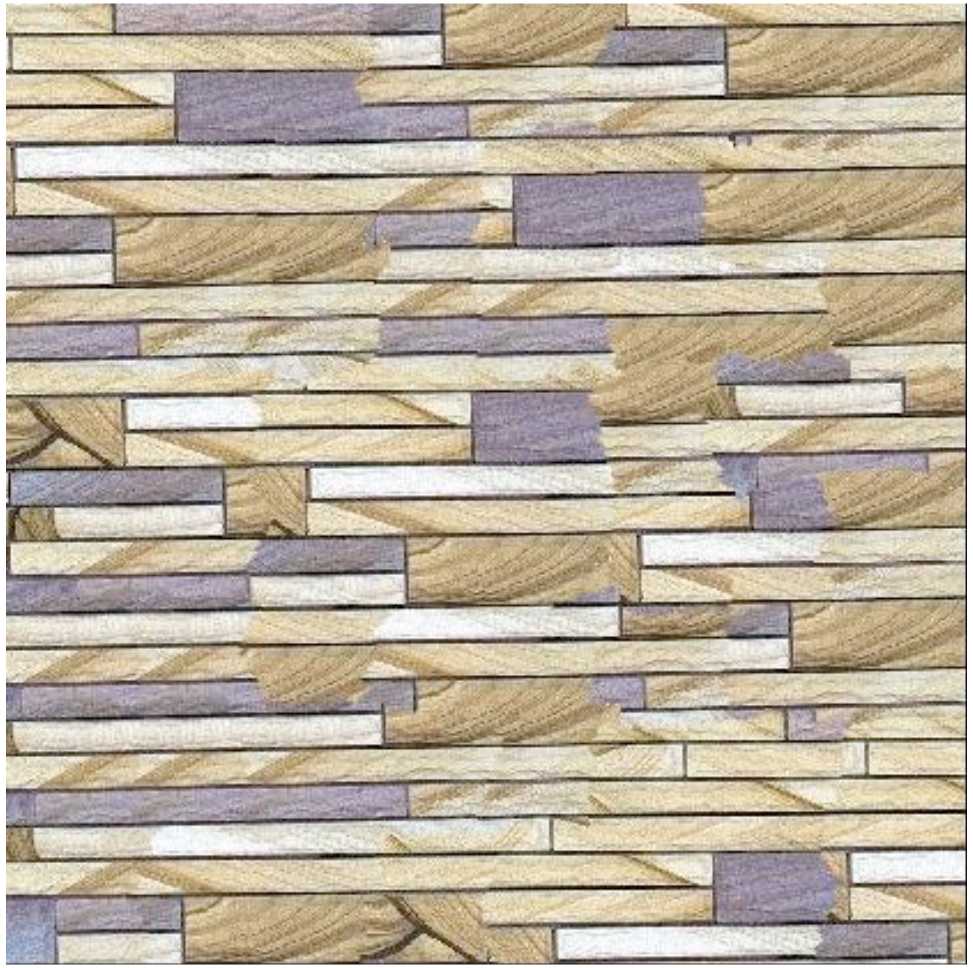













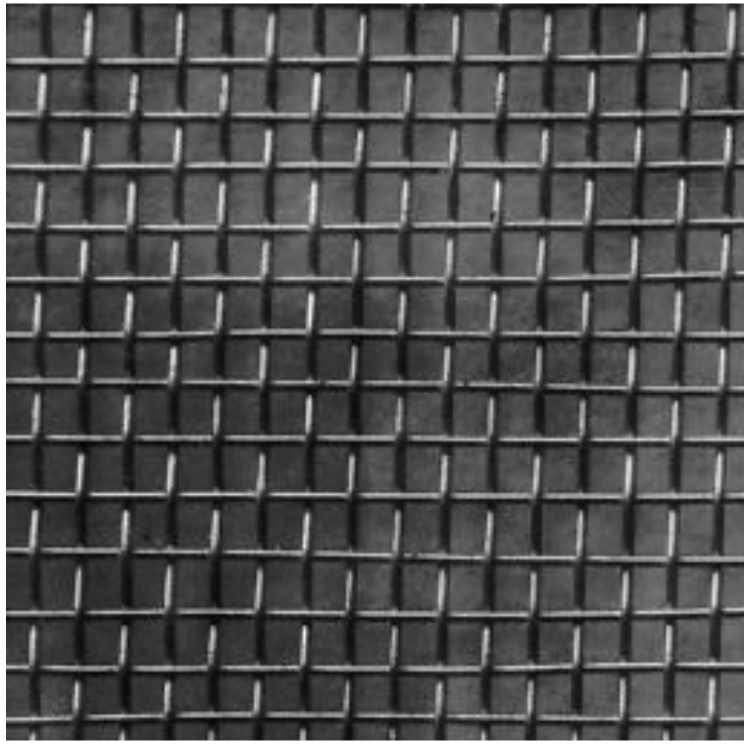

College Inn
Chicken Broth


College Inn
Chicken Broth

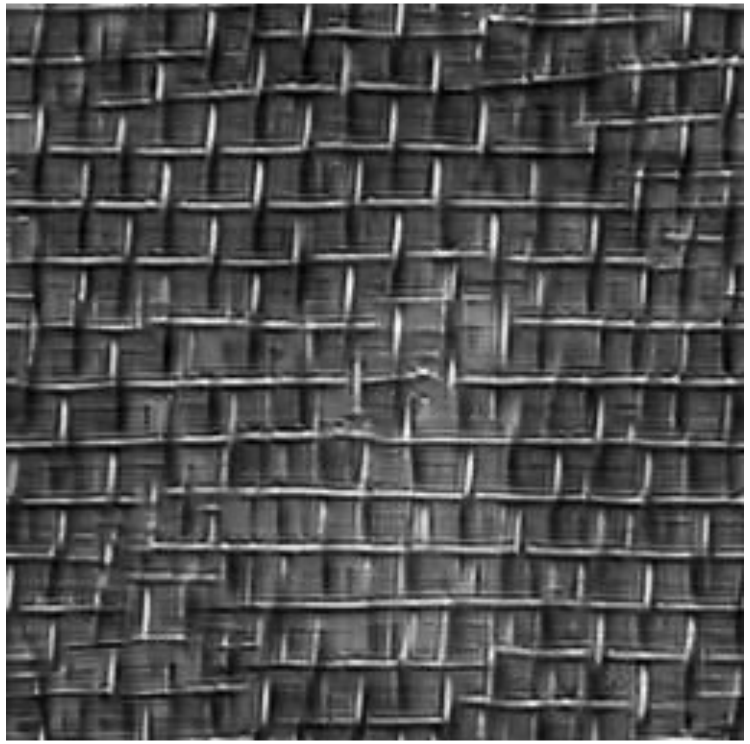




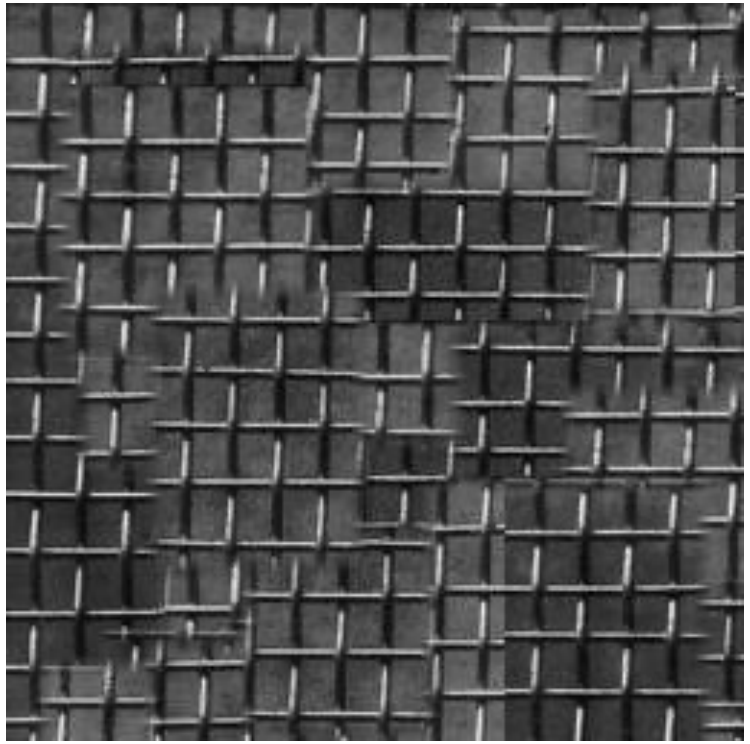
Failures
(Chernobyl Harvest)



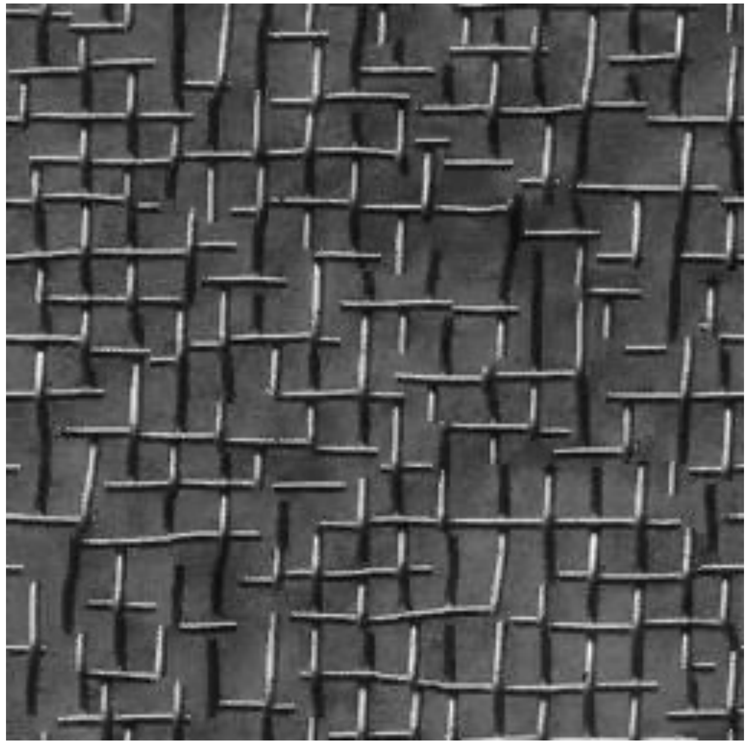
input image



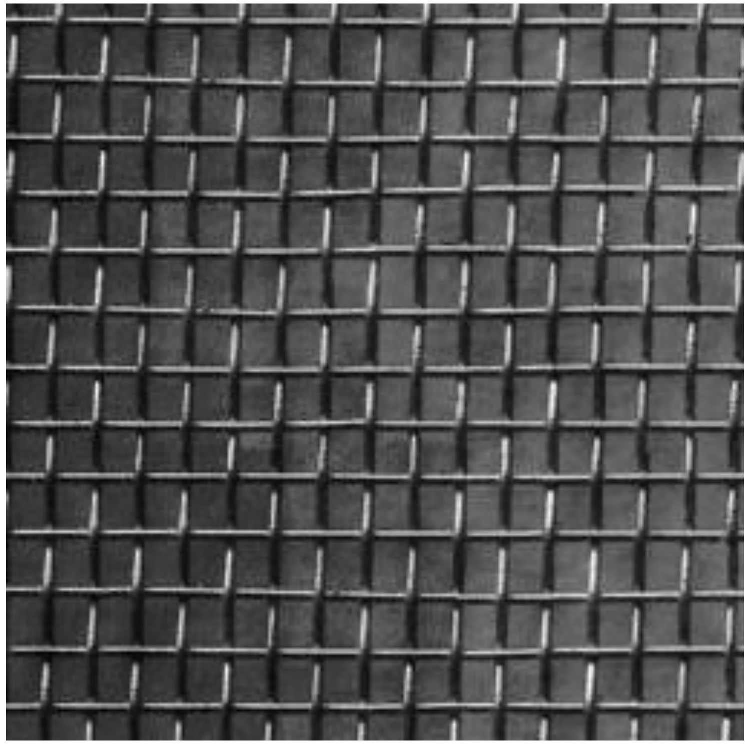
Portilla & Simoncelli



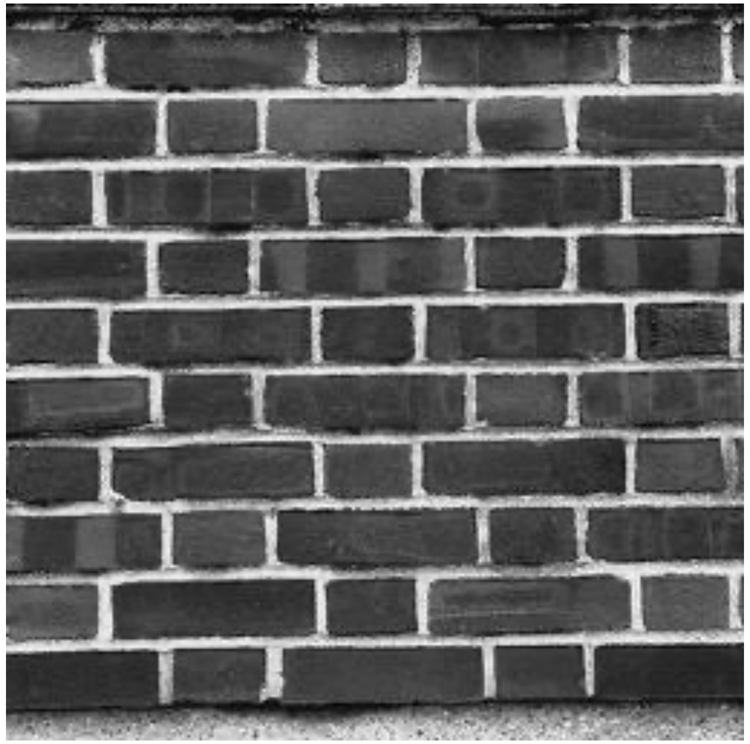
Xu, Guo & Shum



Wei & Levoy



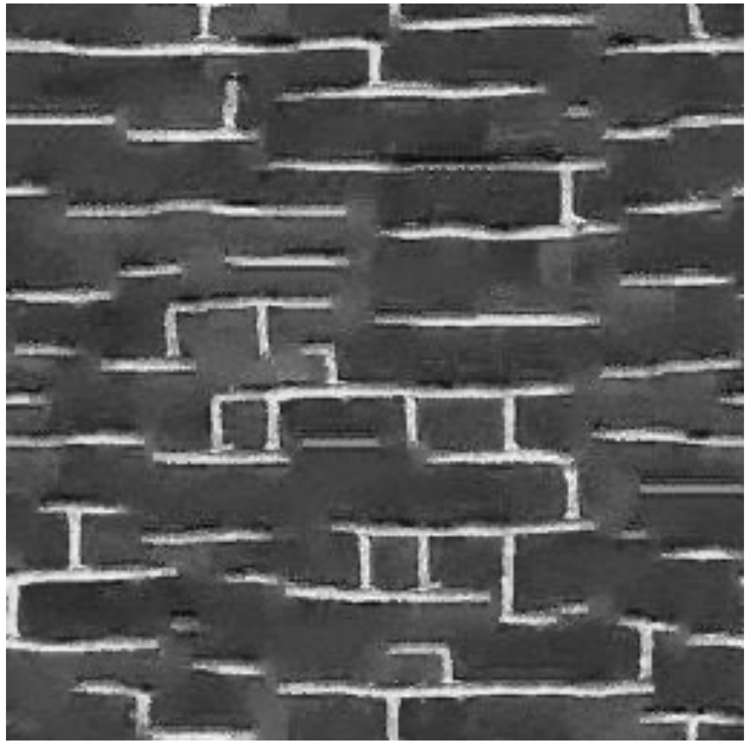
Efros and Freeman



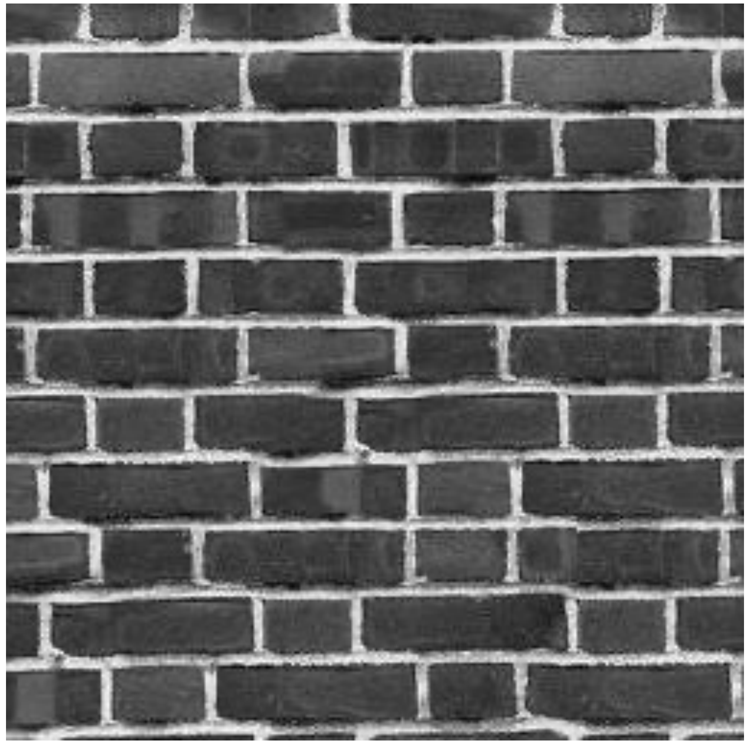
input image

Portilla & Simoncelli

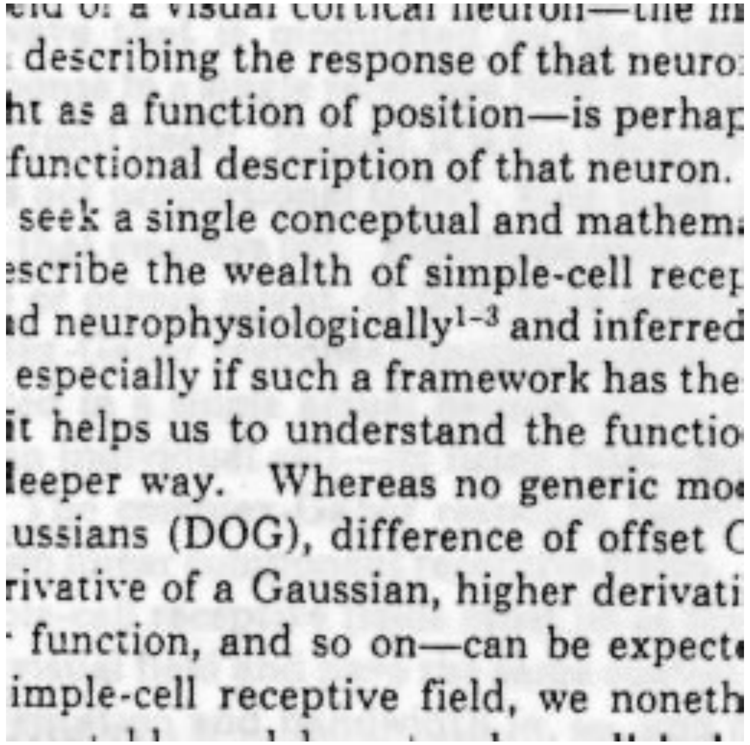
Xu, Guo & Shum



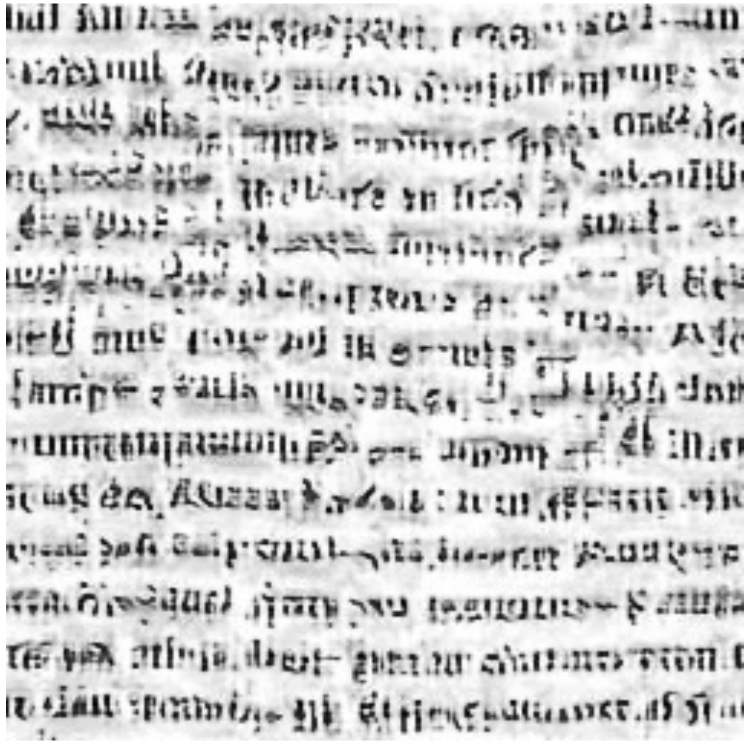
Wei & Levoy



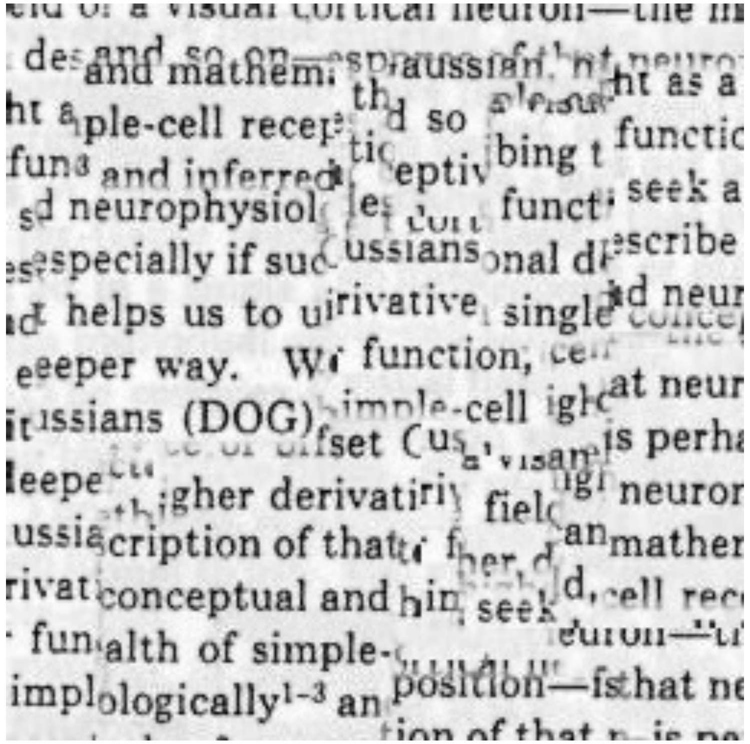
Efros and Freeman



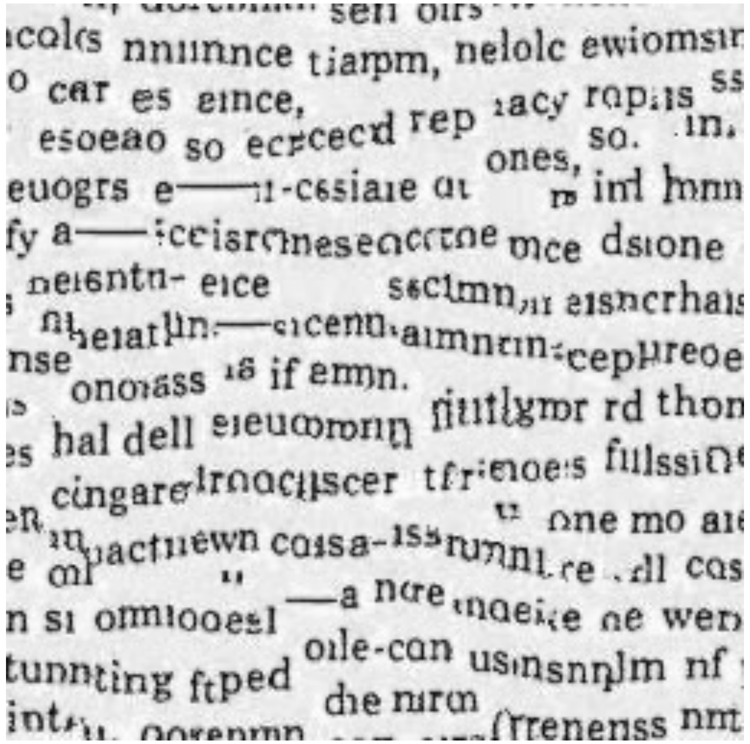
input image



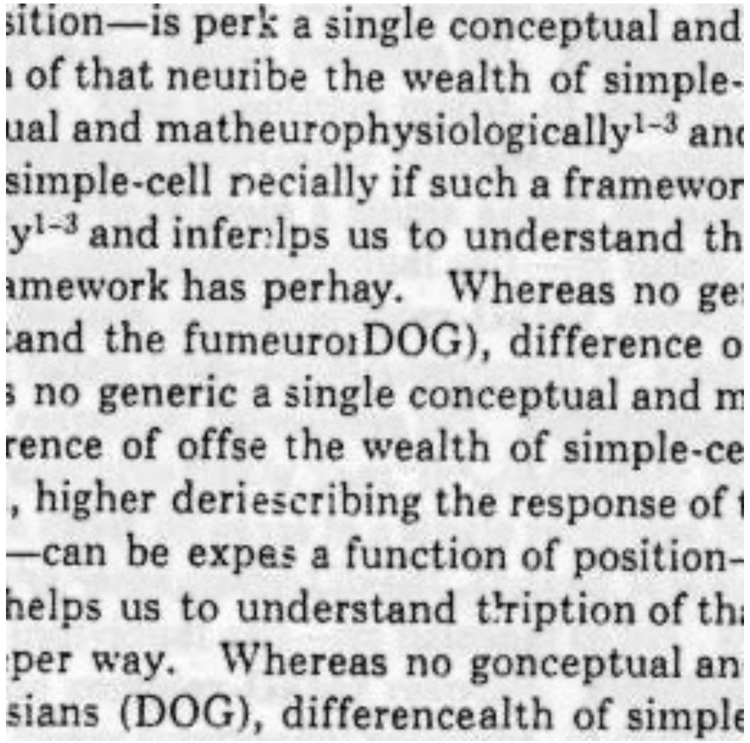
Portilla & Simoncelli



Xu, Guo & Shum



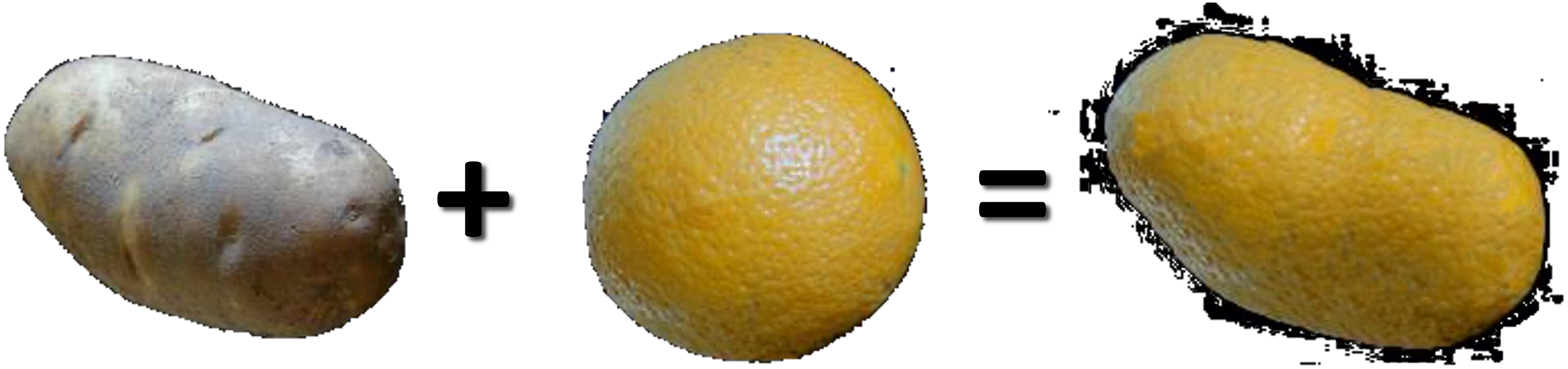
Wei & Levoy



Efros and Freeman

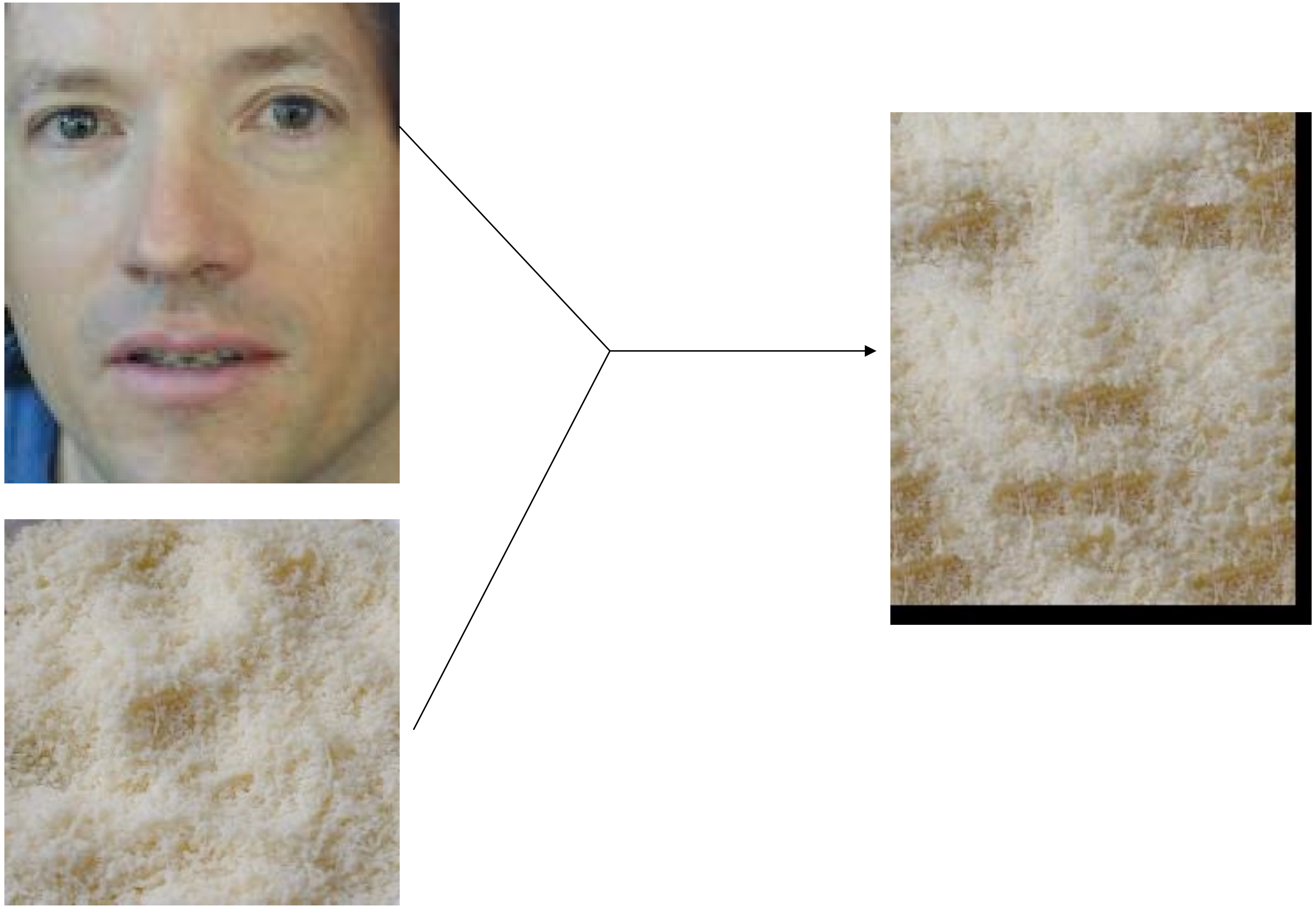
Application: Texture Transfer

- Try to explain one object with bits and pieces of another object:



Texture Transfer

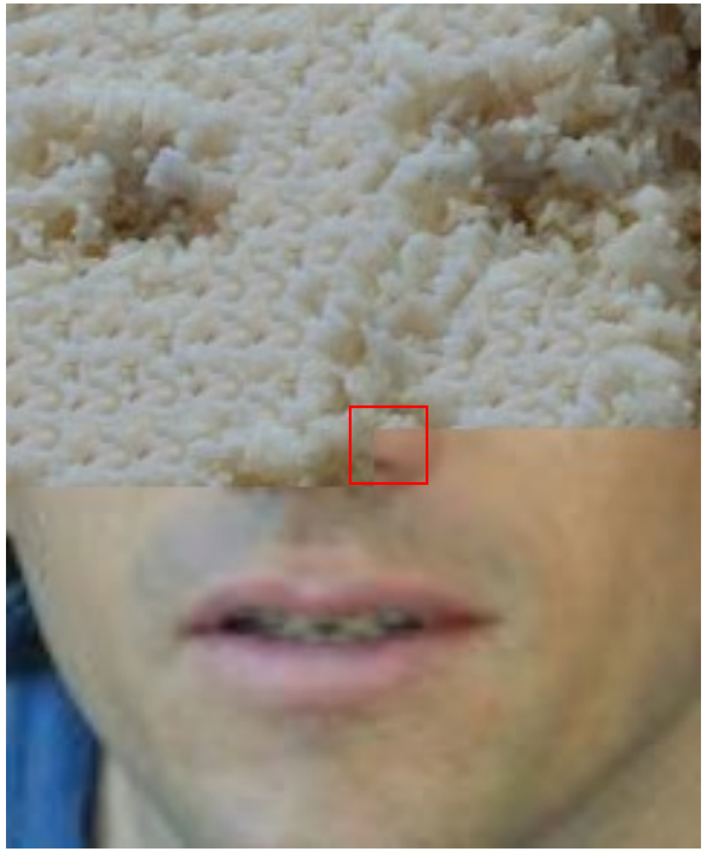
Constraint



Texture sample

Texture Transfer

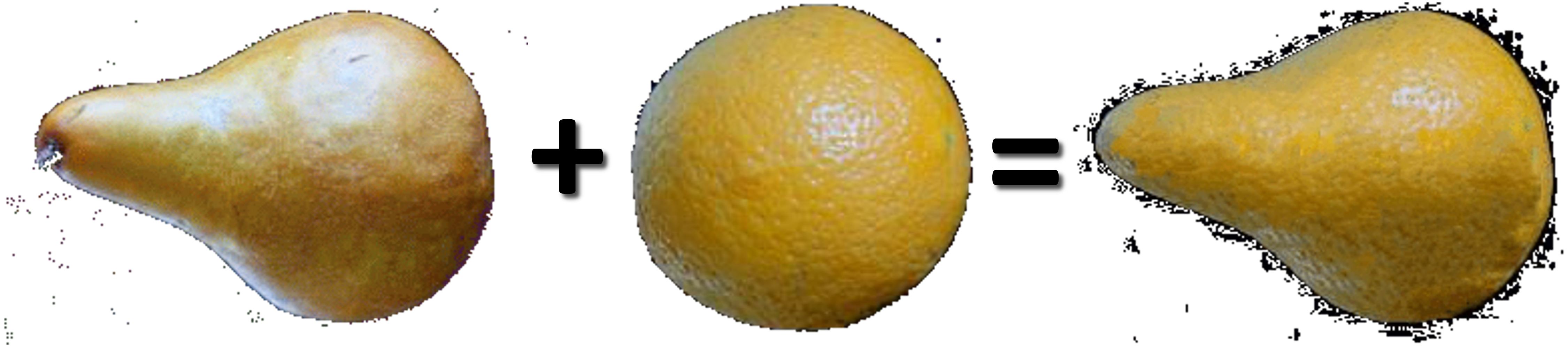
- Take the texture from one image and "paint" it onto another object



Same as texture synthesis, except an additional constraint:

1. Consistency of texture
2. Similarity to the image being "explained"

Texture Transfer



Texture Transfer

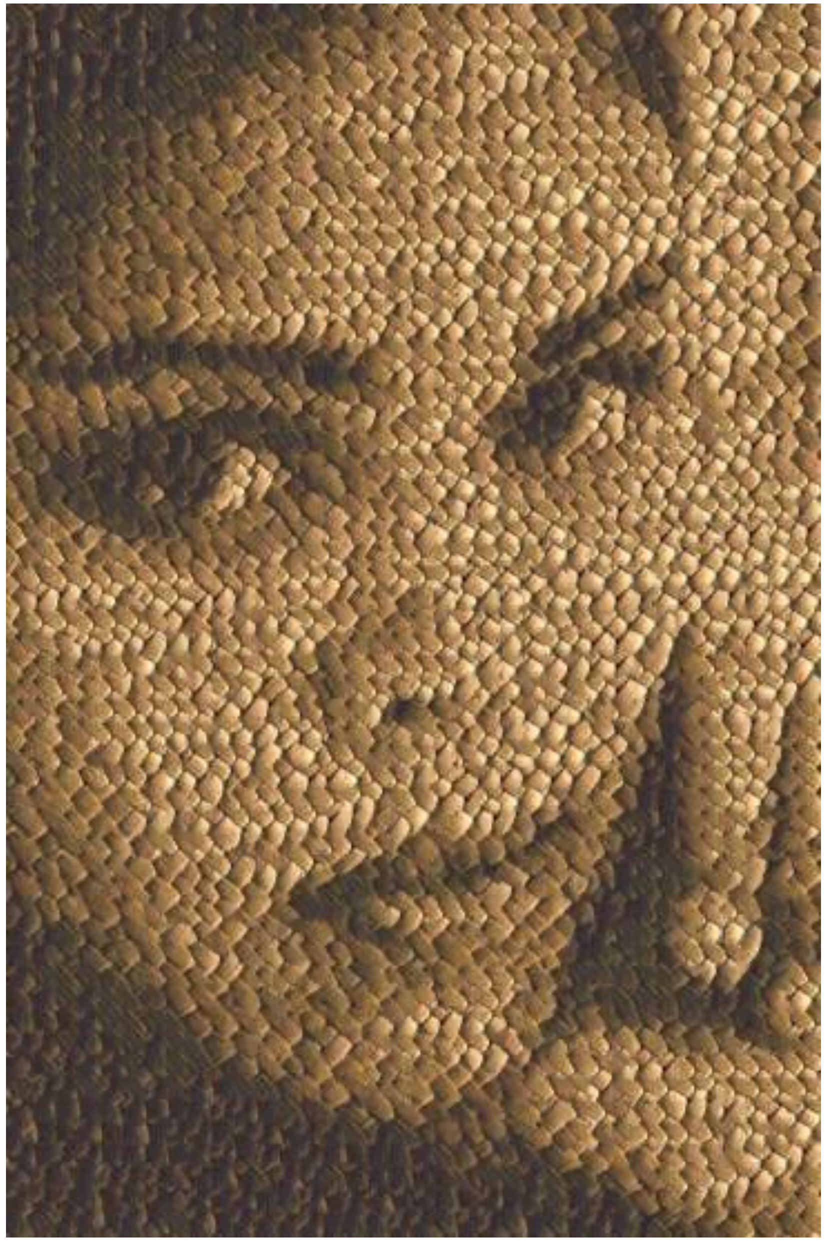


Image Analogies

Aaron Hertzmann[1,2]

Chuck Jacobs[2]

Nuria Oliver[2]

Brian Curless[3]

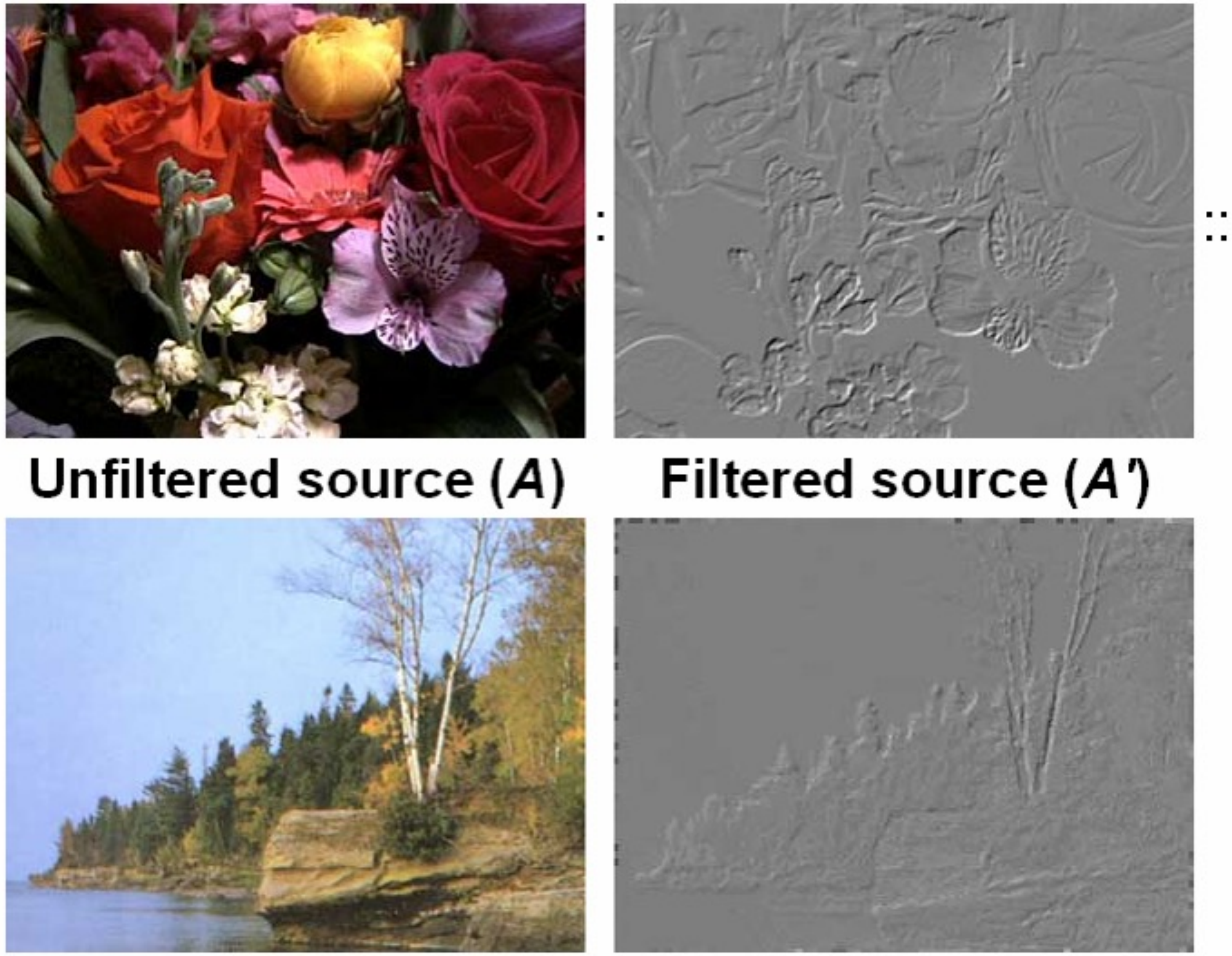
David Salesin[2,3]

[1]New York University

[2]Microsoft Research

[3]University of Washington

Edge Filter



Unfiltered source (A) Filtered source (A')

Unfiltered target (B) Filtered target (B')

Artistic Filters


A


A'

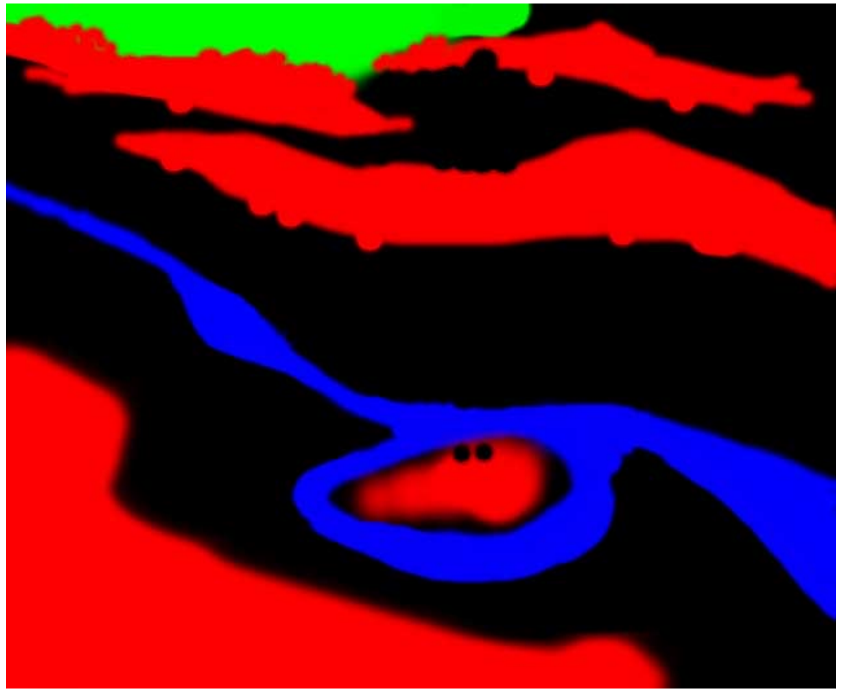

B


B'

Texture-by-numbers

A

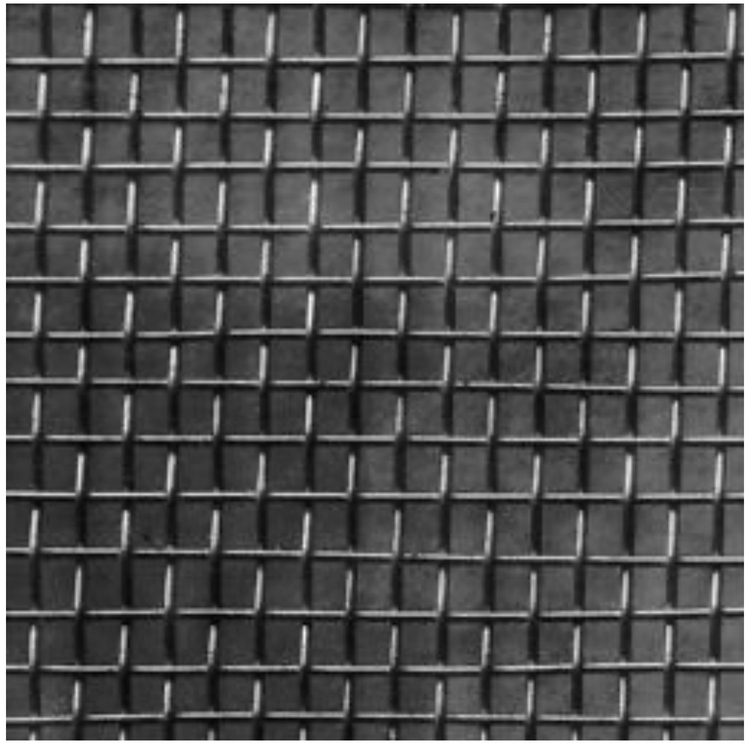
A’



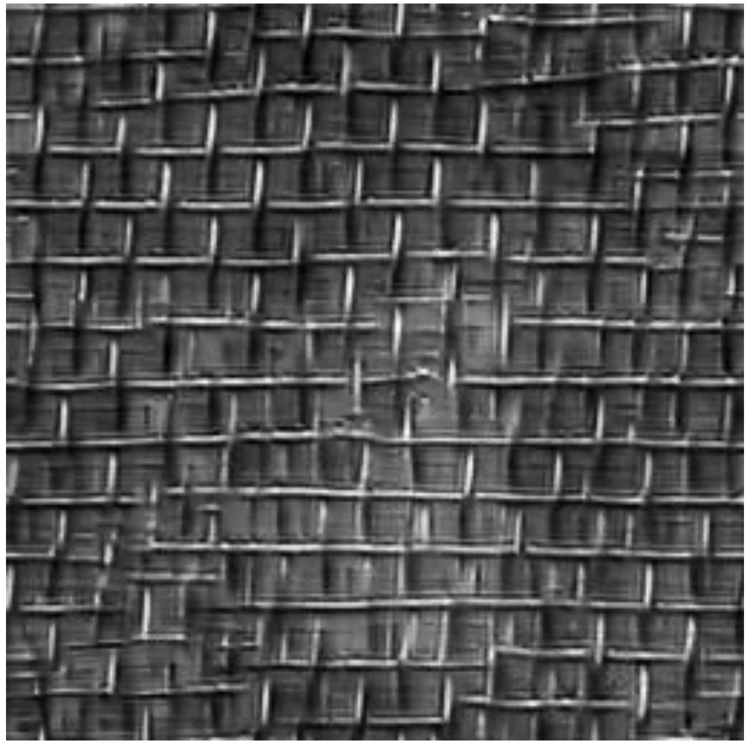
B

B’

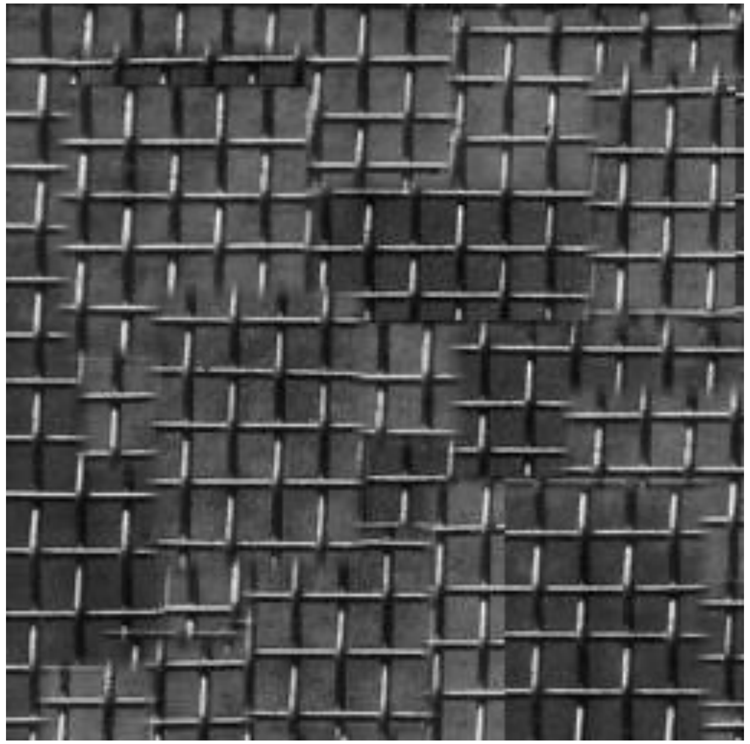
Parametric Texture Synthesis



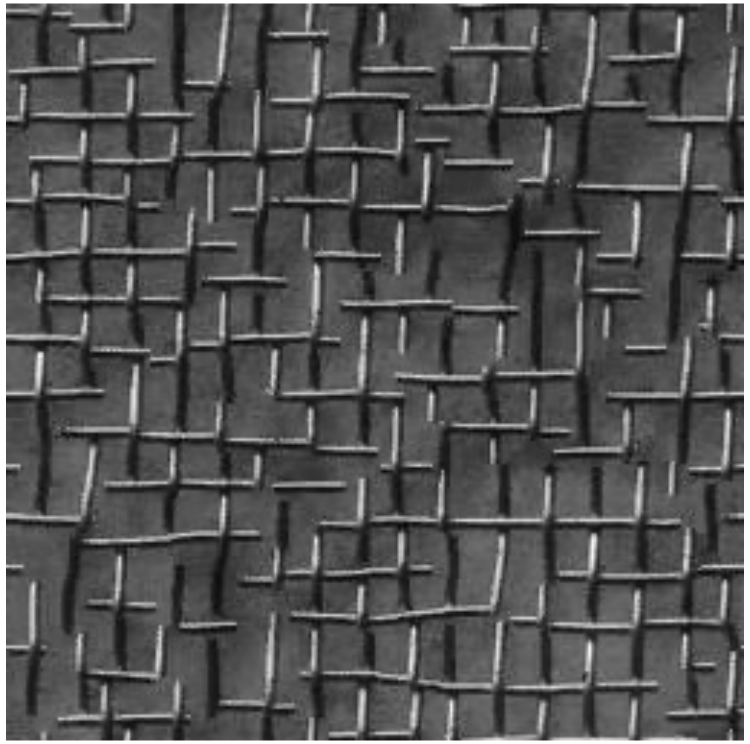
input image



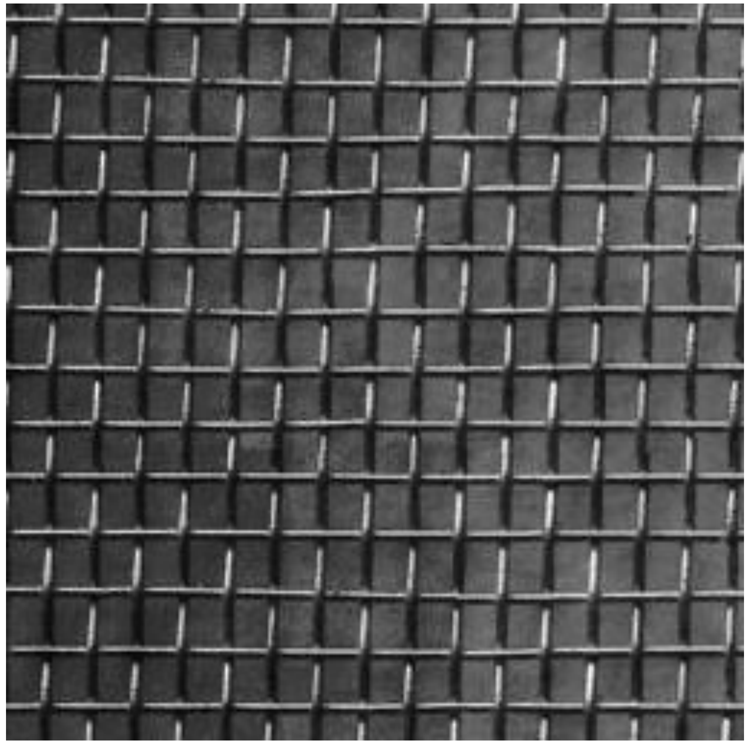
Portilla & Simoncelli



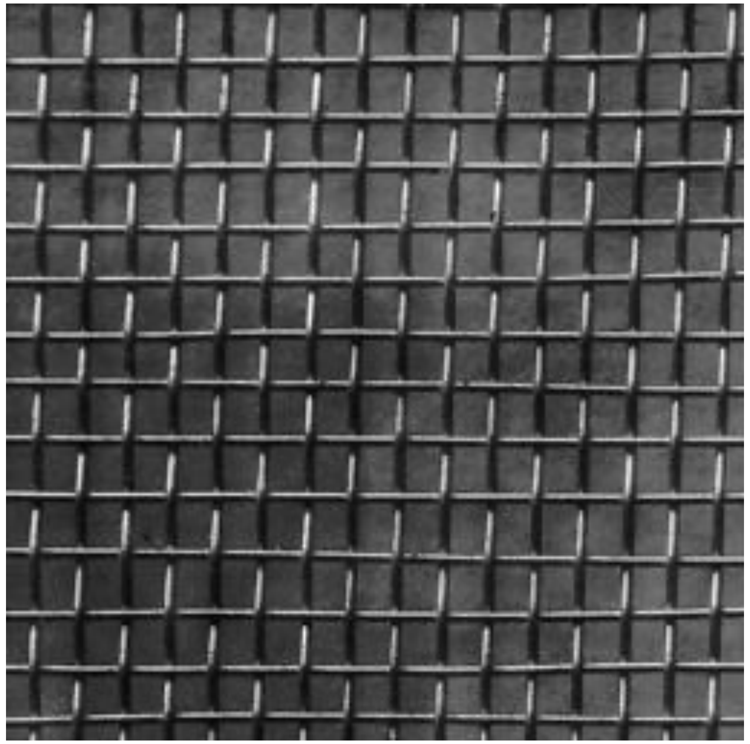
Xu, Guo & Shum



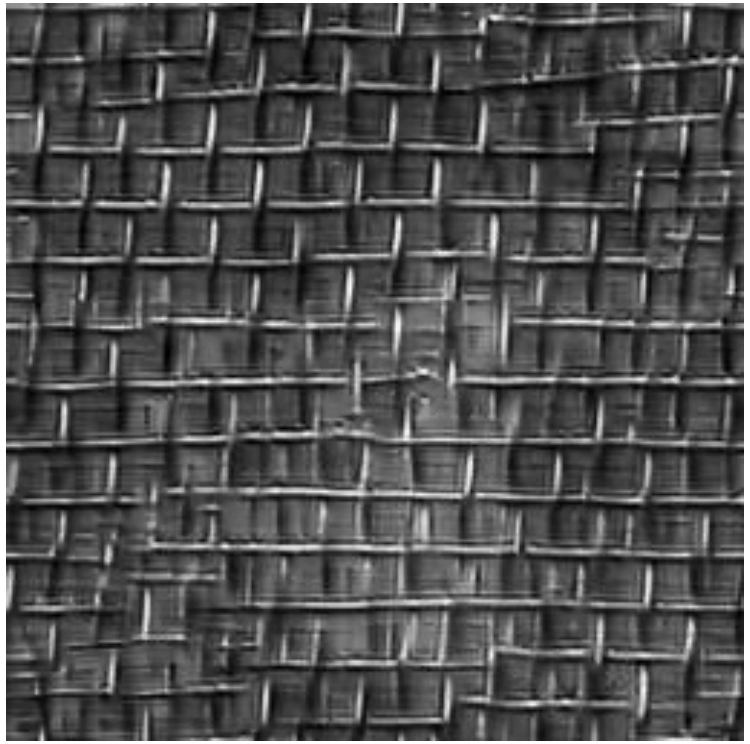
Wei & Levoy



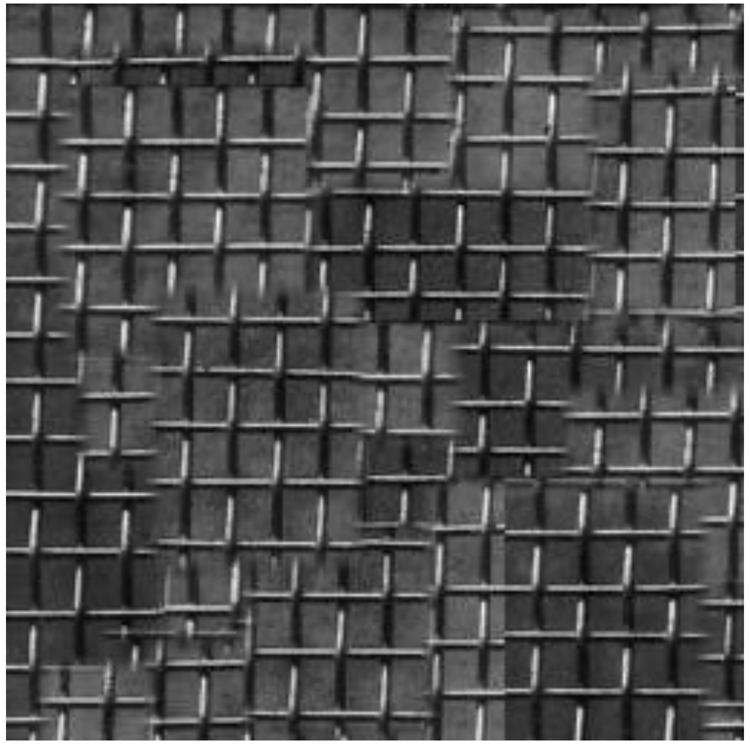
Efros and Freeman



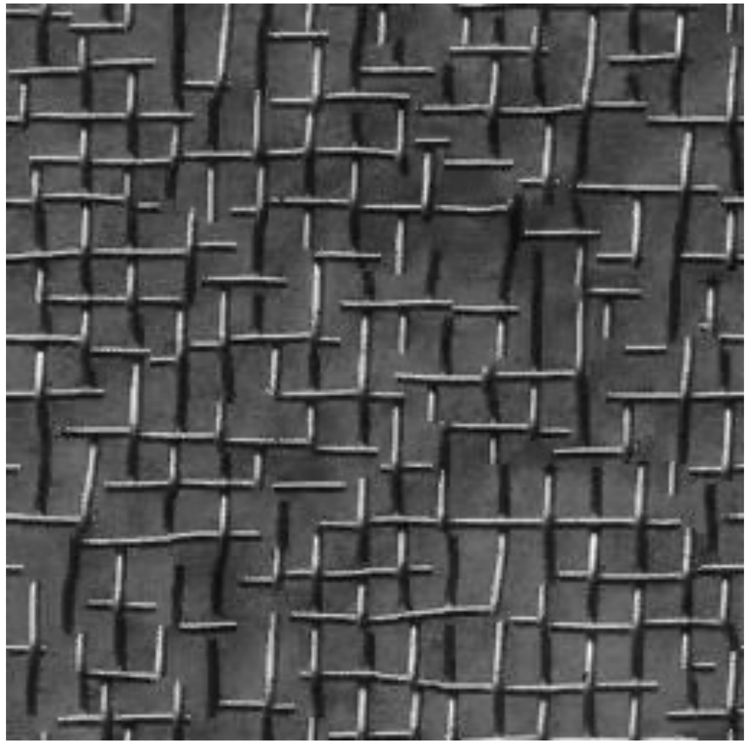
input image



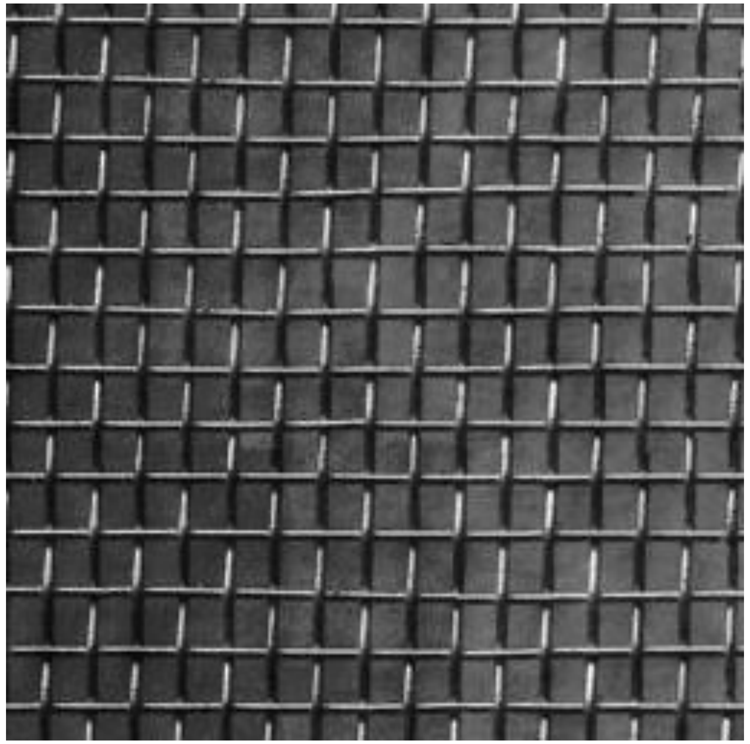
Portilla & Simoncelli



Xu, Guo & Shum

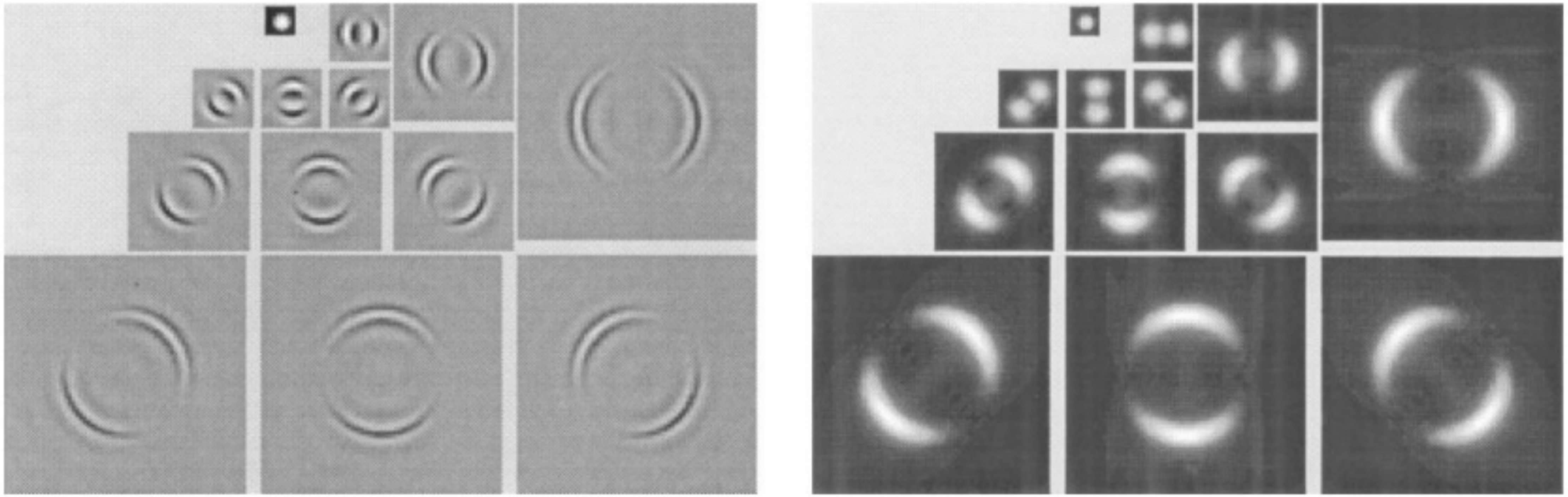


Wei & Levoy



Efros and Freeman

Parametric Texture Synthesis



Histogram and cross-channel correlation using wavelet basis

Statistics $\longrightarrow$ $\mathcal{E}(\phi_j(y)) \approx \mathcal{E}(\phi_j(\hat{y}))$

Wavelet features

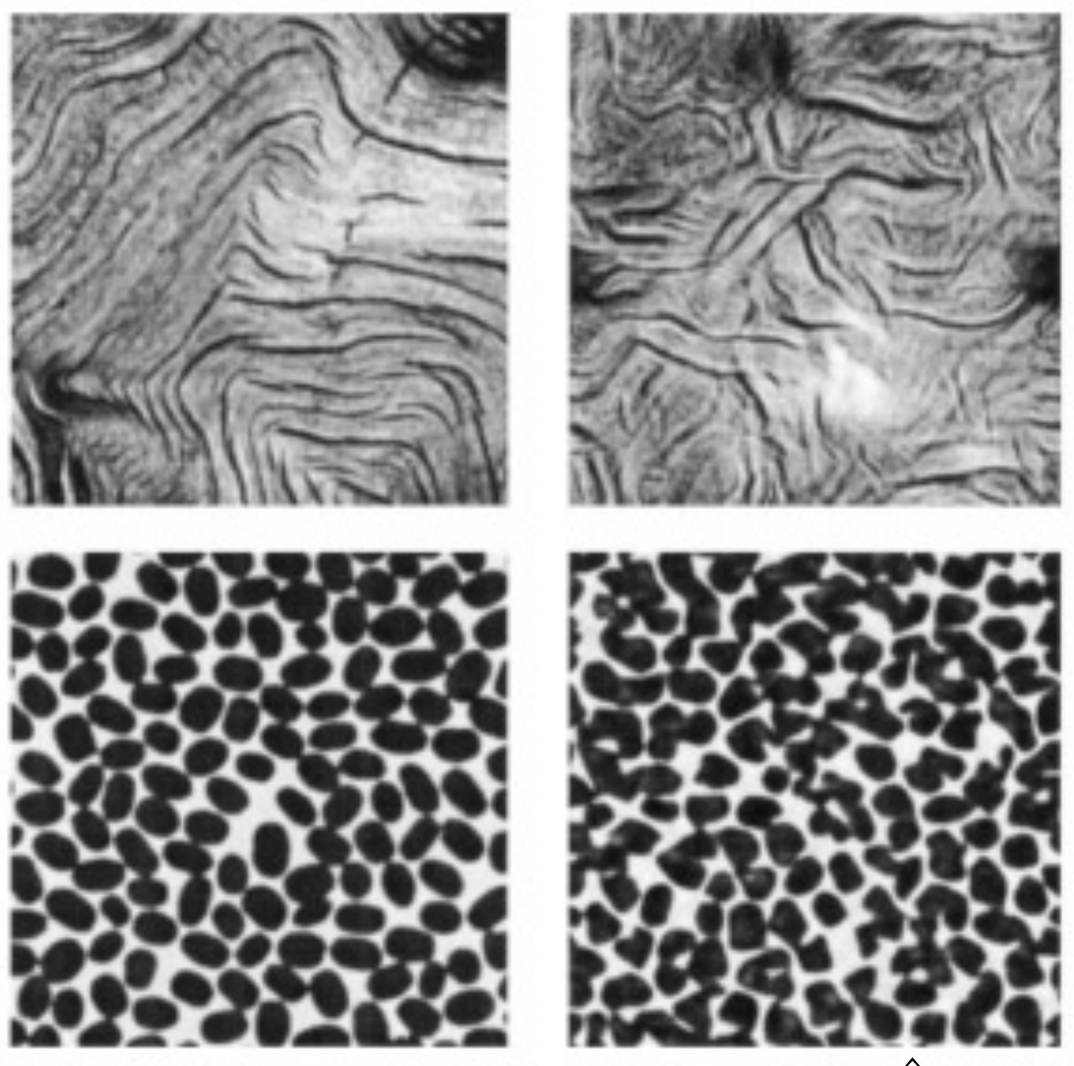
A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients

Portilla and Simoncelli, IJCV 1999

Parametric Texture Synthesis

<u>Objective function</u>
Given input texture y, feature descriptor ϕ,
and statistics summary function $\mathcal{E}$
We aim to optimize the output image $\hat{y}$

$$\hat{y}^* = \arg\min_{\hat{y}} ||\mathcal{E}(\phi_j(\hat{y})) - \mathcal{E}(\phi_j(y))||$$



Original y Output $\hat{y}$

A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients
Portilla and Simoncelli, IJCV 1999

Deep Learning Version

Gram matrix:
- Cross Correlation of CNN features
- Invariant to the feature locations

$$V = [v_1, v_2, \cdots, v_n]$$

$$G_{ij} = \langle v_i, v_j \rangle \quad G = V^\top V$$

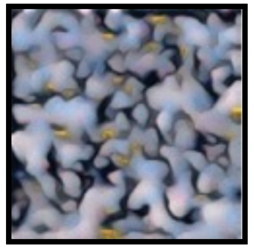
$$Gram^{(j)}(x) = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}.$$

h, w: pixel locations index
c: channel index
H, W: height and width of feature map
C: the number of total channels

[Gatys et al. 2015]

Style Reconstruction (Style Loss)

$\hat{y}$ y

|Gram() − Gram()|

optimized output style image

Gram = Gram Matrix of a deep network's features (e.g., ImageNet classifier)

Style Loss

weight (j)-th layer

$$\arg\min_{\hat{y}} \sum_{j}^{M} \lambda_j ||Gram^{(j)}(\hat{y}) - Gram^{(j)}(y)||^2$$


pool3
pool4
original
Portilla & Simoncelli

Number of parameters

Different layers

The same network architecture with random weights

Neural Style Transfer

content image

+

style image

=

output result

[Gatys et al. 2016]

Content Reconstruction (Perceptual Loss)

$\hat{y}$ x

$|F($  $) - F($  $)|$

optimized output content image

F is a deep network (e.g., ImageNet classifier)

Content Loss

weight (i)-th layer

$$\arg\min_{\hat{y}} \sum_{i}^{N} \lambda_i ||F^{(i)}(\hat{y}) - F^{(i)}(x)||_1$$

Content Reconstruction (Perceptual Loss)

Conv1_2 Conv2_2 Conv3_2 Conv4_2 Conv5_2

Neural Style Transfer

$\hat{y}$

$|\mathrm{Gram}($  $) - \mathrm{Gram}($

y

 $)|$

optimized output

style image

$\hat{y}$

$+ \quad |\mathrm{F}($  $) - \mathrm{F}($

x

 $)|$

optimized output

content image

$$\arg\min_{\hat{y}} \mathcal{L}_{\text{style}}(\hat{y}, y) + \lambda \mathcal{L}_{\text{content}}(\hat{y}, x)$$


A
B
C
D

Different Initializations

Fast Neural Style Transfer

- Optimization-based method

$$\arg\min_{\hat{y}} \mathcal{L}_{\text{style}}(\hat{y}, y) + \lambda \mathcal{L}_{\text{content}}(\hat{y}, x)$$

- Feedforward network

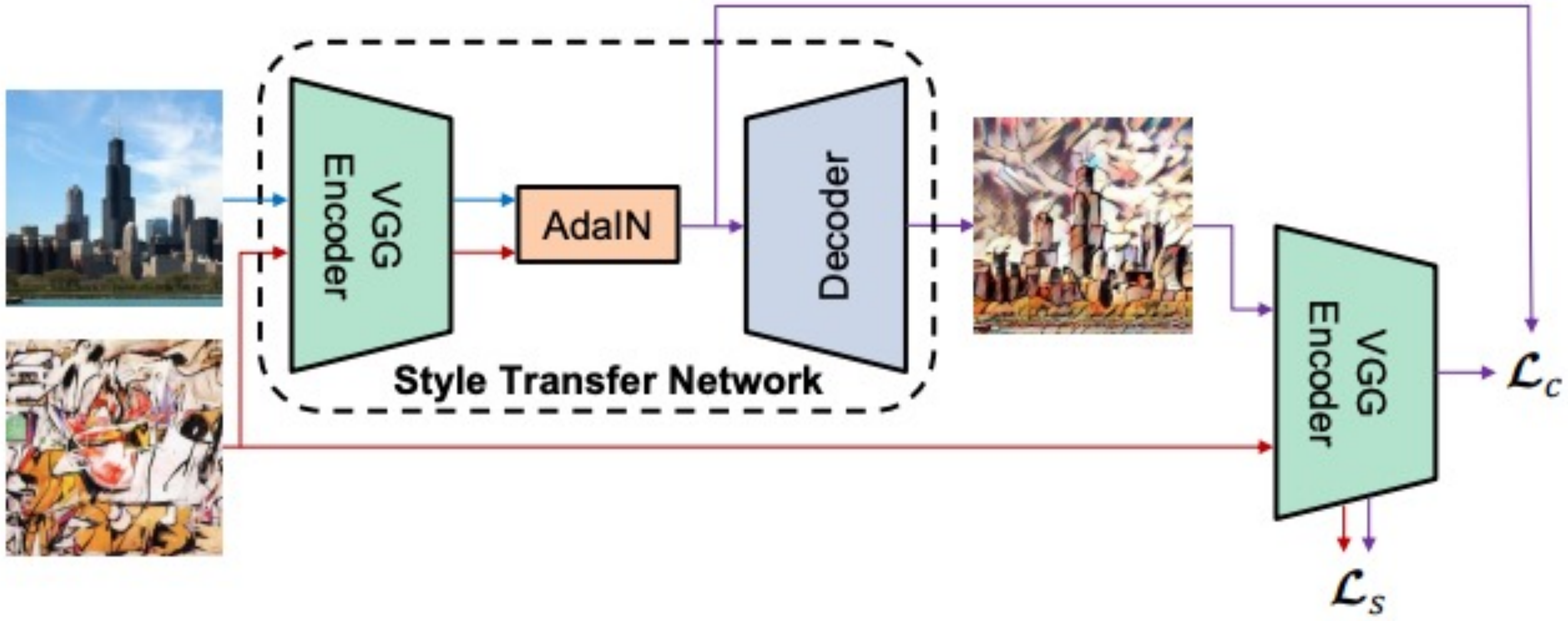
$$\arg\min_{G} \mathbb{E}_x \, \mathcal{L}_{\text{style}}(G(x), y) + \lambda \mathcal{L}_{\text{content}}(G(x), x)$$

[Johnson et al. 2016]

Arbitrary Style Transfer with AdaIN

- Feedforward network with any style

$$\arg\min_G \mathbb{E}_{x,y}\, \mathcal{L}_{\text{style}}(G(x,y),y)+\lambda\mathcal{L}_{\text{content}}(G(x,y),x)$$



Huang et al. 2017

Arbitrary Style Transfer with AdaIN

Huang et al. 2017

Neural Style Transfer
vs.
Image-to-Image Translation


Input
Style Image I
Style image II
Entire collection
CycleGAN
Photo → Van Gogh
Input
Style image I
Style image II
Entire collection
CycleGAN
horse → zebra

Photo Style Transfer

Deep Photo Style Transfer

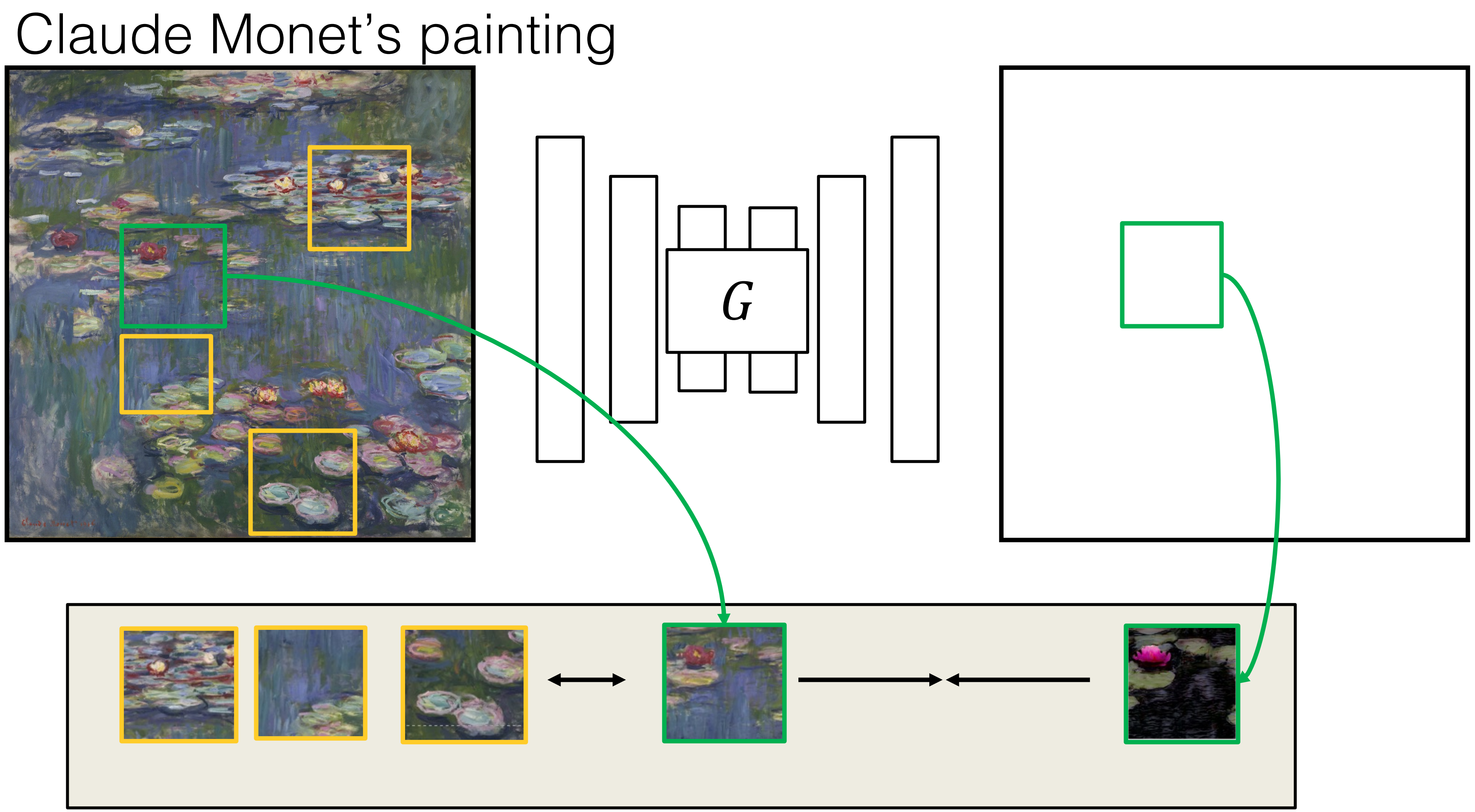
(a) Reference style image (b) Input image (c) Neural Style (distortions) (d) Our result

Local color transfer? (hard to transfer texture)

[Luan et al., 2017]

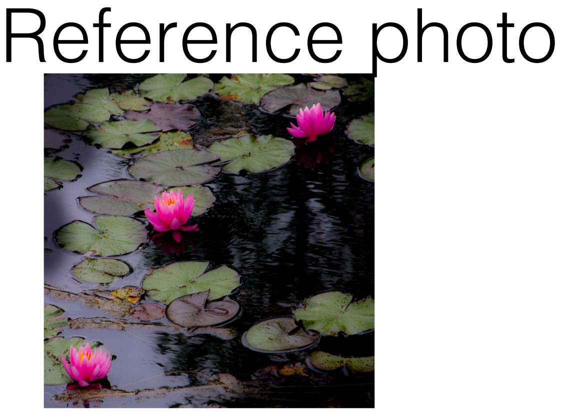
Single Image Translation

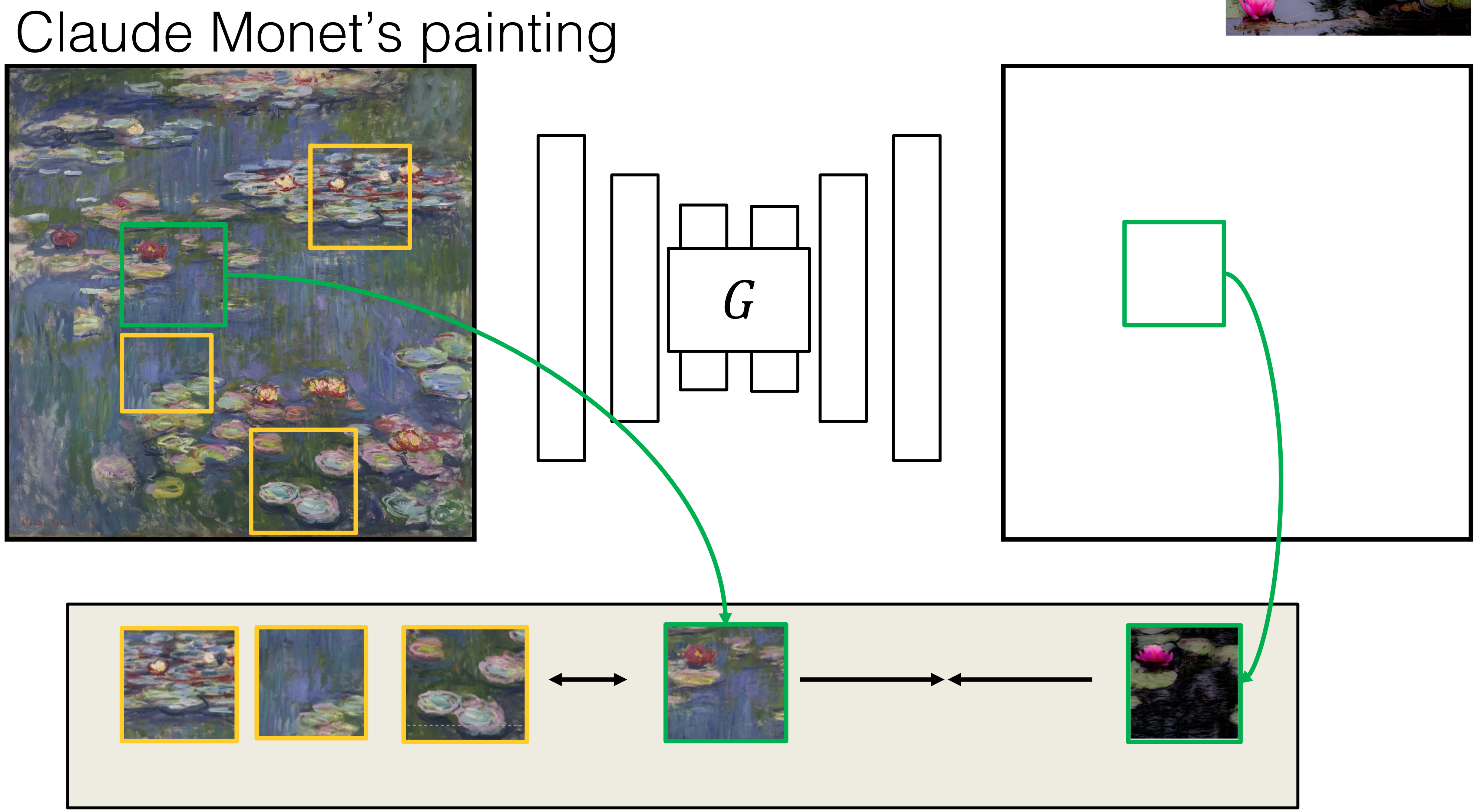
Single Image Translation



Internal contrastive loss is well-suited for single image translation.
Also see InGAN (Shocher et al., ICCV'19), SinGAN (Shaham et al., ICCV'19)

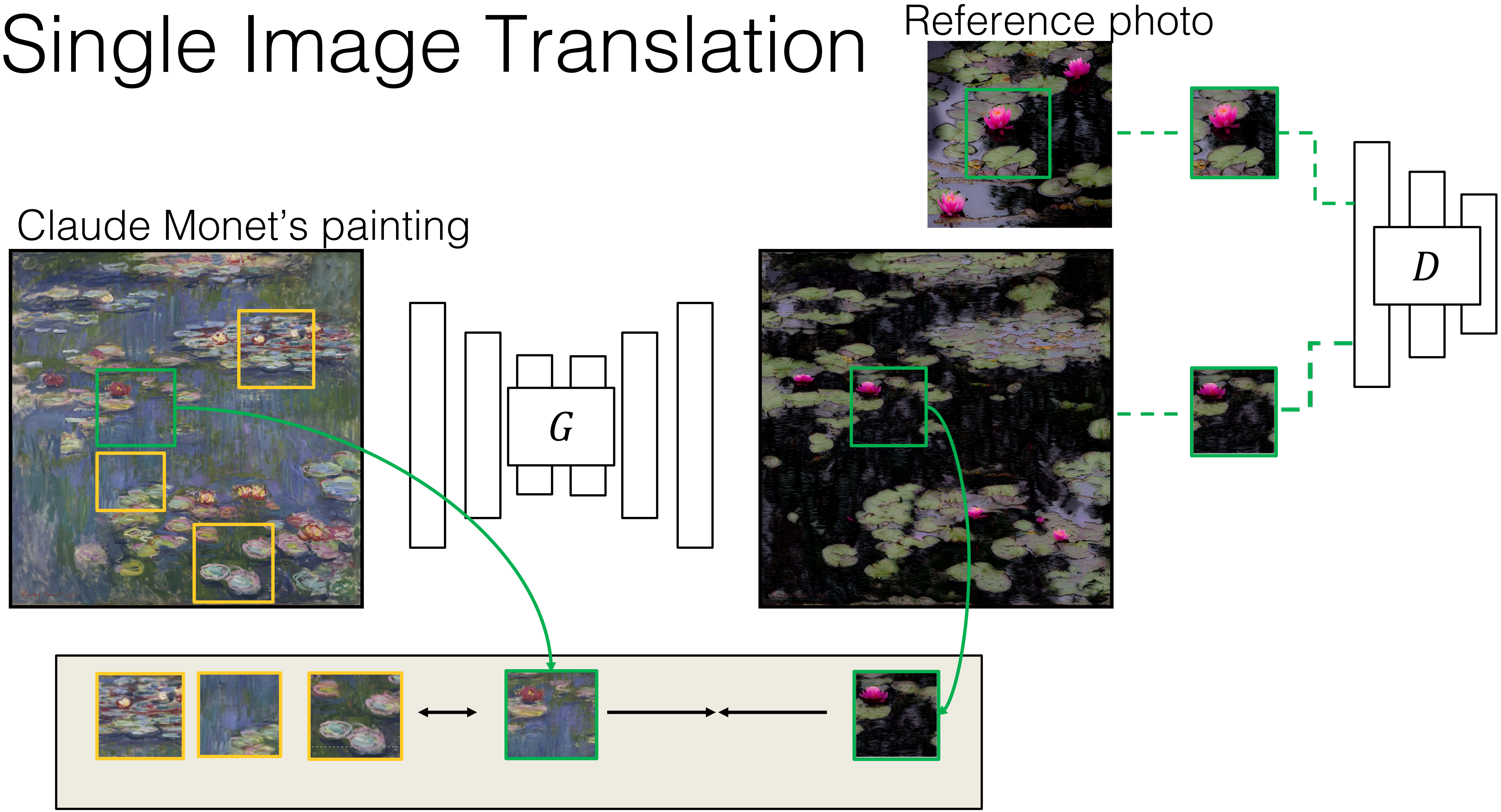
Single Image Translation



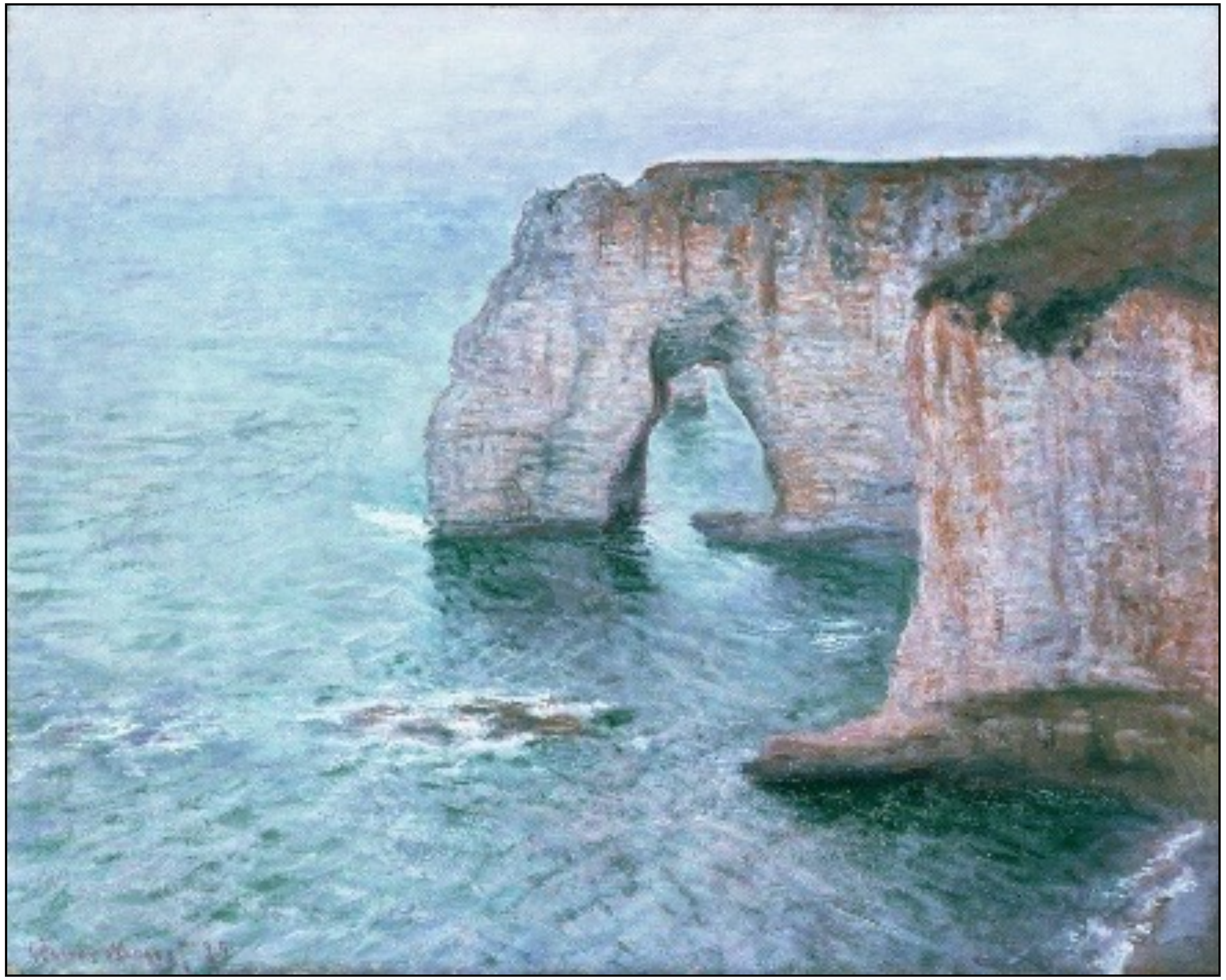


Internal contrastive loss is well-suited for single image translation.
Also see InGAN (Shocher et al., ICCV’19), SinGAN (Shaham et al., ICCV’19)

Single Image Translation



Internal contrastive loss is well-suited for single image translation.
Also see InGAN (Shocher et al., ICCV'19), SinGAN (Shaham et al., ICCV'19)



Painting

Reference

Painting

Reference

Painting

Gatys et al. CVPR’16

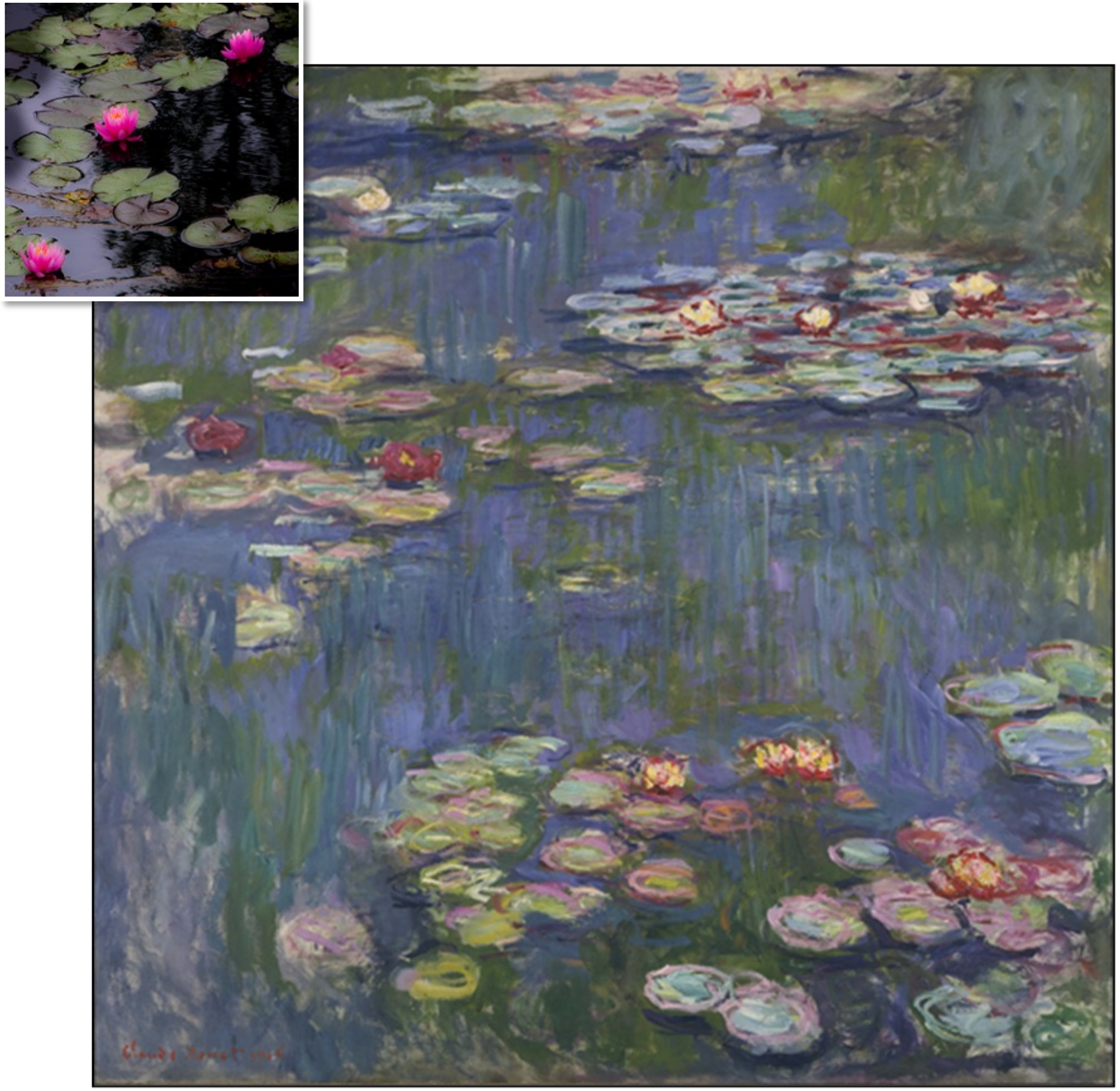
Reference

Painting

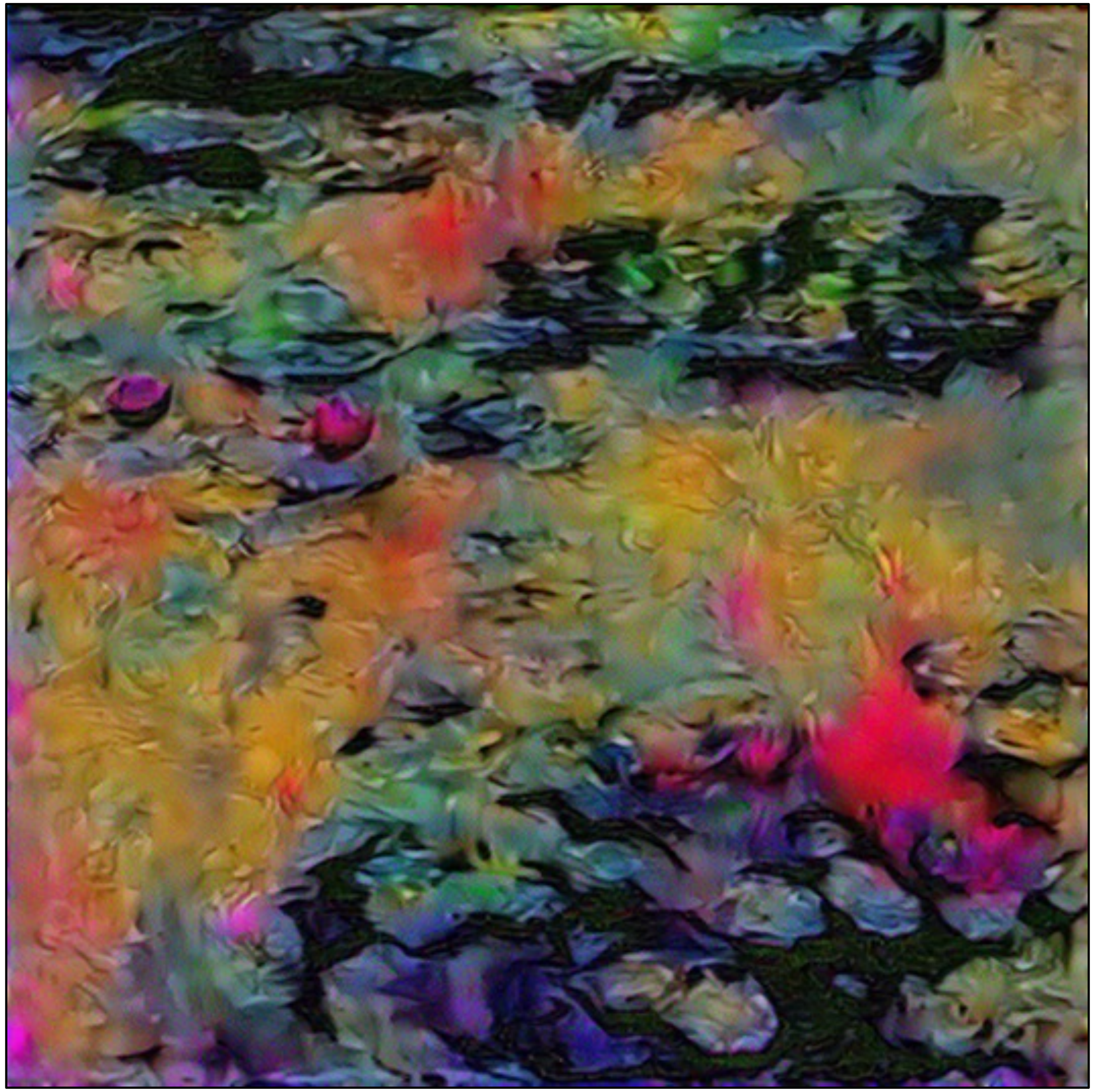
Single Image translation (CUT)

Painting

Reference

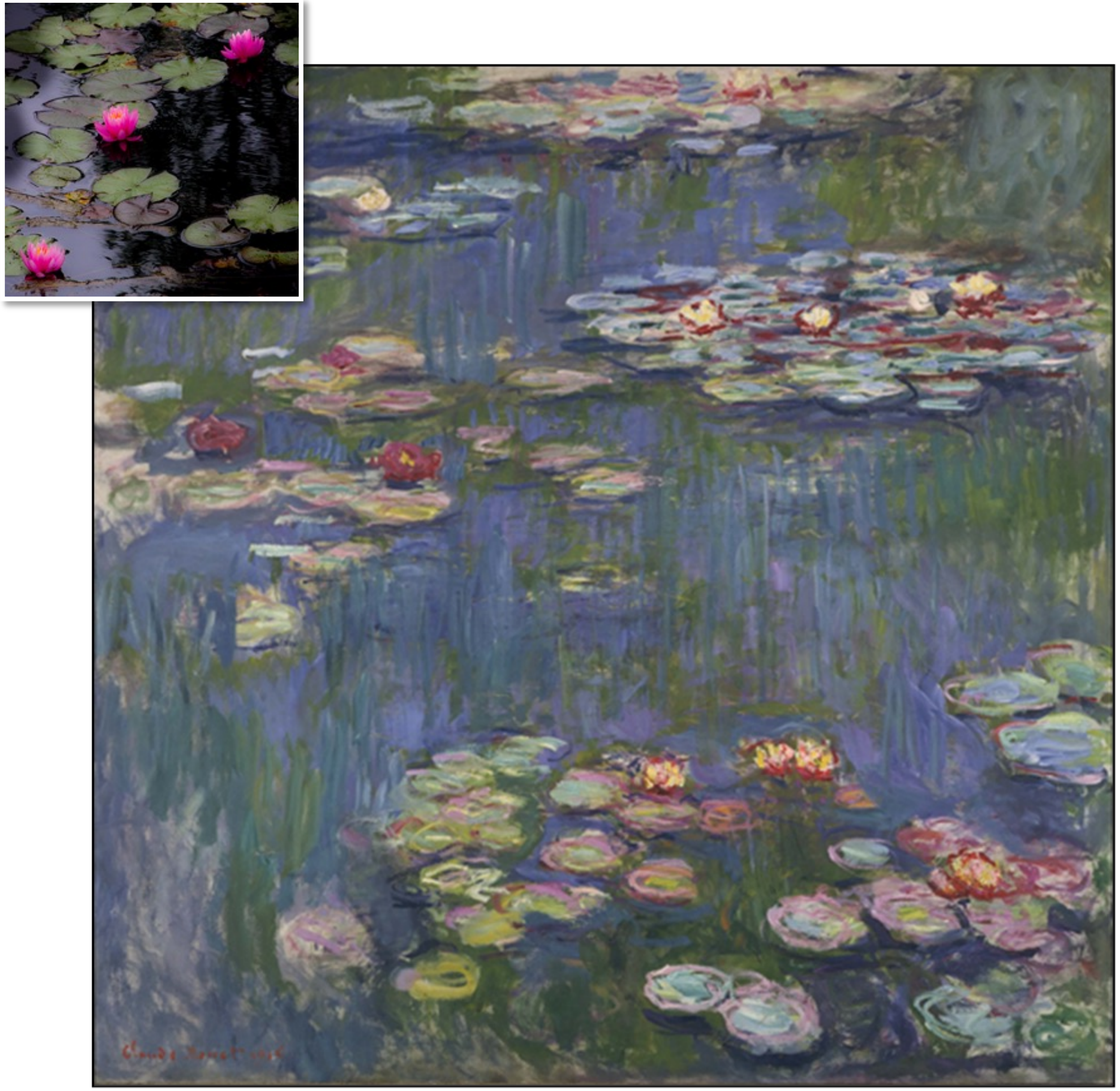


Painting



Gatys et al. CVPR'16

Reference



Painting

Single Image translation (CUT)

Style Transfer vs. Image-to-Image Translation

- Data (how to define Style)
  - A single image? A collection of images
- Applications
  - Photo -> Painting (Neural Style Transfer, Image-to-Image Translation)
  - Photo -> Photo (Image-to-Image Translation, Photo Style Transfer (Color))
  - Painting -> Photo (Image-to-Image Translation, Deep Image Analogy)
- Algorithms:
  - Patch-based method (or dense correspondence)
  - Optimization-based method
  - Feed-forward network
- Loss functions
  - Style Loss: GAN loss, Gram matrix loss
  - Content Loss: Perceptual Loss, Cycle-consistency loss, Contrastive Loss (InfoNCE)