



Style and Content, Texture Synthesis

Jun-Yan Zhu

16-726, Spring 2025

Many slides are borrowed from Alyosha Efros, Lvmin Zhang,
Maneesh Agrawala, Bill Freeman

photo © [Gatys et al., 2016]

Collection Style Transfer



Photograph ©Alexei Efros



Monet



Van Gogh



Cezanne



Ukiyo-e

Style and Content Separation

A

Classification

A	B	C	D	E
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
A	B	C	D	E
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
A	B	C	D	E
B	C	A	E	D

Domain Adaptation

B

Extrapolation

A	B	C	D	E
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
A	B	C	D	E
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>
A	B	C	D	E
?	?	C	D	E

Paired Image-to-Image Translation

C

Translation

A	B	C	D	E	?	?	?
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>			
A	B	C	D	E			
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>			
A	B	C	D	E	?	?	?
?	—	—	—	?	F	G	H

Unpaired Image-to-Image Translation

Training
Generalization

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

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



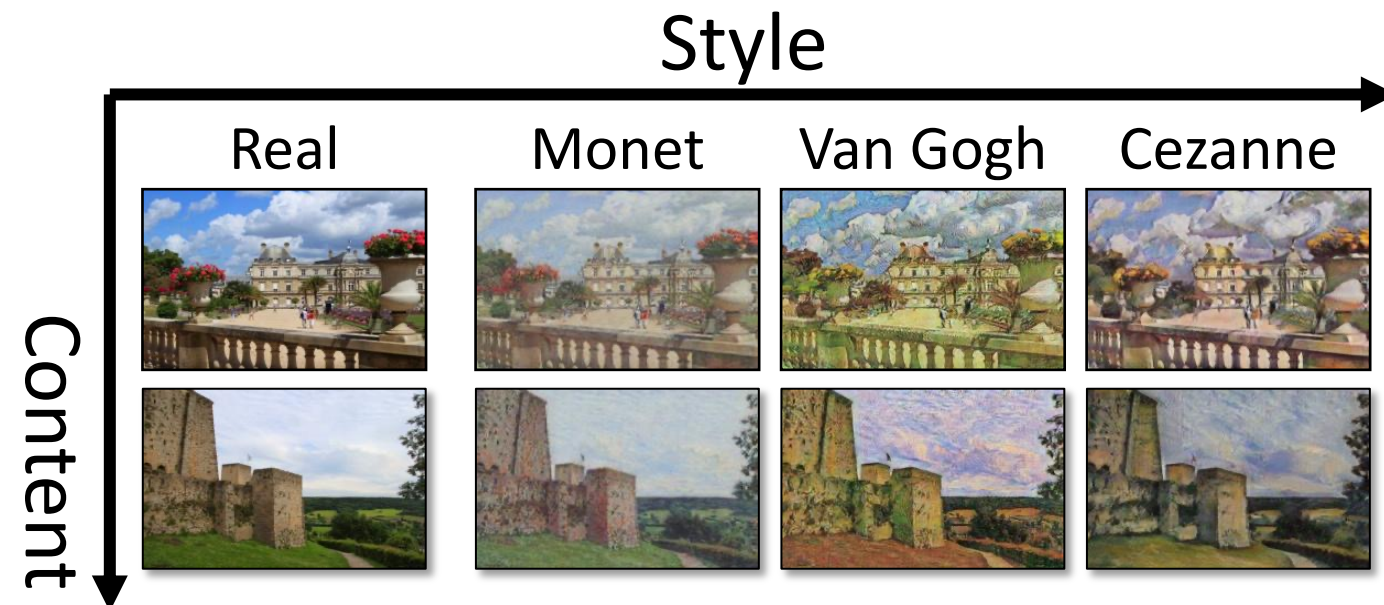
$p(x) \rightarrow p(y)$ change **style**

Cycle-consistency loss

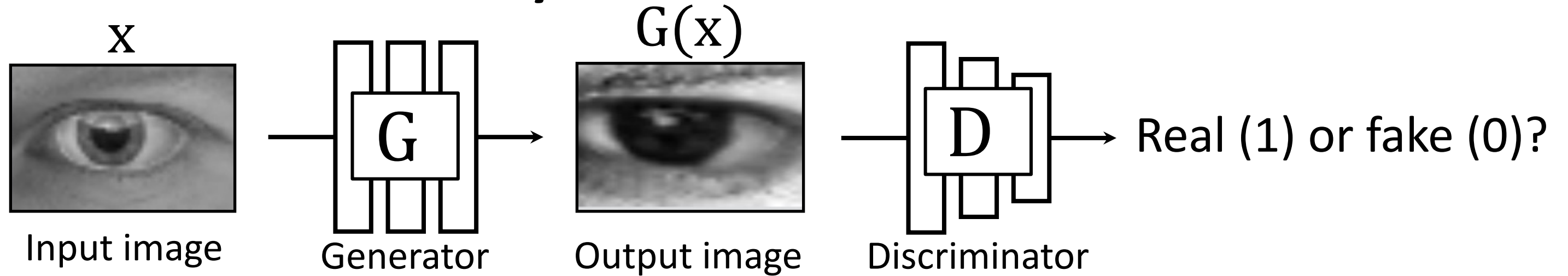
$$\mathbb{E}_x ||F(G(x)) - x||_1$$



Bidirectional: preserve **content**



Style and Content

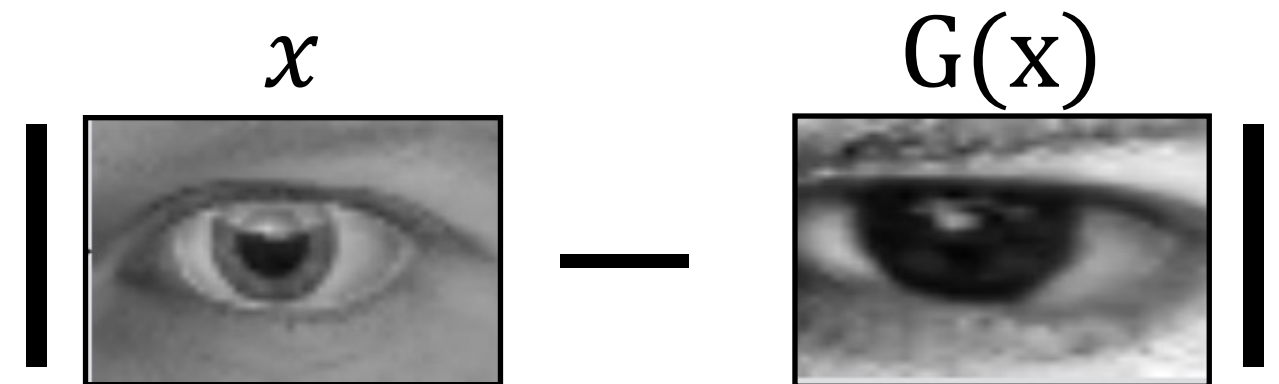


Adversarial loss (change style)

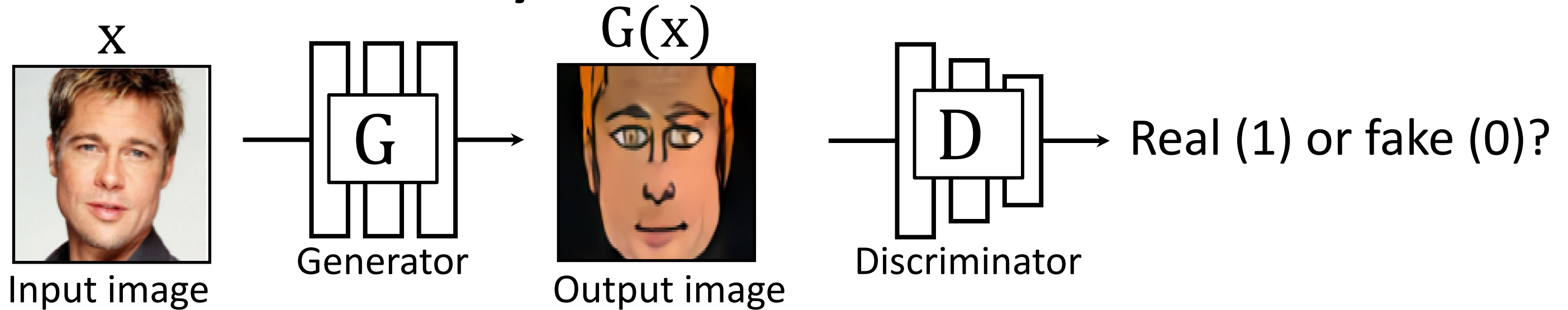
$$\mathbb{E}_x \log(1 - D_Y(G(x))) + \mathbb{E}_y \log D_Y(y)$$

L1 loss (preserve content in pixel space)

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



Style and Content

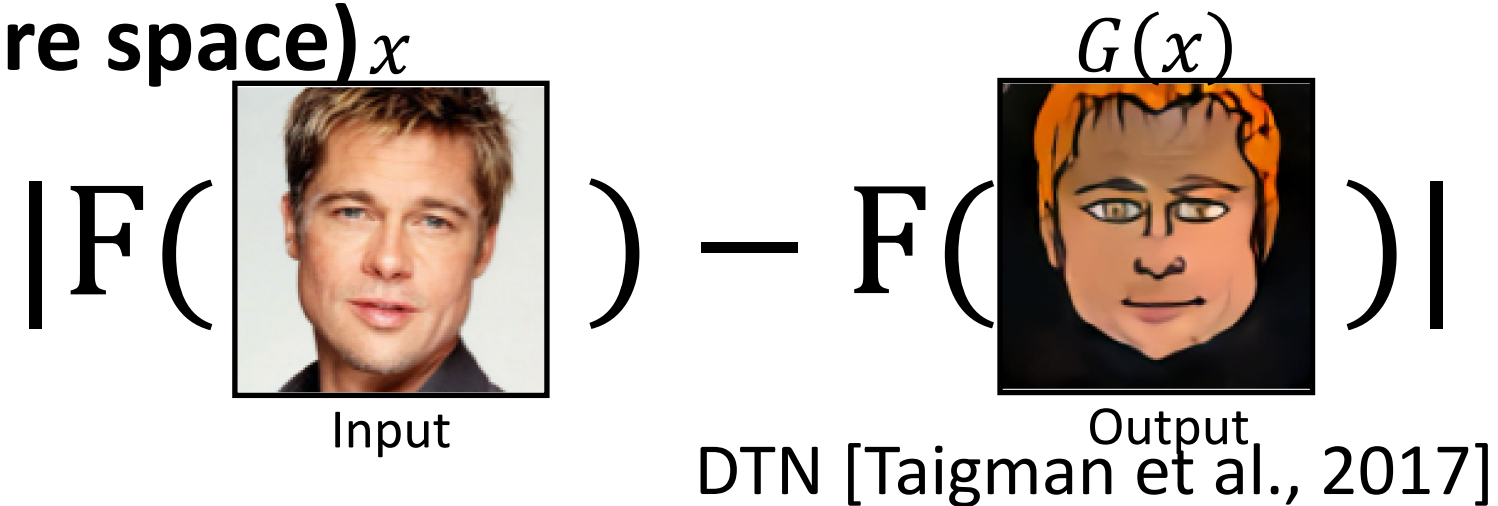


Adversarial loss (change style)

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

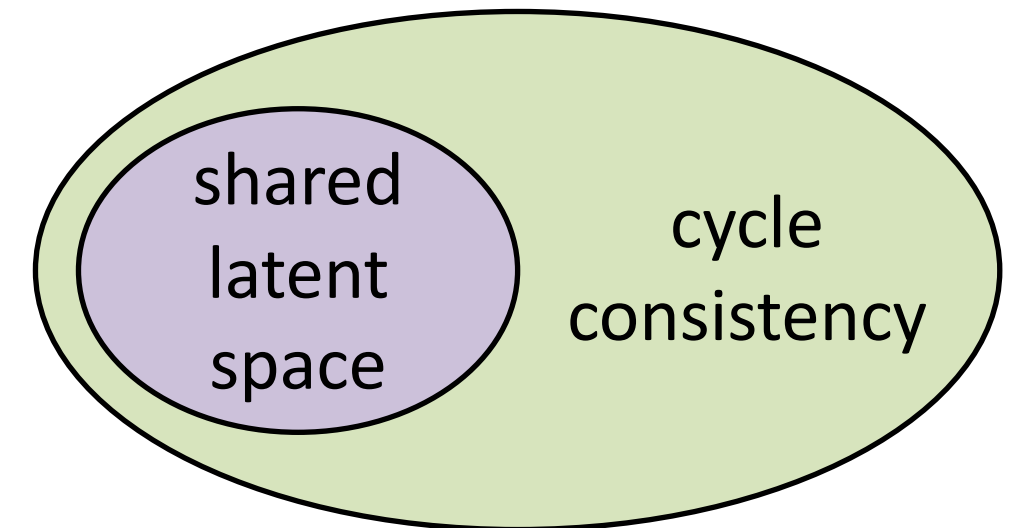
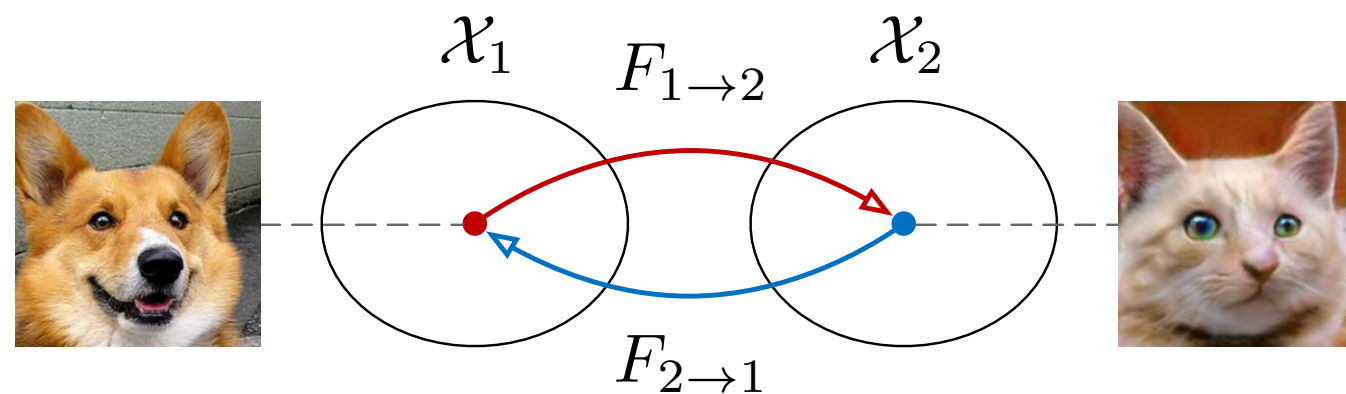
Feature loss (Preserve content in feature space) $_x$

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



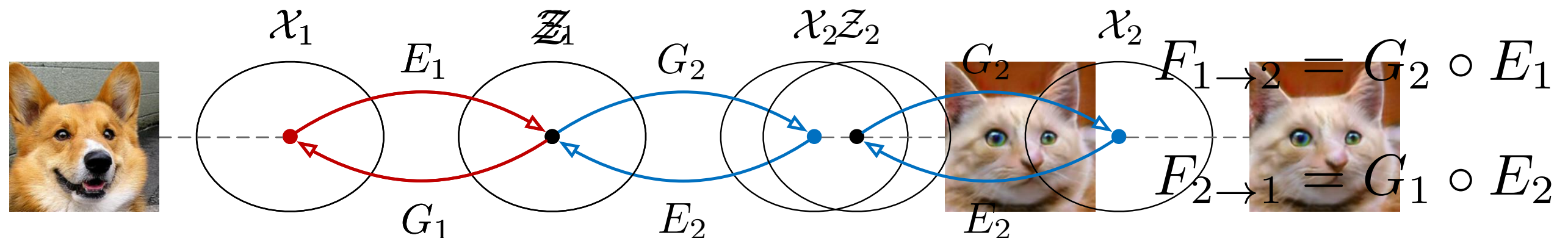
CycleGAN and UNIT

- CycleGAN (**cycle consistency**)



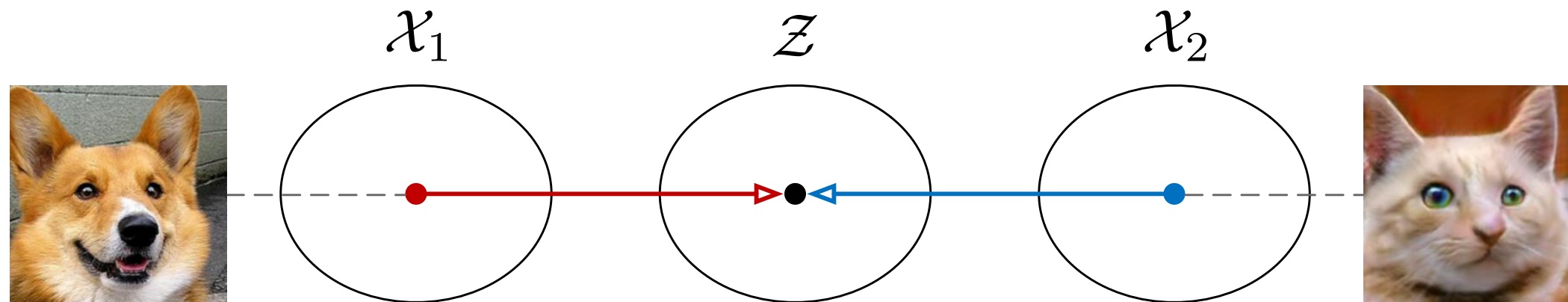
- UNIT (**shared latent space**) [Liu et al. 2017]

shared latent space $\Rightarrow$ cycle consistency



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**

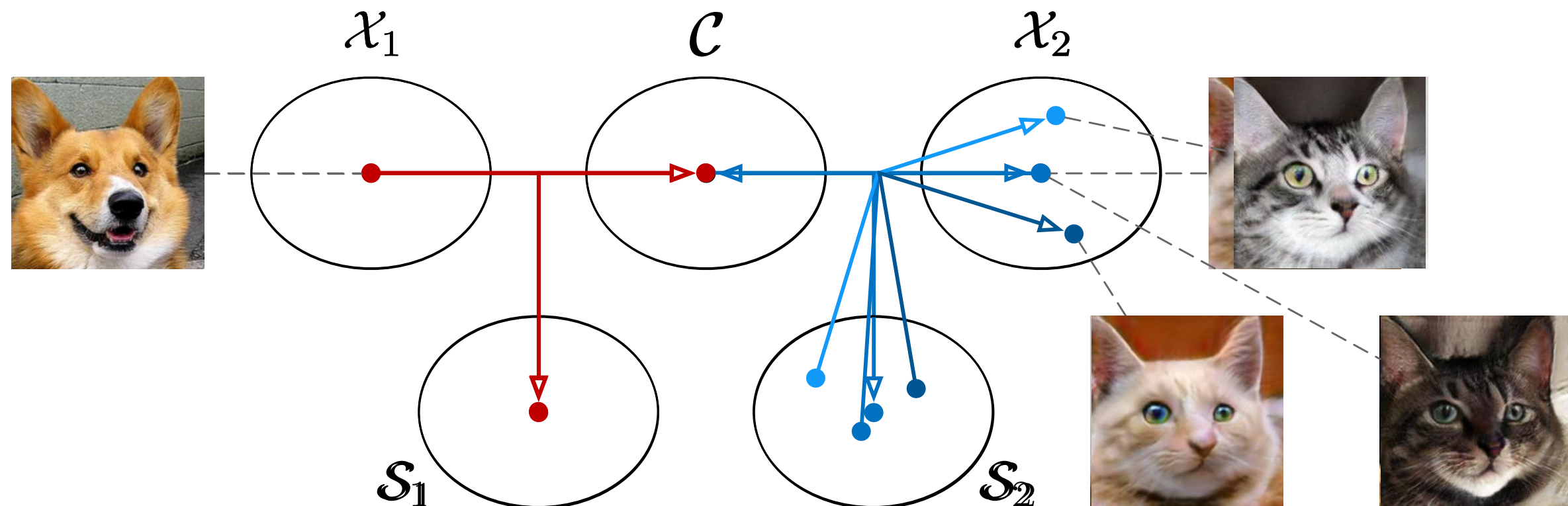
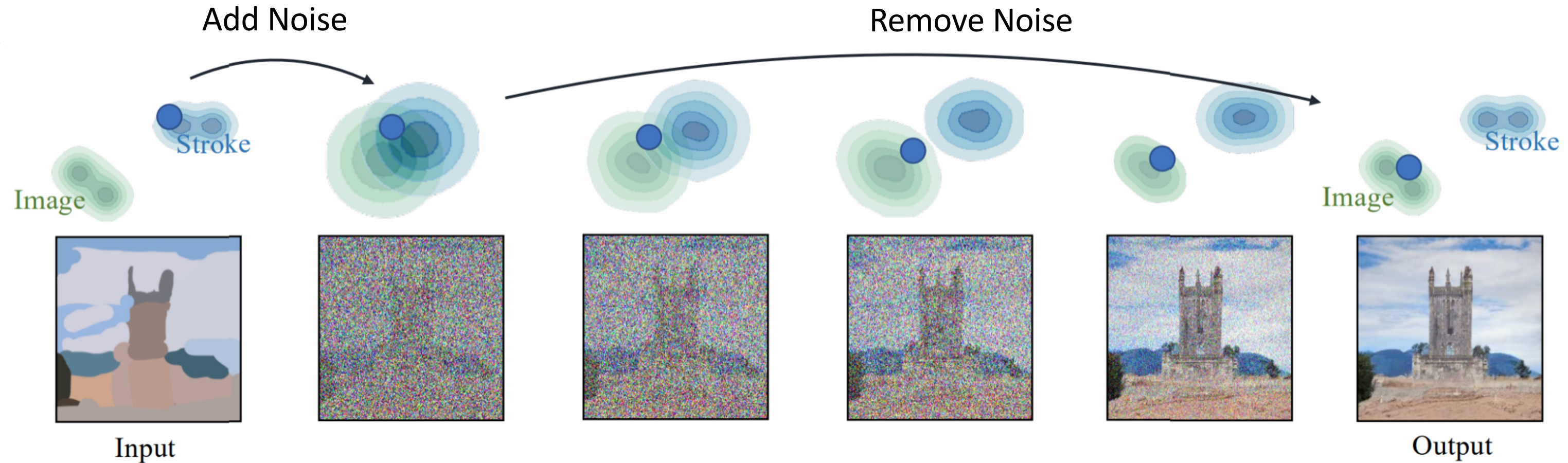


Image-to-Image Translation with Diffusion Models

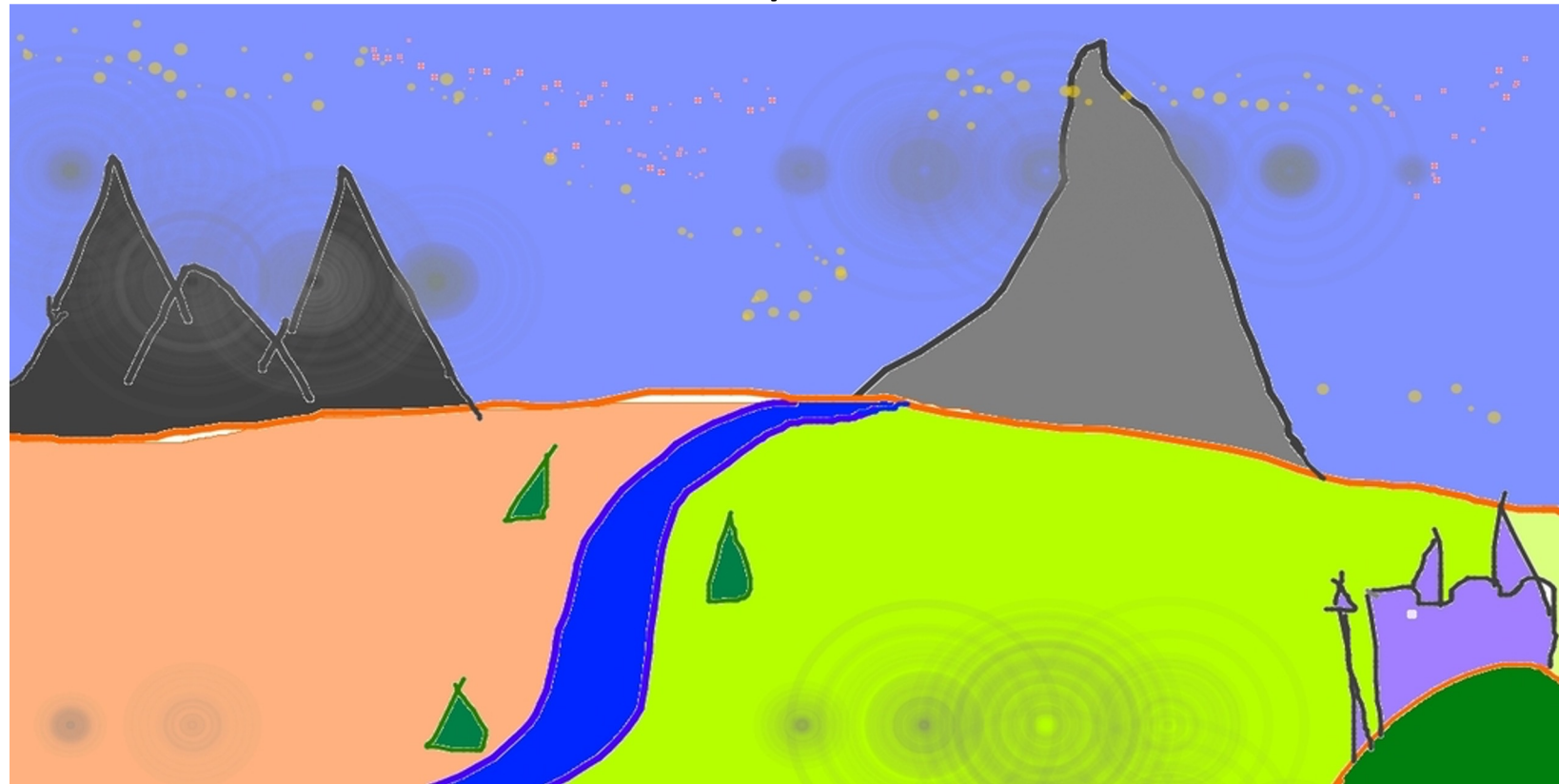
Guided Image Synthesis

SDEdit (<https://arxiv.org/abs/2108.01073>) recipe: diffuse $\rightarrow$ denoise



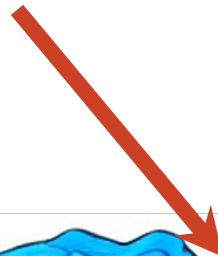
Guided Image Synthesis

input

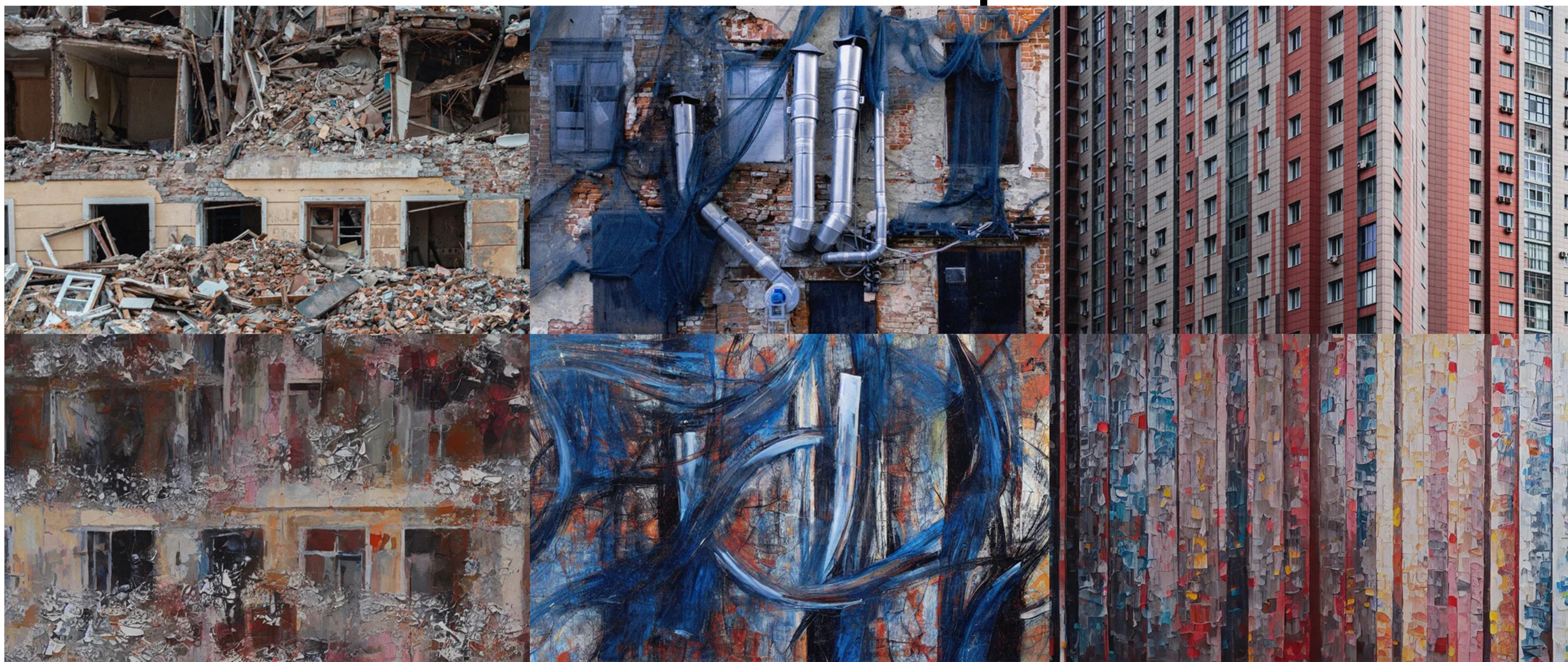




“Upgrade” your child’s artwork



abstract art from photos

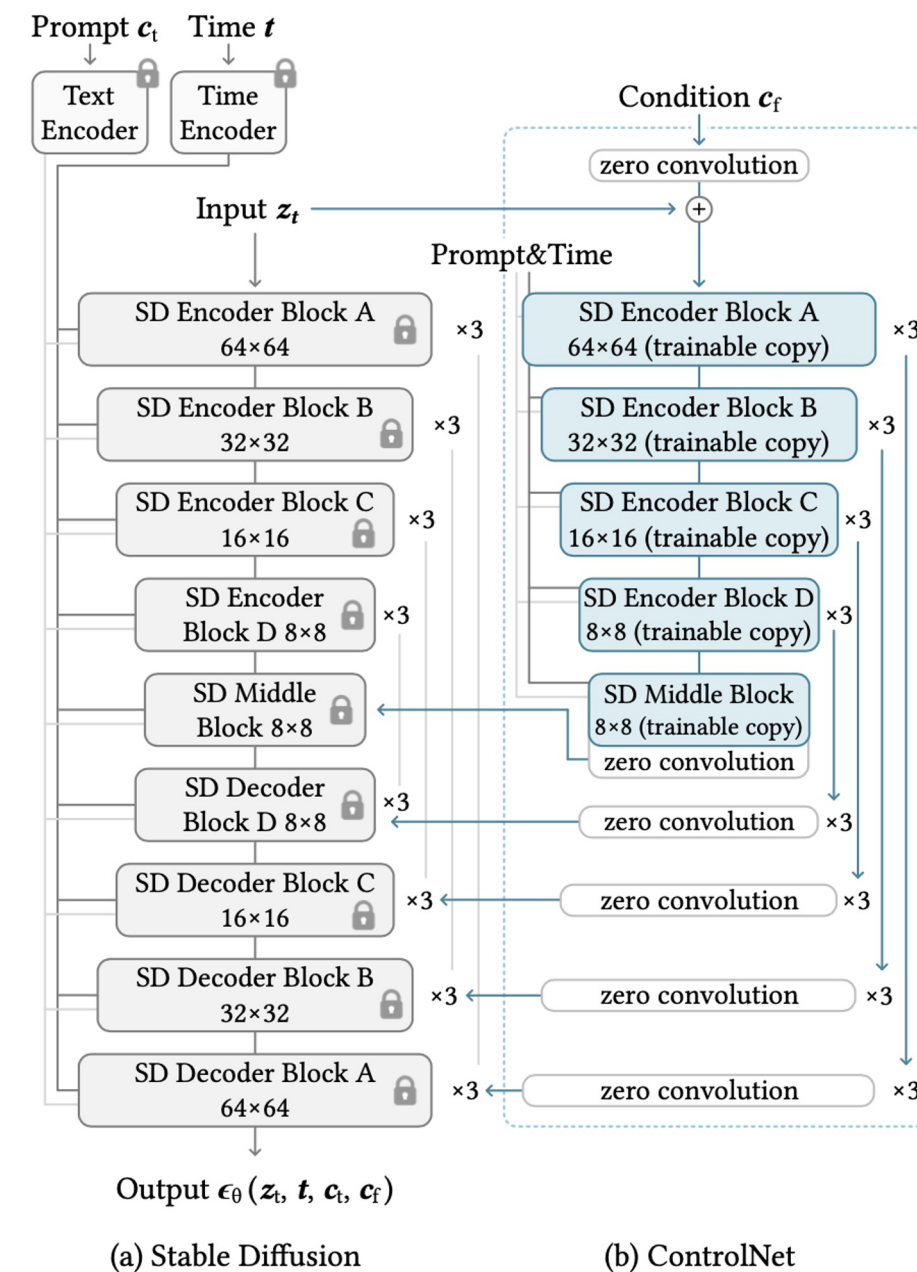
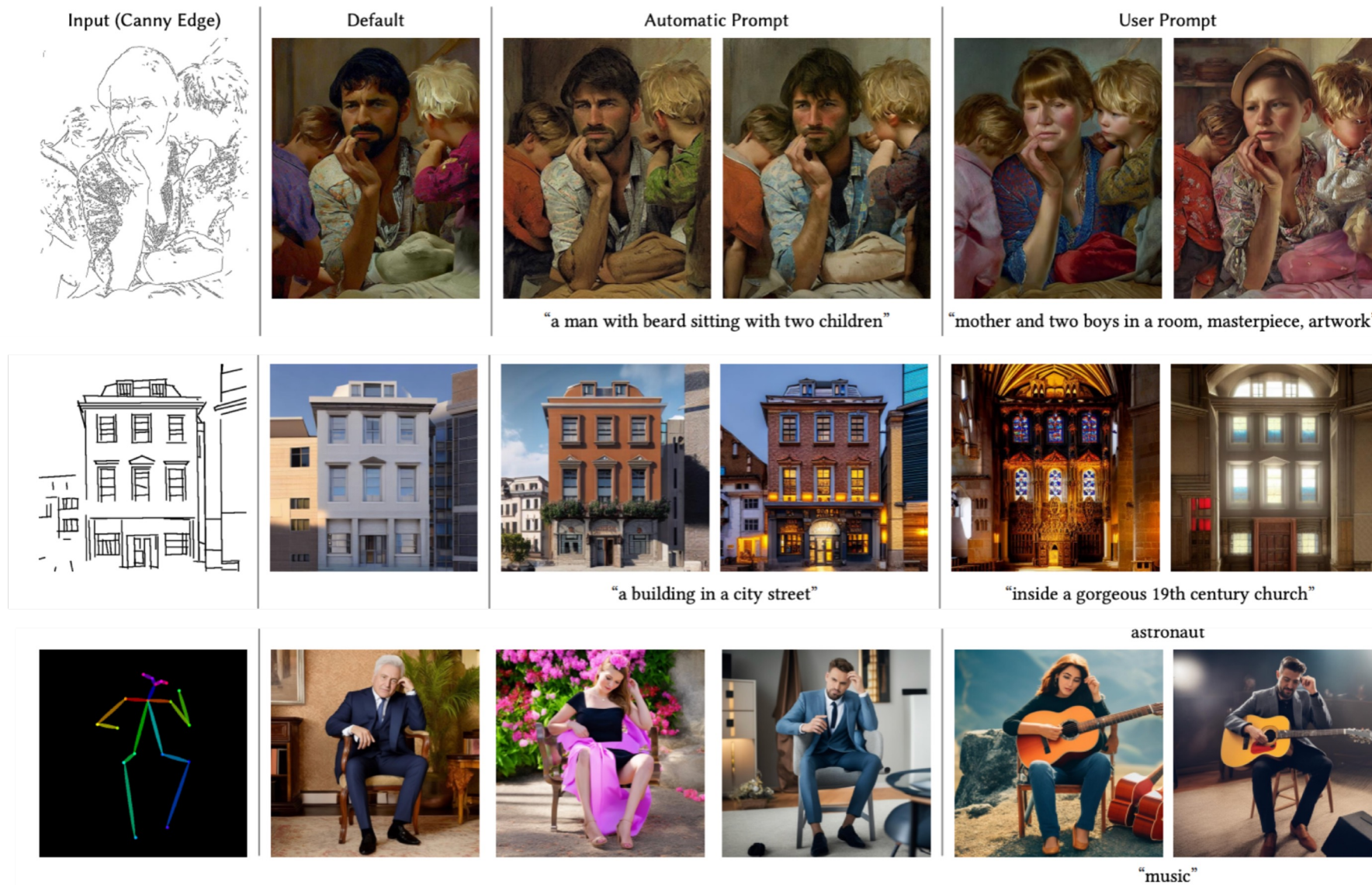


original post by [u/Pereulkov](https://www.reddit.com/user/Pereulkov)

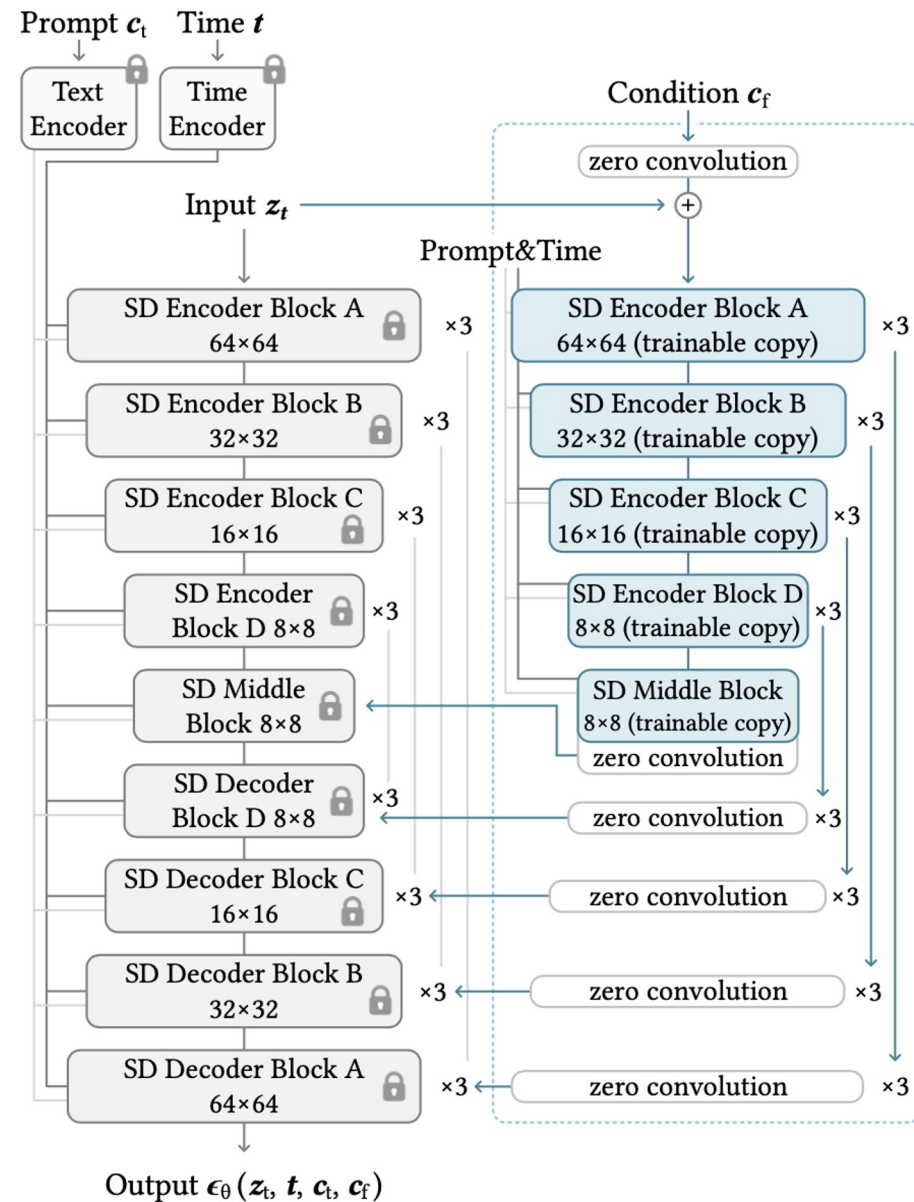
[https://www.reddit.com/r/StableDiffusion/comments/xhhyad/i made abstract art from my photos/](https://www.reddit.com/r/StableDiffusion/comments/xhhyad/i_made_abstract_art_from_my_photos/)

ControlNet: Adding Conditional Control to Text-to-Image Diffusion Models

ControlNet



Architecture of ControlNet

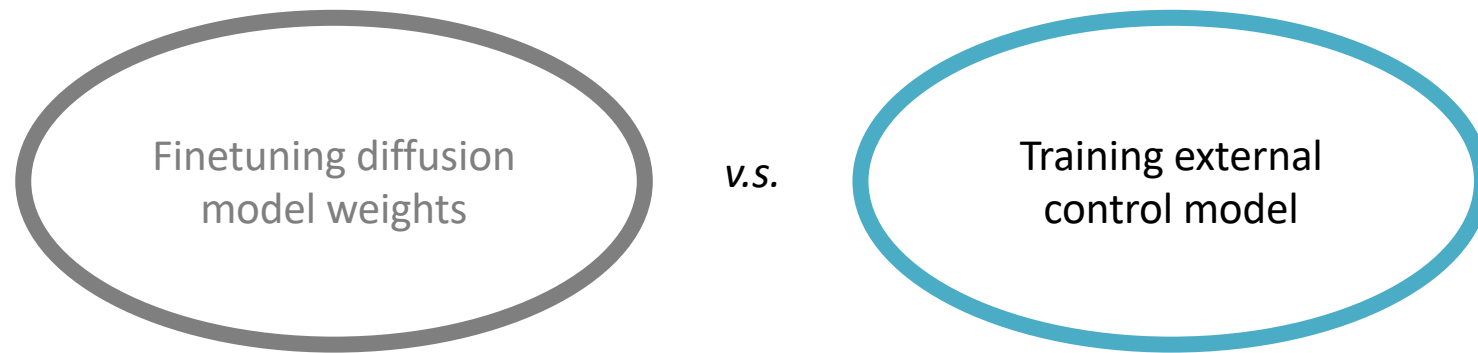


(a) Stable Diffusion

(b) ControlNet

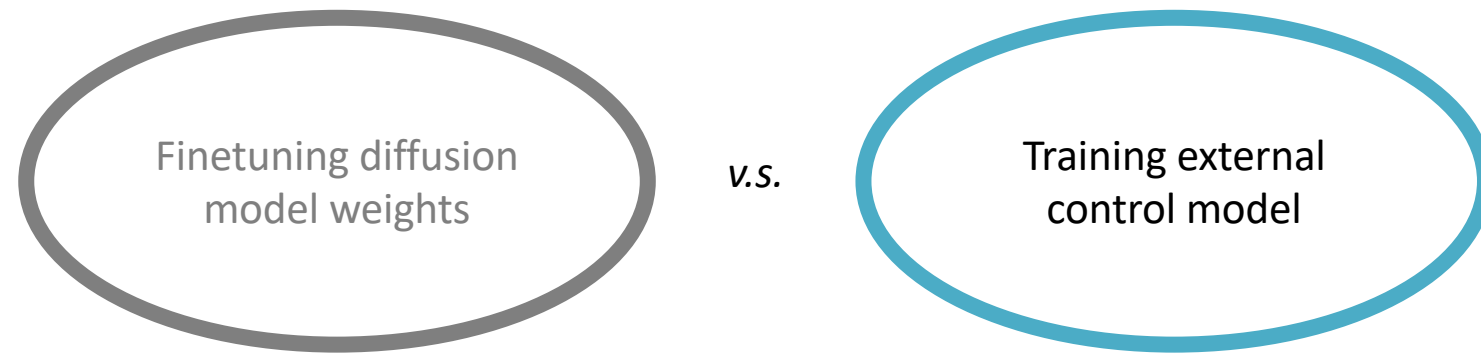
- Using external model to process control signals.
- Re-using pretrained weights as the backbone of control model.
- Connecting with zero-initialized layers to reduce initial noise.

Using external model to process control signals

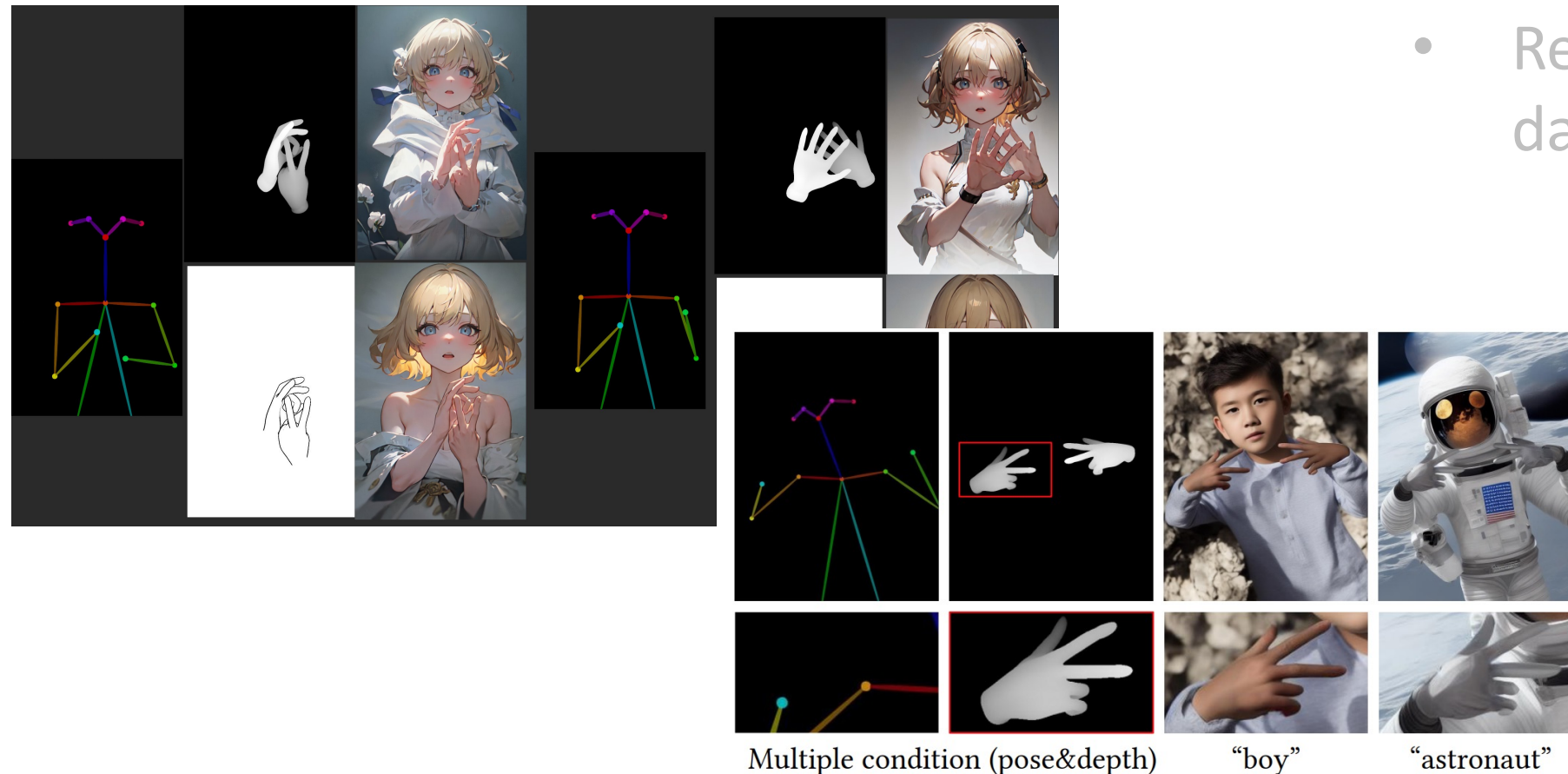


- Compossible control (multiple conditions)
- Minimal influence to the base model (the base model can be changed)
- Reduced overfitting risk (training with small dataset becomes easier)

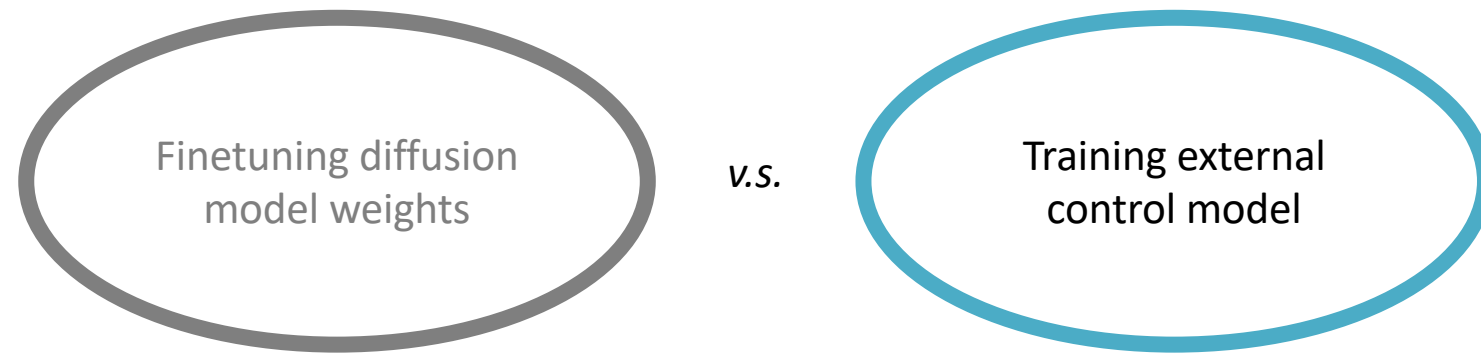
Using external model to process control signals



- Compossible control (multiple conditions)
- Minimal influence to the base model (the base model can be changed)
- Reduce overfitting (training with small dataset becomes easier)



Using external model to process control signals



- Compossible control (multiple conditions)
- Minimal influence to the base model (the base model can be changed)
- Reduce overfitting (training with small dataset becomes easier)



“house”



SD 1.5



Comic Diffusion



Protogen 3.4

Using external model to process control signals

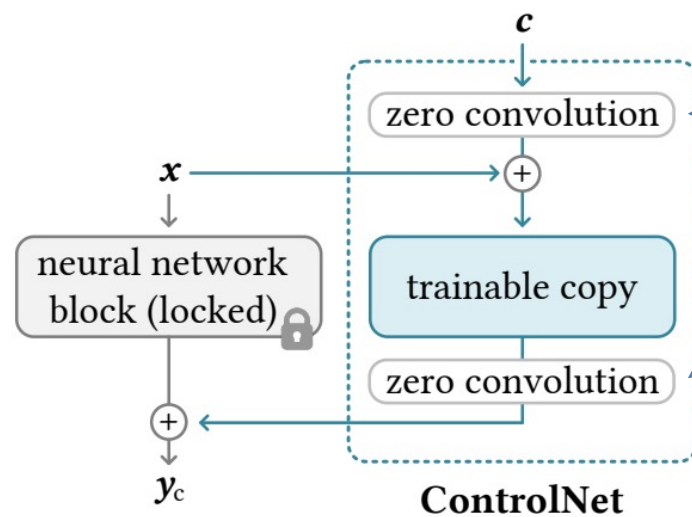


without ControlNet
(using Stability's "official" method to add
the channels to input layer, same as their
depth-to-image structure)

SD + ControlNet

- Compossible control (multiple conditions)
- Minimal influence to the base model (the base model can be changed)
- Reduce overfitting (training with small dataset becomes easier)

Using zero-initialized layers to reduce initial noise



Zero-initialized connection layers

- Reduce initial harmful noise
- Protect the trainable copy

Applications



Input Canny edge



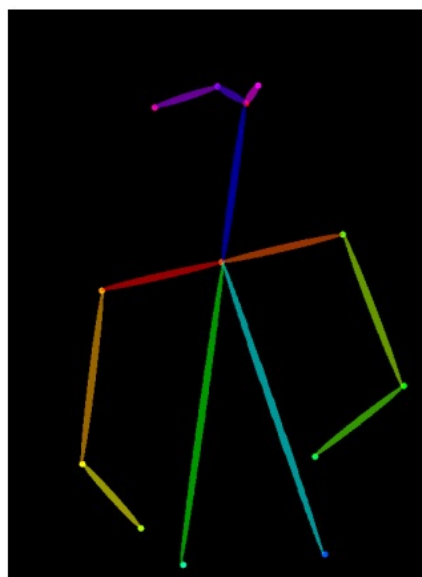
Default



“masterpiece of fairy tale, giant deer, golden antlers”



“..., quaint city Galic”



Input human pose



Default

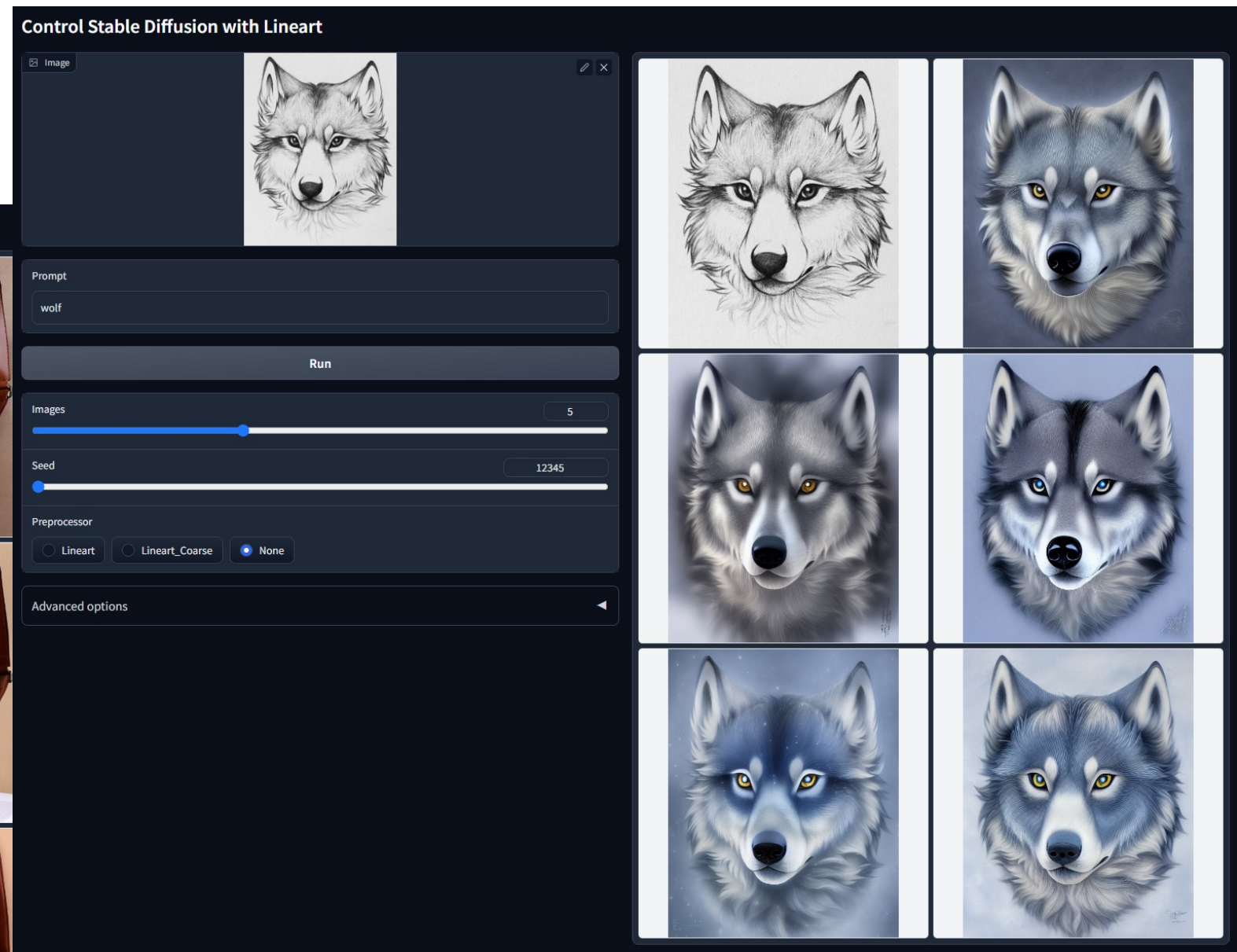
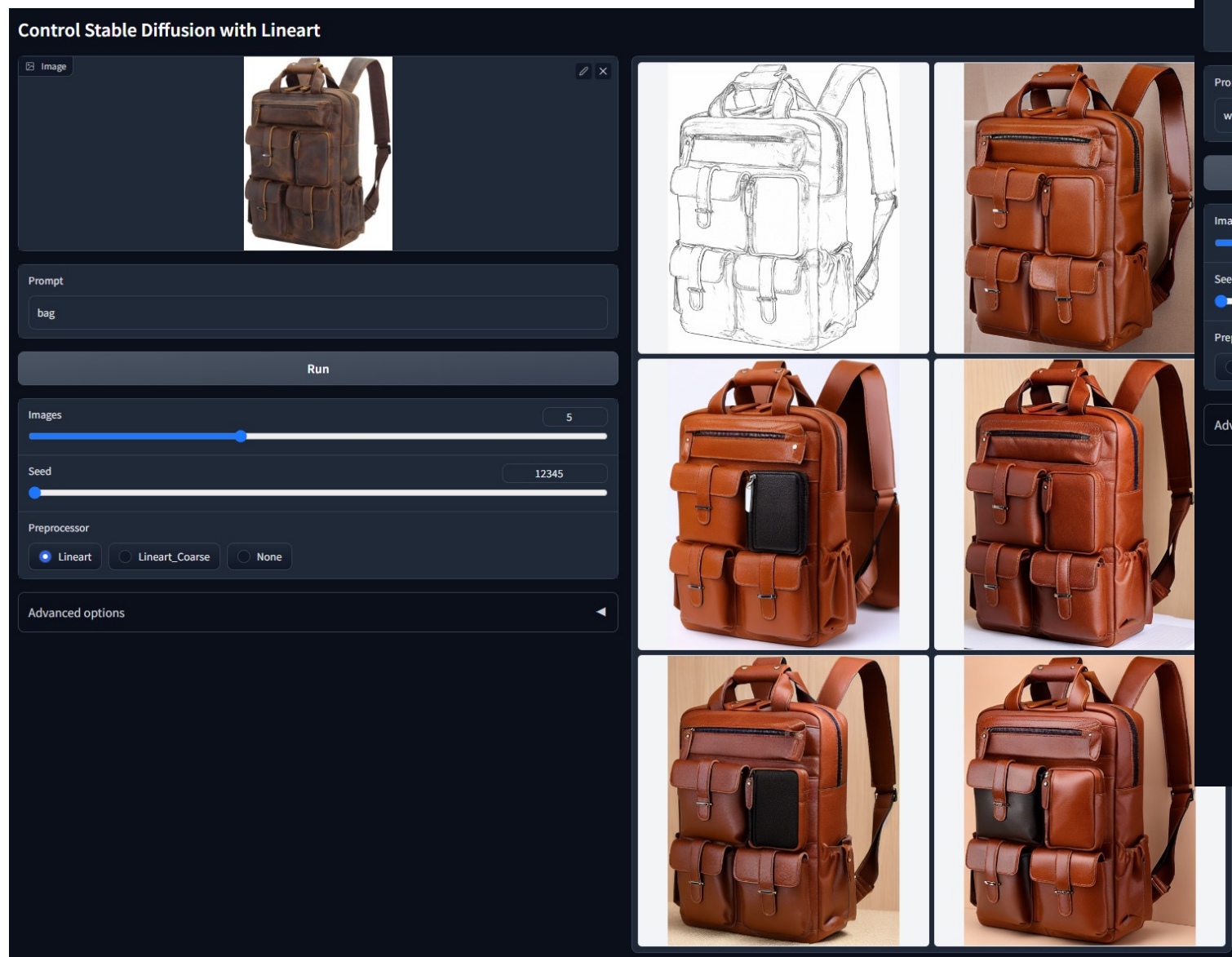


“chef in kitchen”

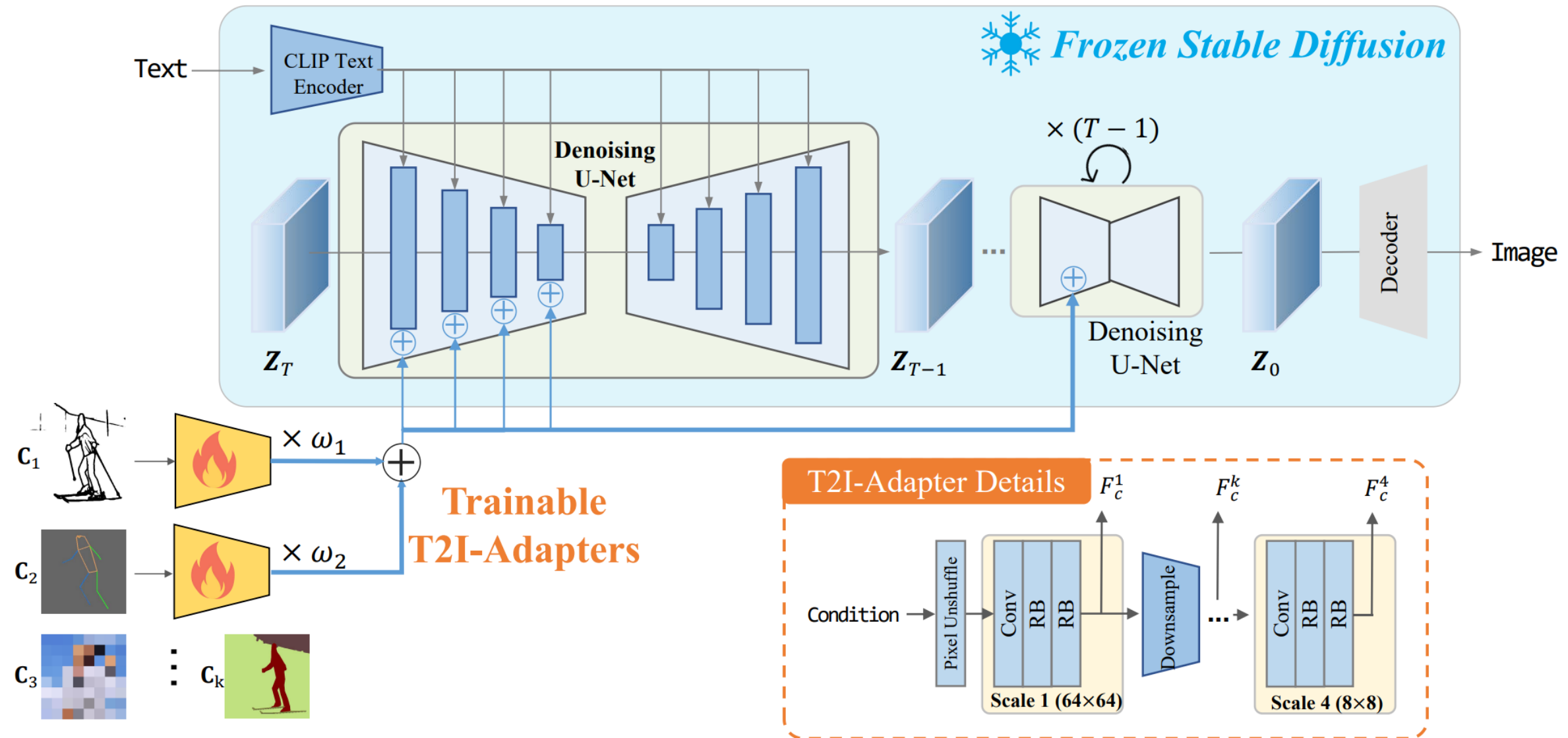


“Lincoln statue”

Applications



Conditional Diffusion Models



Conditional Diffusion Models

- **Lightweight**
 - The backbone model weight is frozen.
- **Composable**
 - use multiple controls (adapters) together (e.g., depth+pose)
- **Generalizable and reusable**
 - Can work with other backbone models



Style and Content, Texture Synthesis

Jun-Yan Zhu

16-726, Spring 2025

Texture

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



radishes



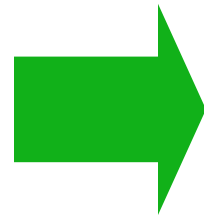
rocks



yogurt

Texture Synthesis

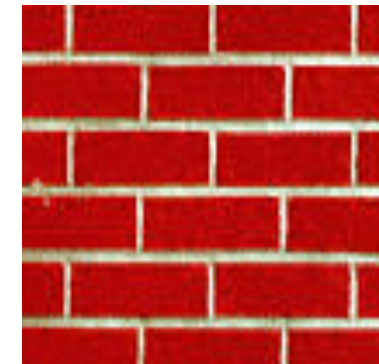
- Goal: create new samples of a given texture
- Applications: virtual environments, inpainting, texturing surfaces



Non-parametric Texture Synthesis

The Challenge

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



repeated

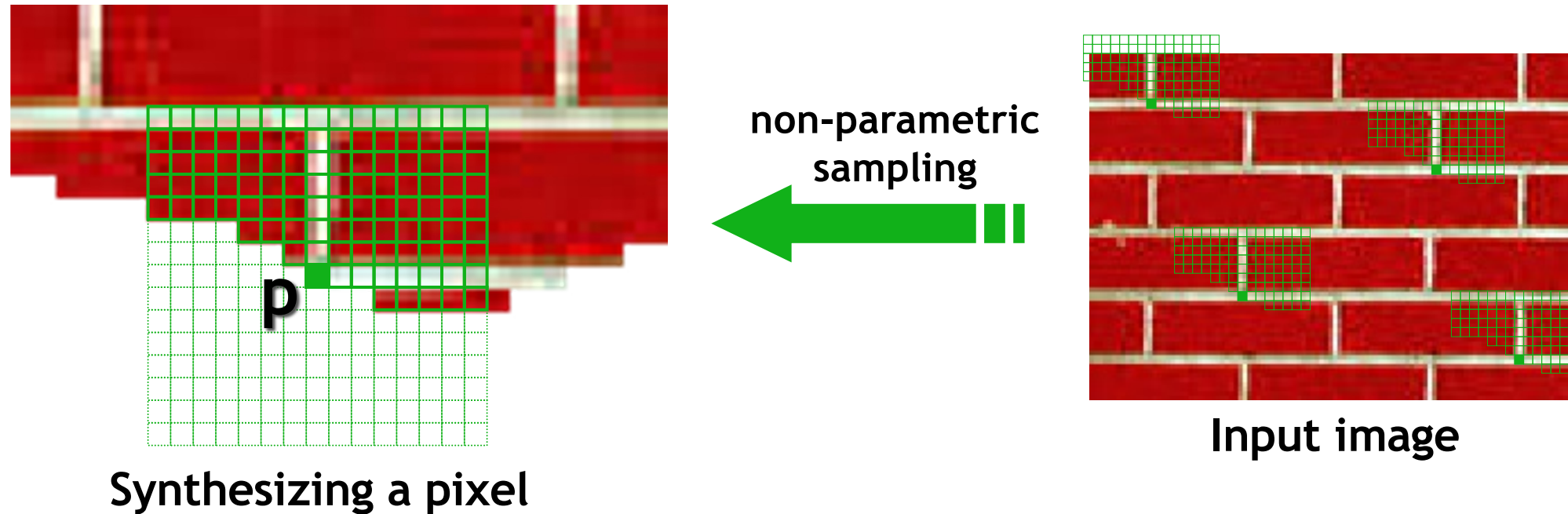


stochastic



Both?

Efros & Leung Algorithm



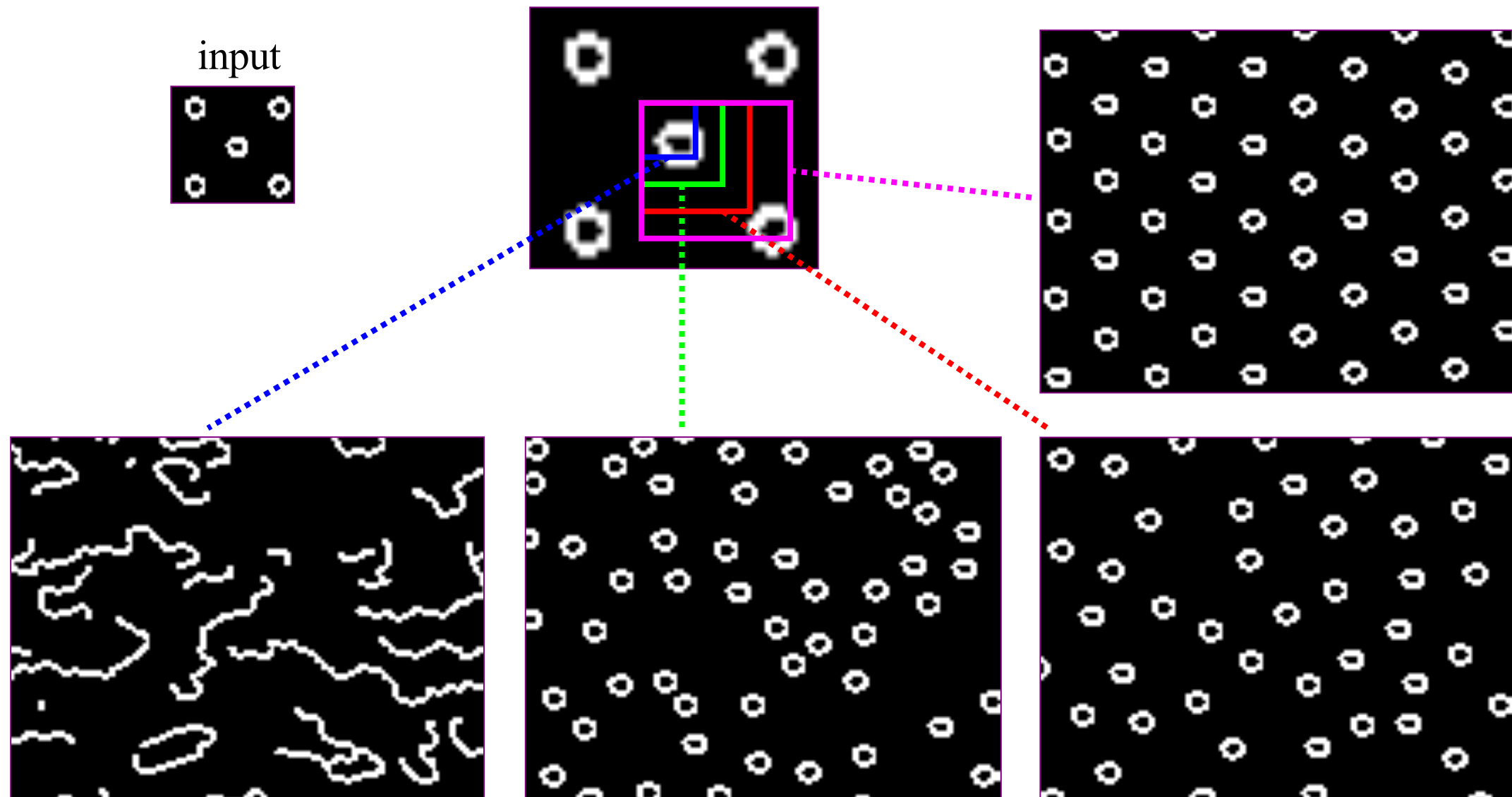
- Assuming Markov property, compute $P(\mathbf{p} | N(\mathbf{p}))$
 - Building explicit probability tables infeasible
 - Instead, we *search the input image* for all similar neighbourhoods — that's our pdf for $\mathbf{p}$
 - To sample from this pdf, just pick one match at random

PixelCNN

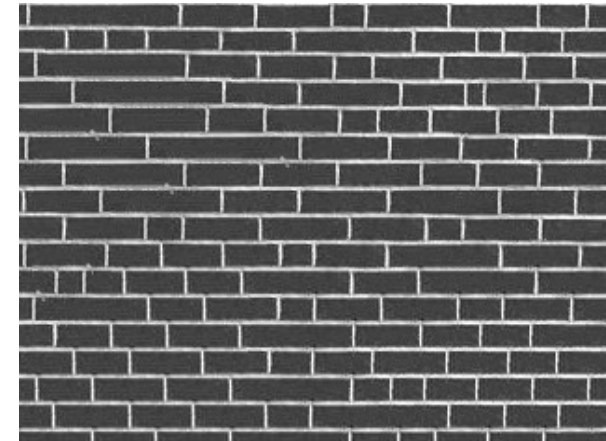
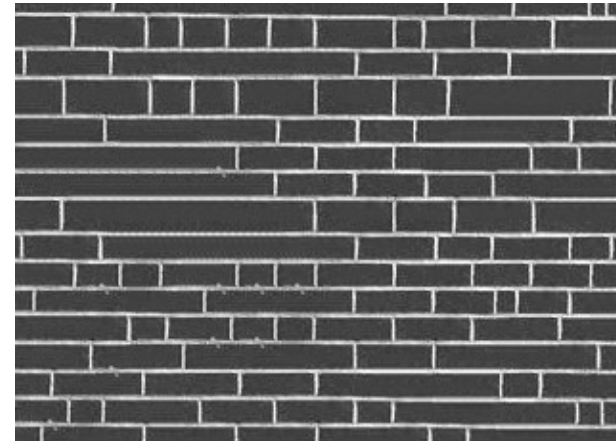
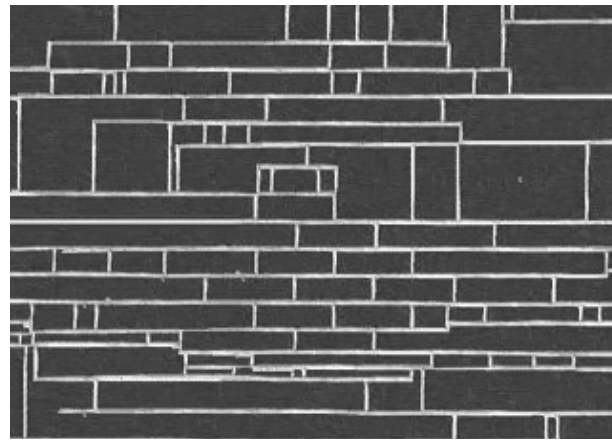
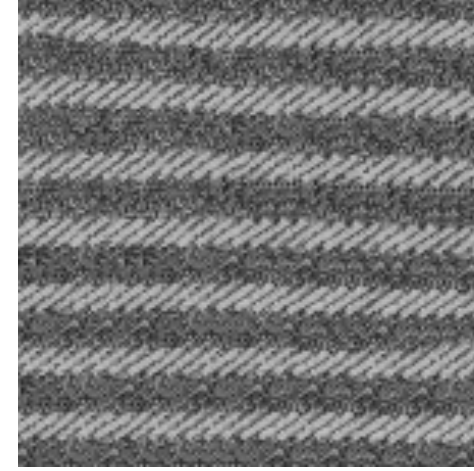
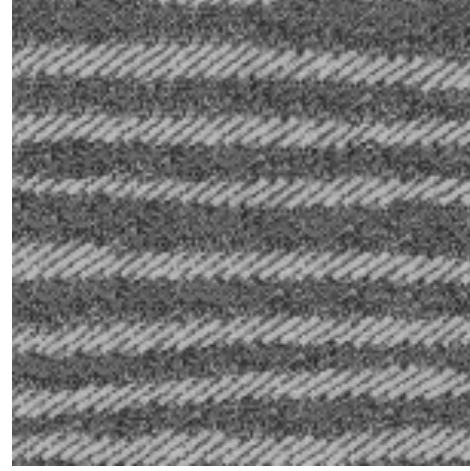
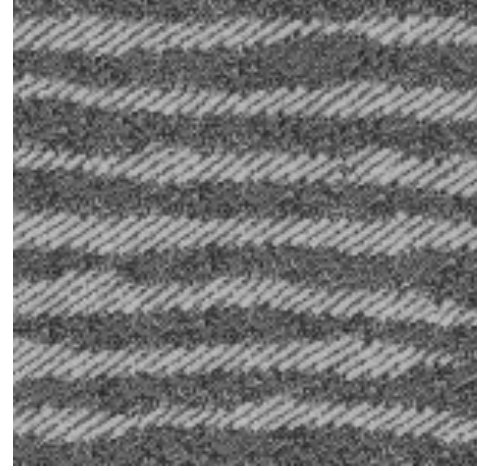
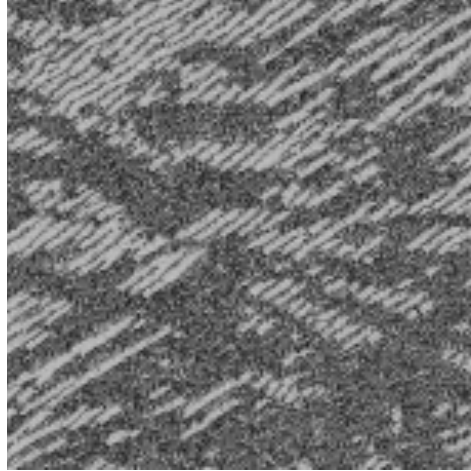
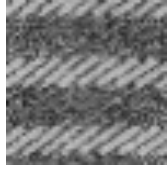
Some Details

- Growing is in “onion skin” order
 - Within each “layer”, pixels with most neighbors 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 closest neighbors
 - Approximates reduction to a smaller neighborhood window if data is too sparse

Neighborhood Window



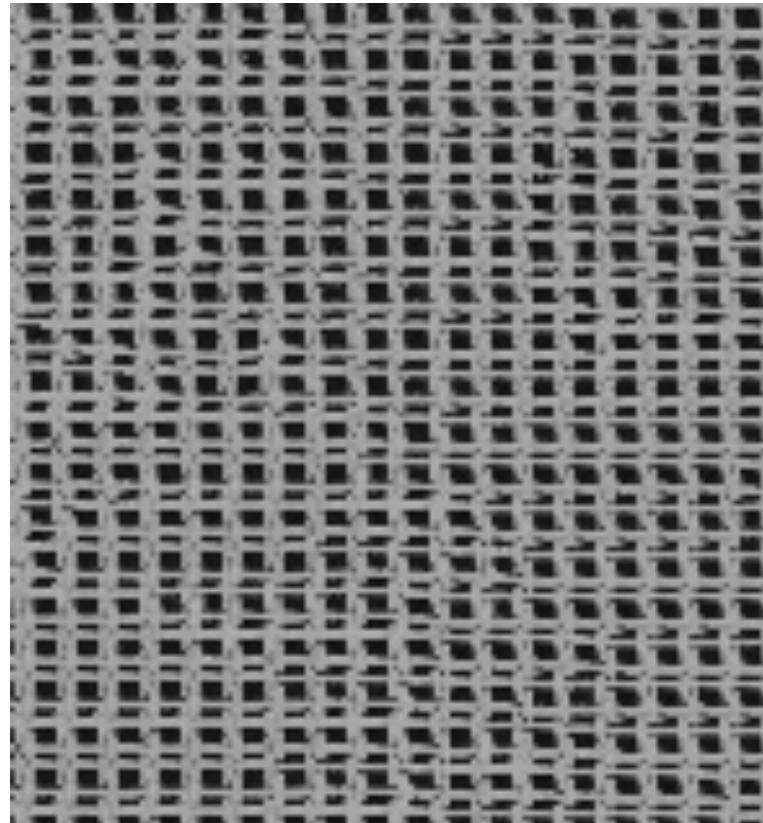
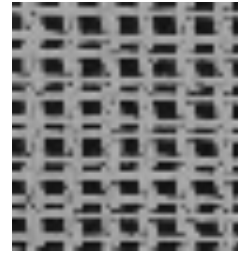
Varying Window Size



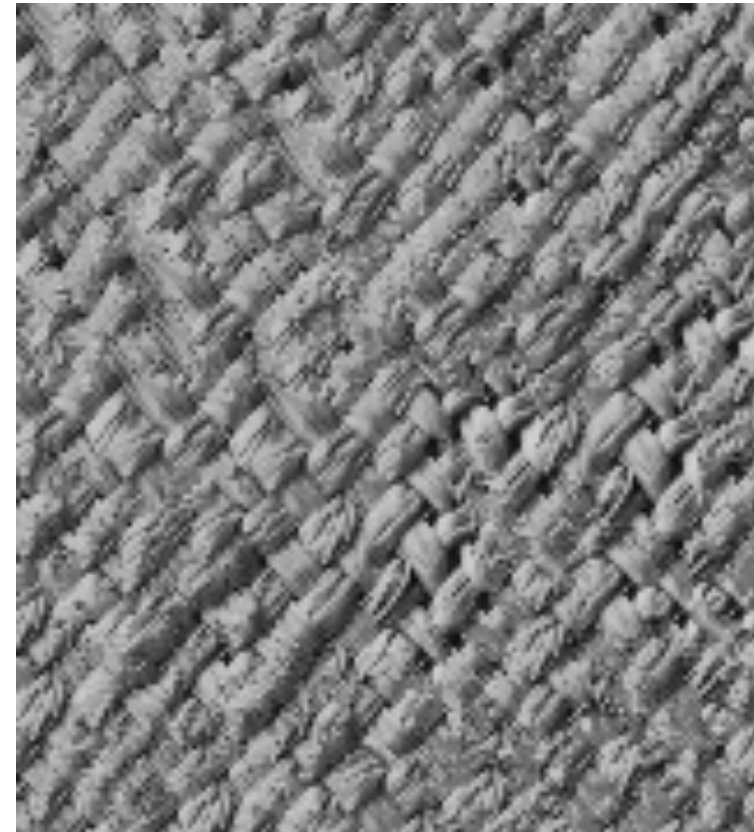
Increasing window size 

Synthesis Results

french canvas

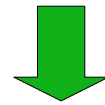


rafia weave

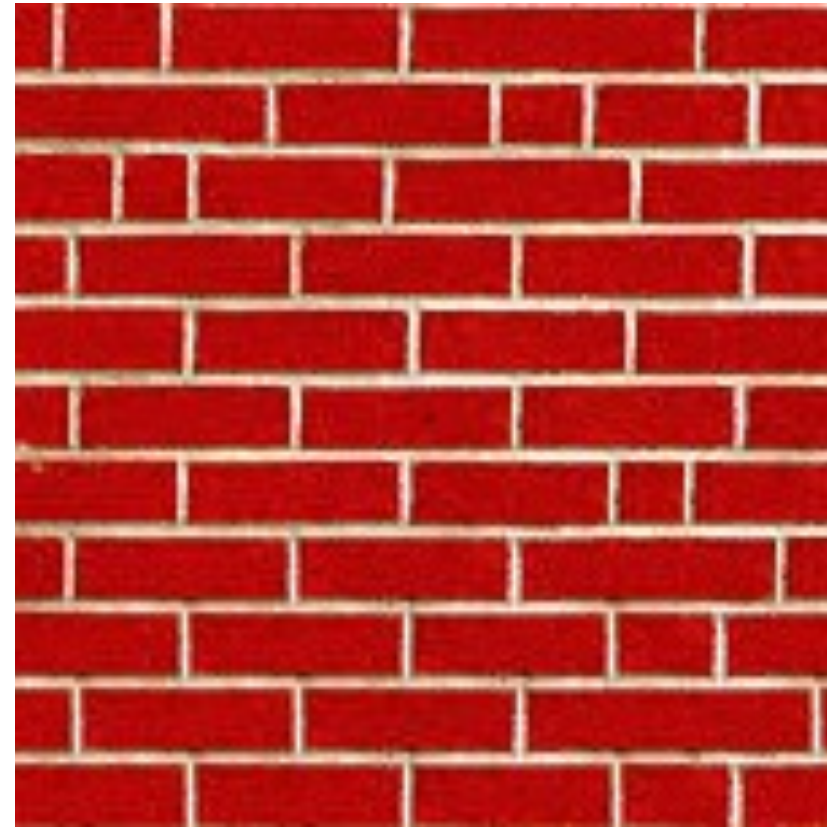
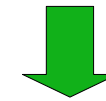


More Results

white bread

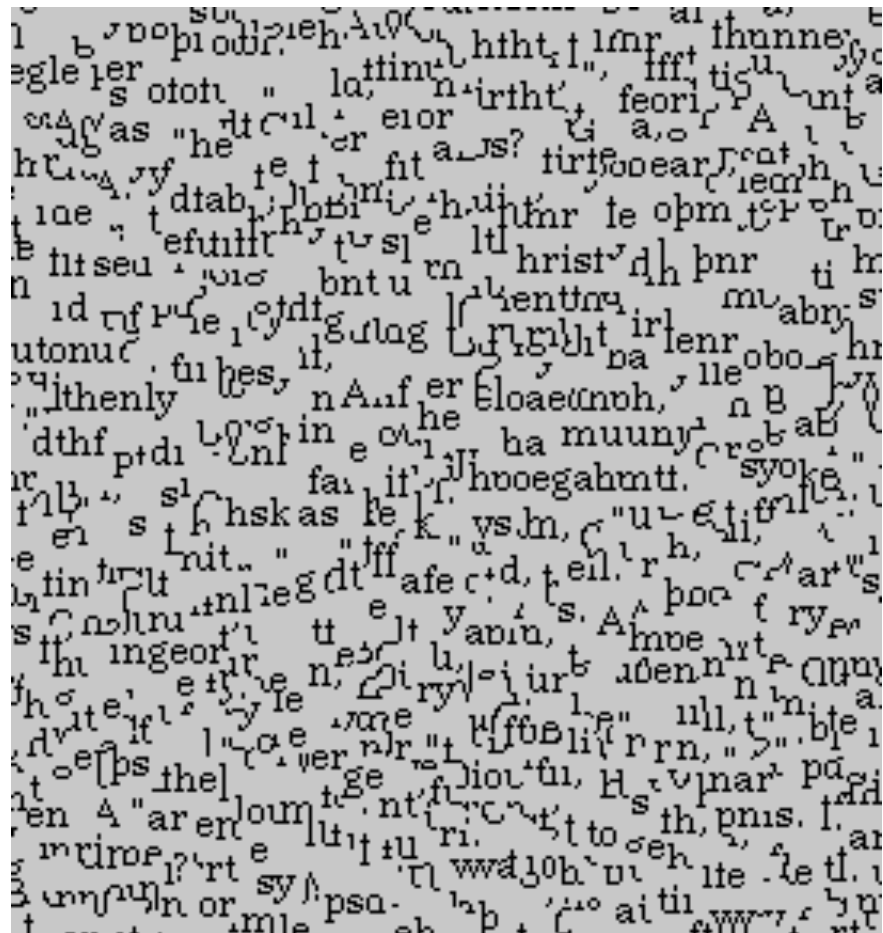


brick wall



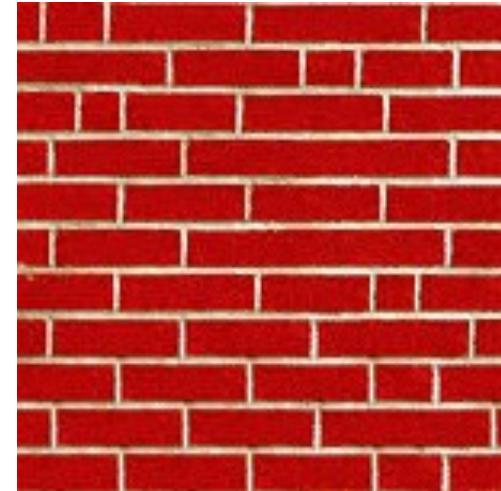
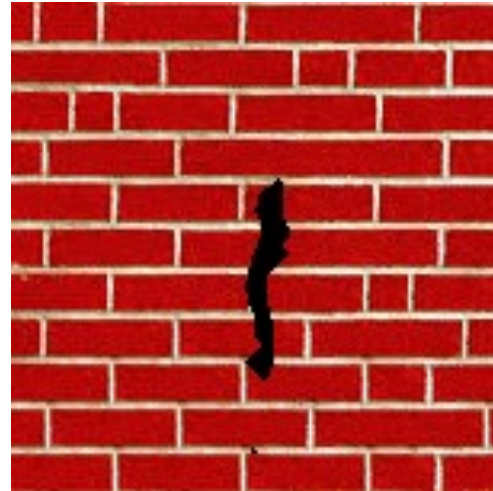
Homage to Shannon

...ing in the unsensational
r Dick Gephardt was fair
rful riff on the looming
nly asked, "What's your
tions?" A heartfelt sigh
story about the emergen
es against Clinton. "Boy
g people about continuin
ardt began, patiently obs
s, that the legal system h
g with this latest tanger

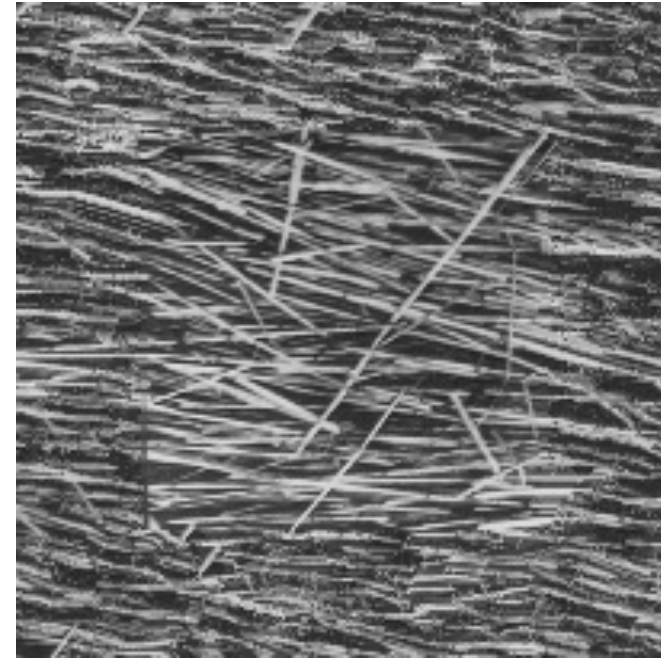
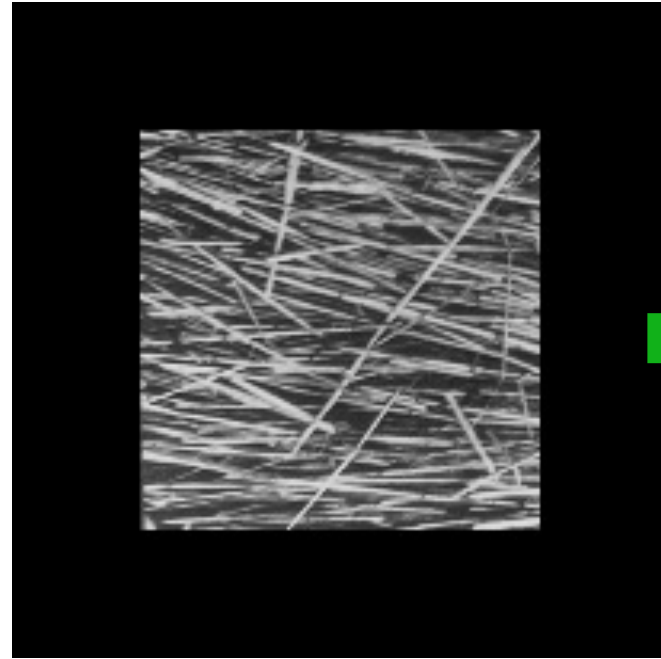


athaim, them. "Whnephartfe lartifelintomimen
el ck Clirtioout omaim thartfelins.f aut s anento
the ry onst wartfe lck Gephtoomimeationl sigab
Chiooufit Clinut Cll riff on. hat's yodn, parut tly
ons ycontonsteht wasked, paim t sahe loo riff on l
nskoneploourtfeas leil A nst Clit, "Wleontongal s
k Cirtioouirtfepe ong pme abegal fartfenstemem
tiensteneltorydt telemephminsverdt was agemer
ff ons artientont Cling peme asurte atich, "Boui s
nal s fartfelt sig pedrtltdt ske abounutie aboutioo
stfaonewas you aboronthardt thatins fain, ped, '
ains, them, pabout wasy arfint couitly d, l n A h
ole emthringbooreme agas fa bontinsyst Clinut
ory about continst Clipeopinst Cloke agatiff out C
stome zinemen tly ardt beoraboul n, thenly as t C
cons faimeme Diontont wat coutlyohgans as fan
ien, phrtfaul, "Wbaut cout congagal comininga
mifmst Clihy abon al coountha.emungairt tfoun
The loocrystan loontieph, intly on, theoplegatick t
ul tatieeontly atie Diontiomt wal s f tbegae ener
nthahsat's enenhhmas fan. "intchthorv ahons v

Hole Filling



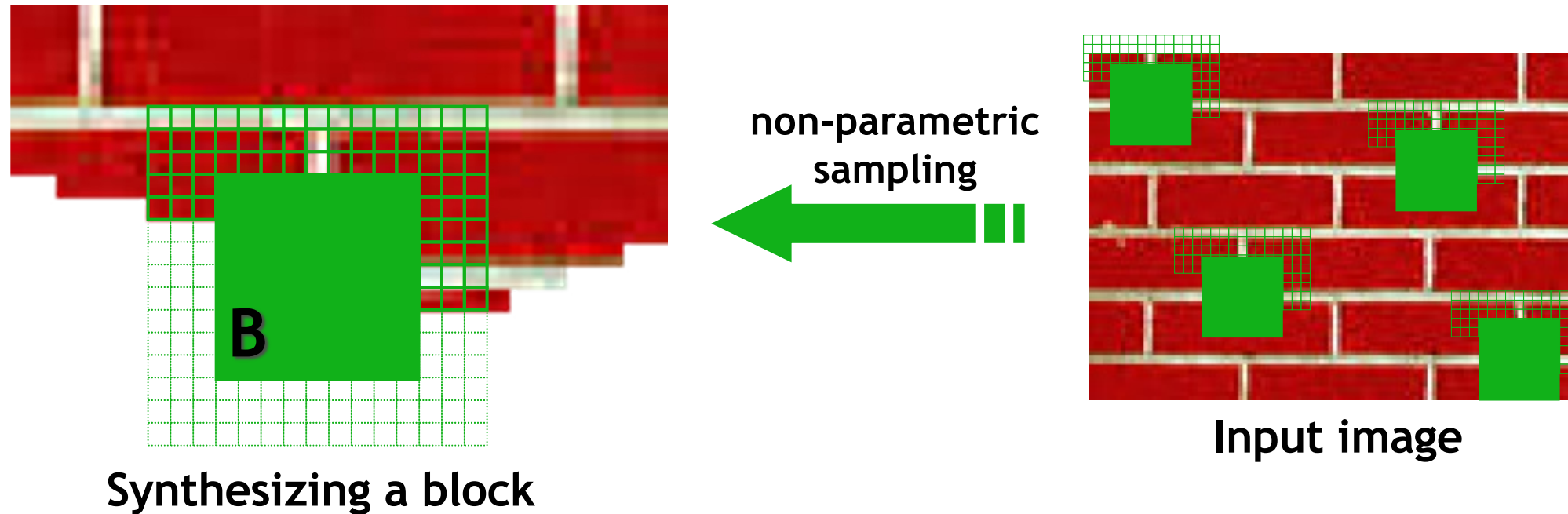
Extrapolation



Summary

- The Efros & Leung algorithm
 - + Very simple
 - + Surprisingly good results
 - + Synthesis is easier than analysis!
 - ...but very slow

Image Quilting [Efros & Freeman]

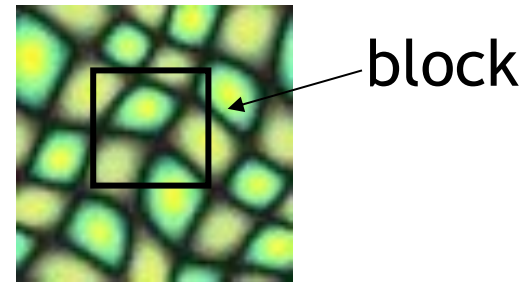


- Observation: neighbor pixels are highly correlated

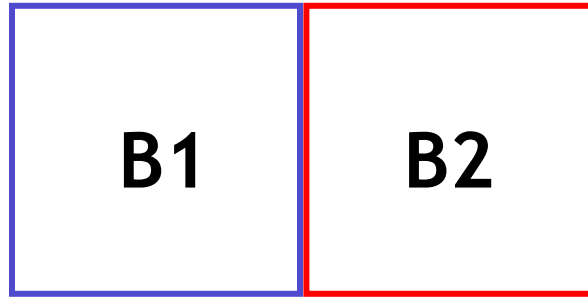
Idea: unit of synthesis = block

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

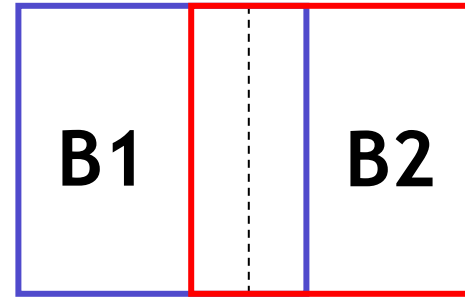
VQ-VAE2



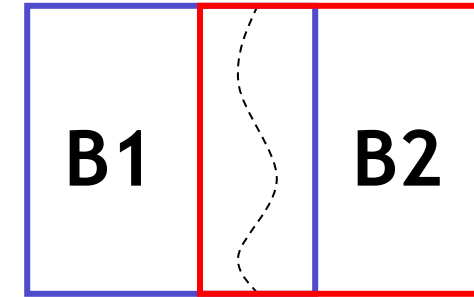
Input texture



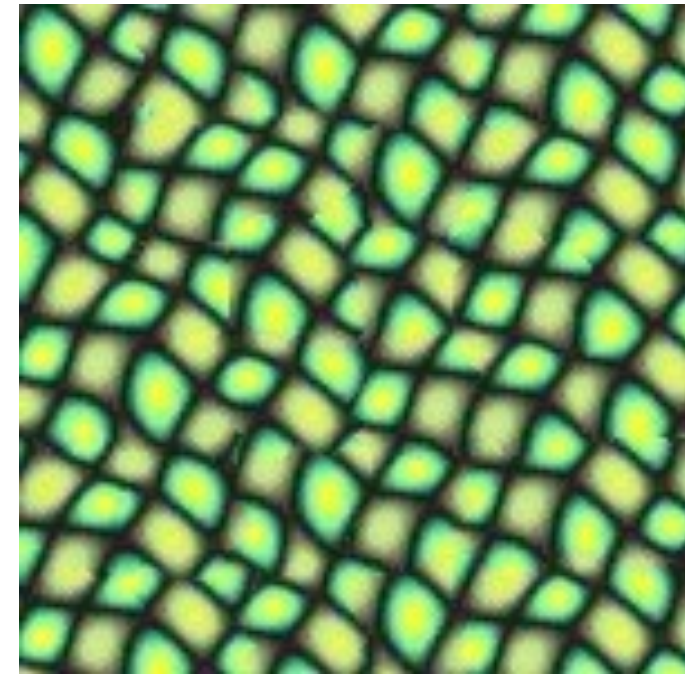
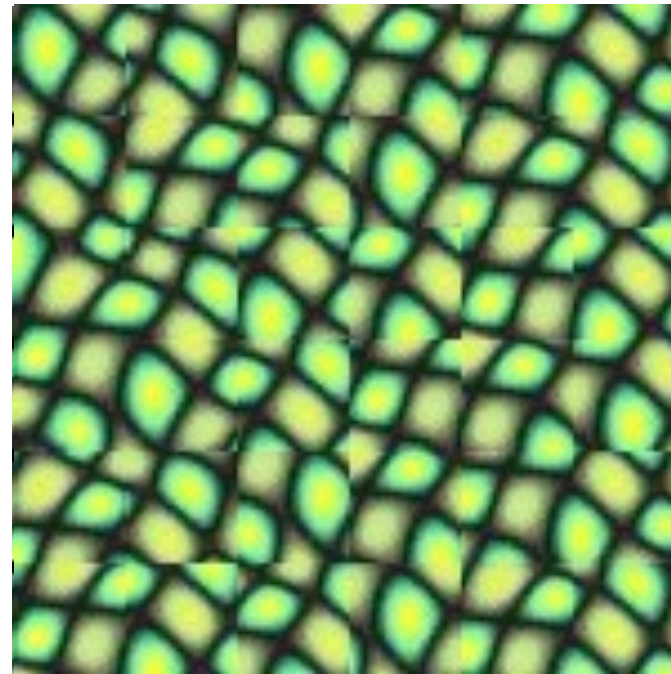
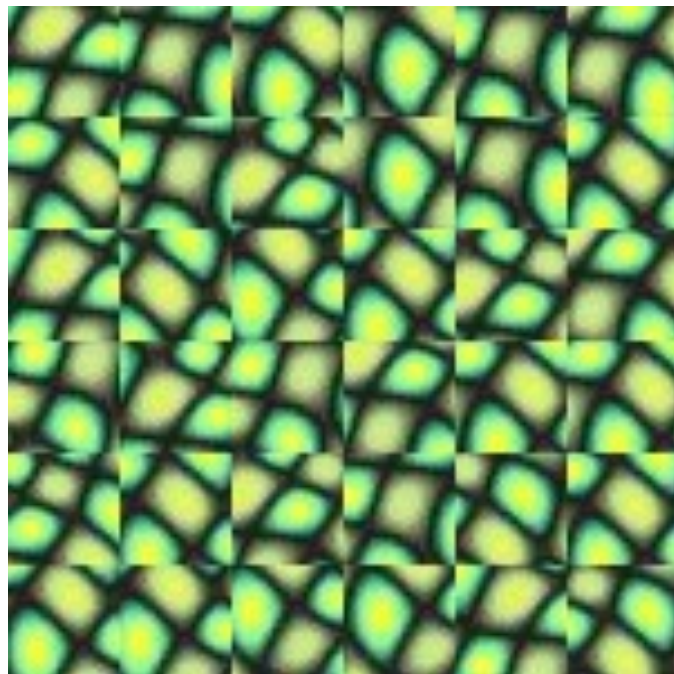
Random placement
of blocks



Neighboring blocks
constrained by overlap

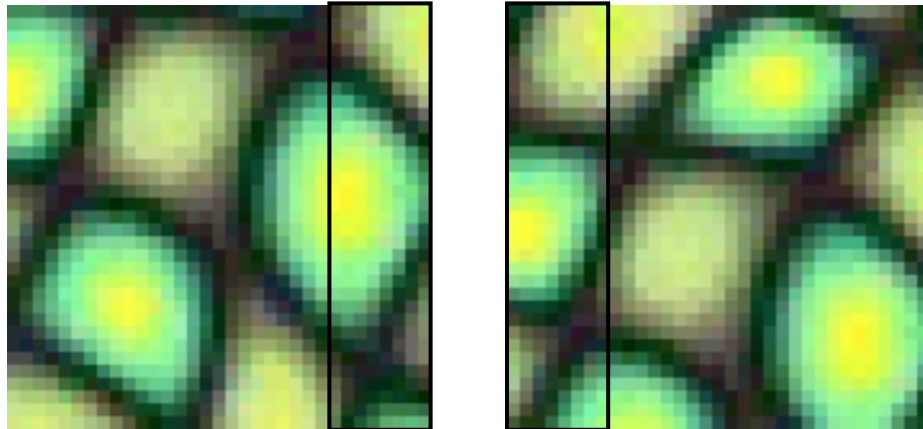


Minimal error
boundary cut

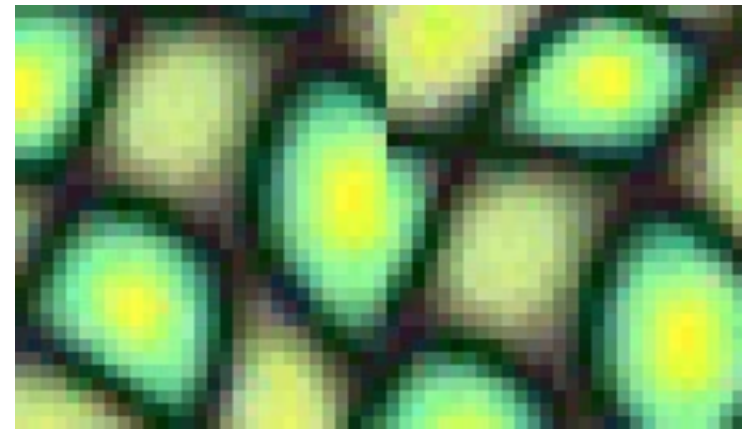


Minimal error boundary

overlapping blocks

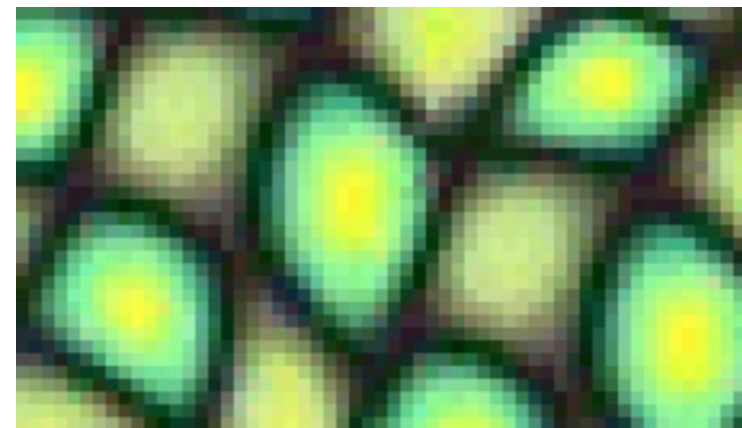


vertical boundary



A diagram illustrating the calculation of overlap error. It shows two vertical blocks of the heatmap image, each with a blue arrow pointing to it from above. These blocks are enclosed in large square brackets. Between the blocks is a minus sign. To the right of the brackets is a superscript '2'. This is followed by an equals sign and a vertical strip of the heatmap image. A red line is drawn along the boundary of this strip, representing the error.

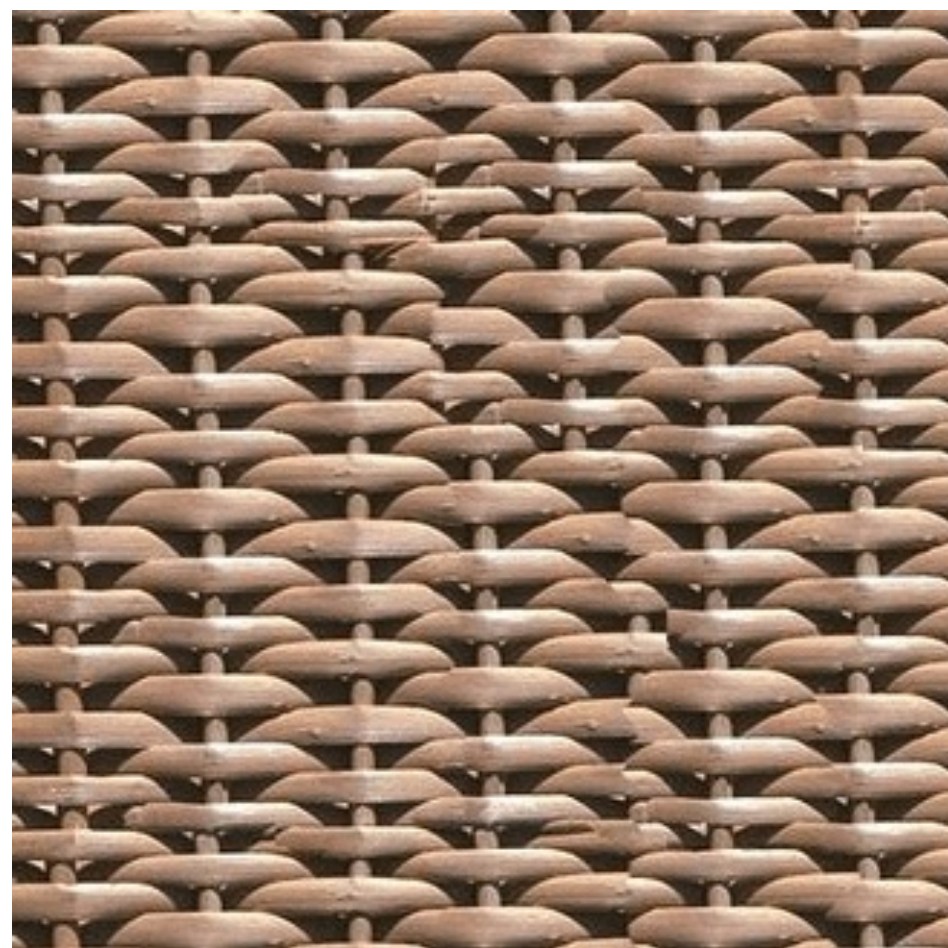
overlap error

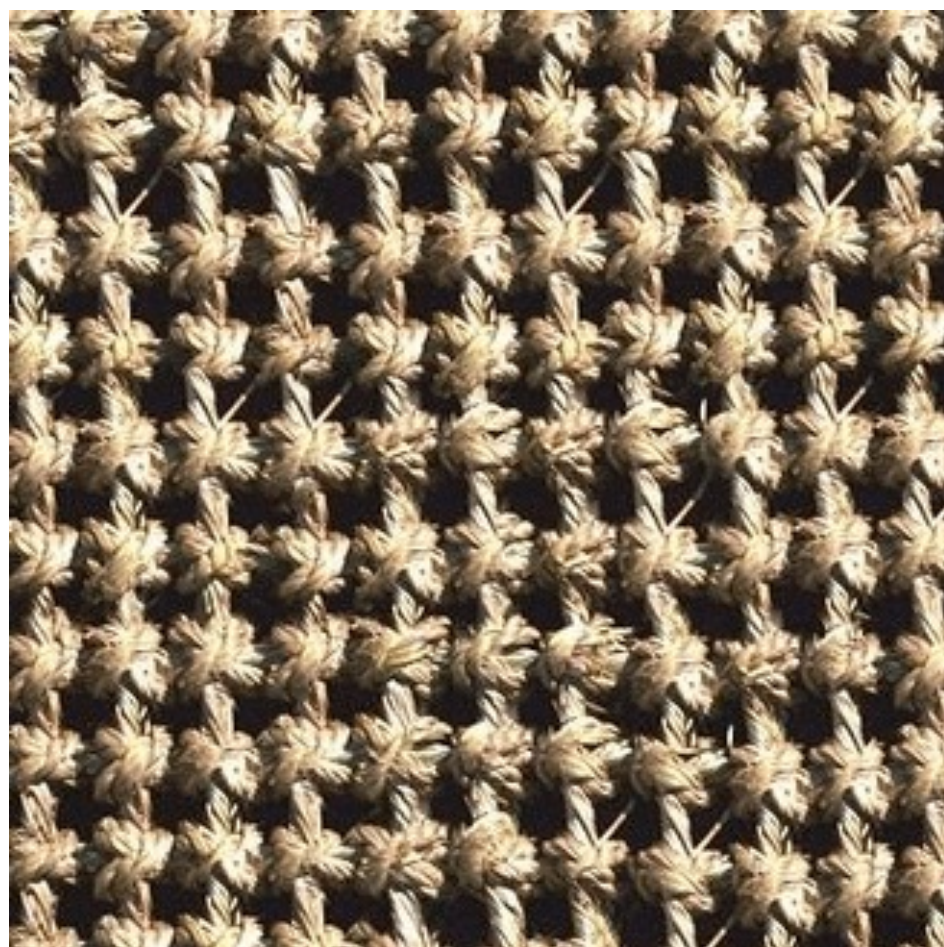


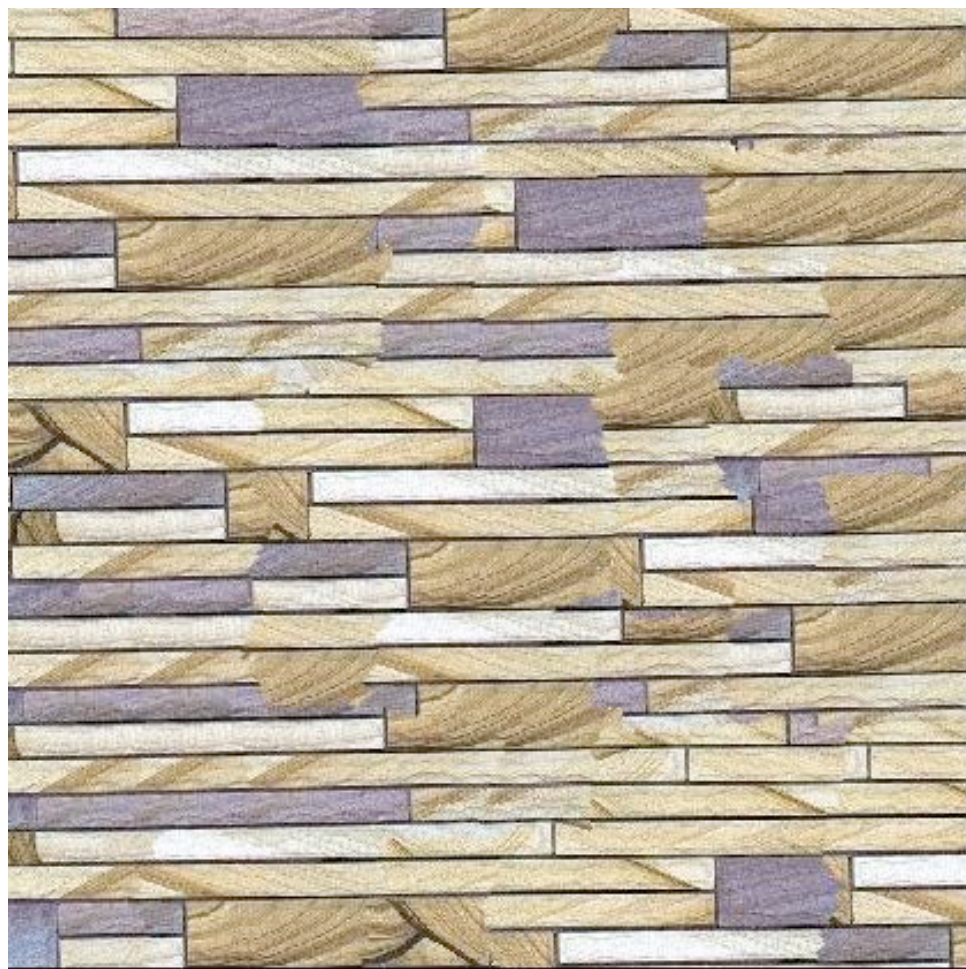
min. error boundary

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

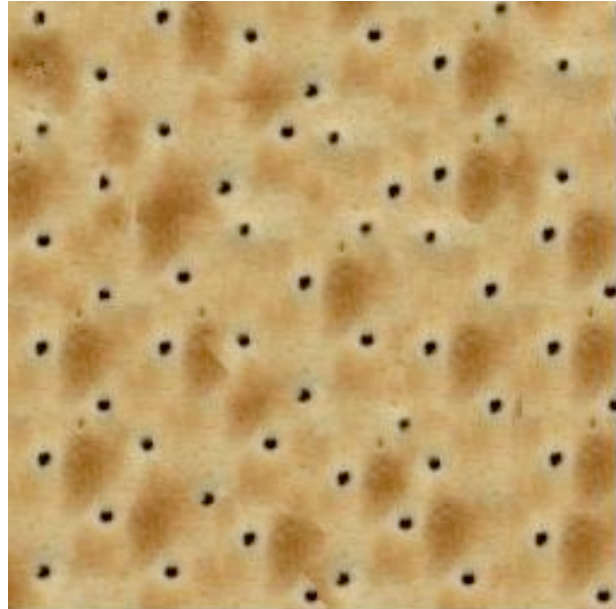
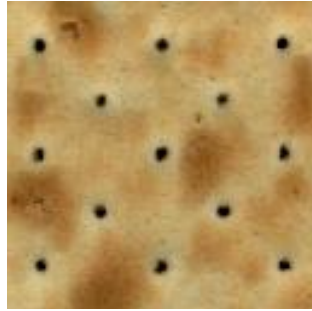






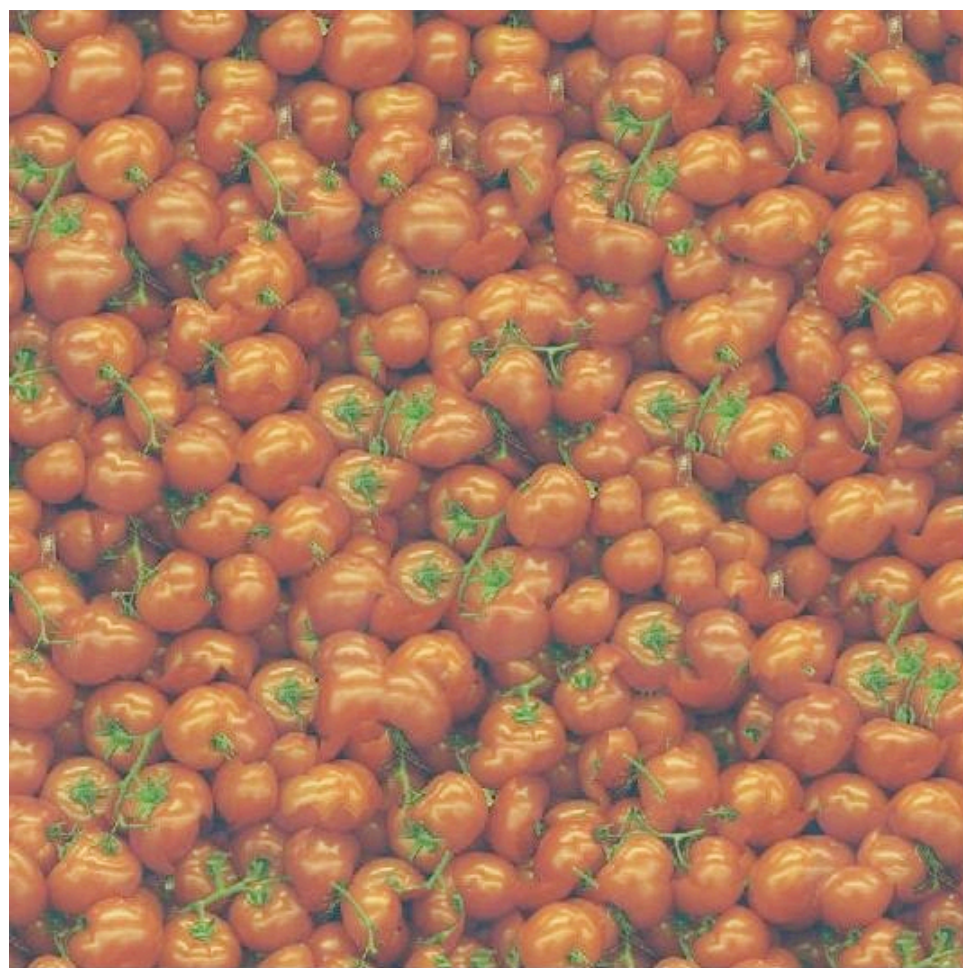


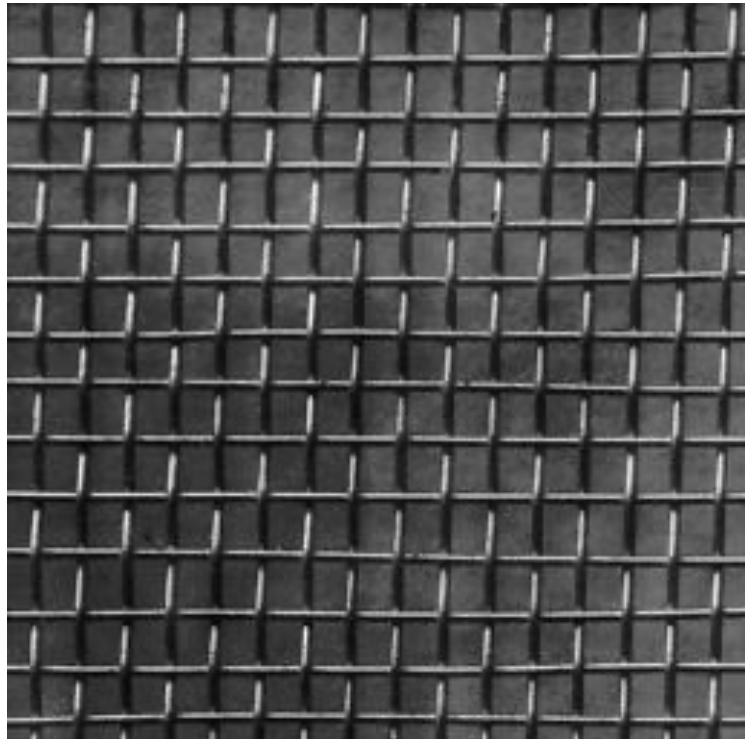




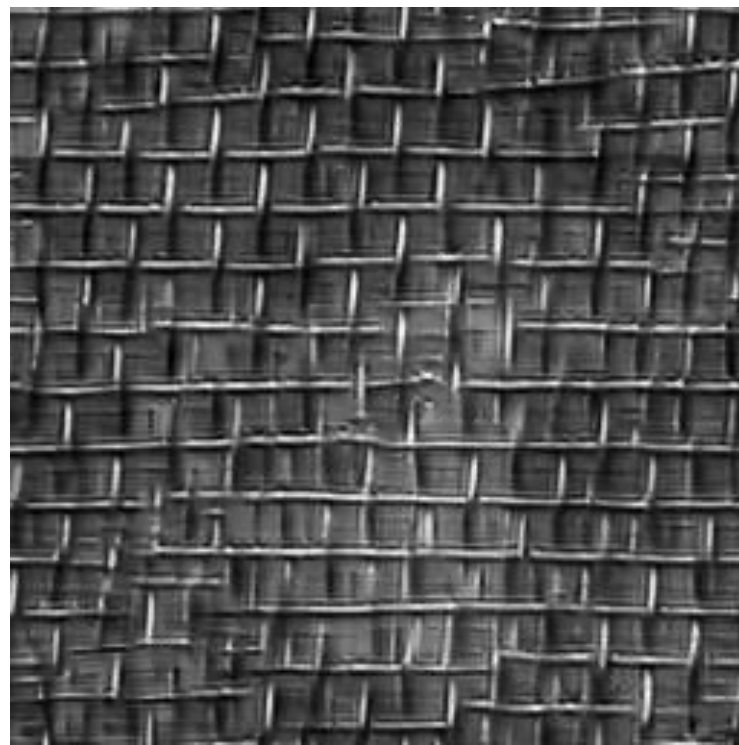


Failures (Chernobyl Harvest)

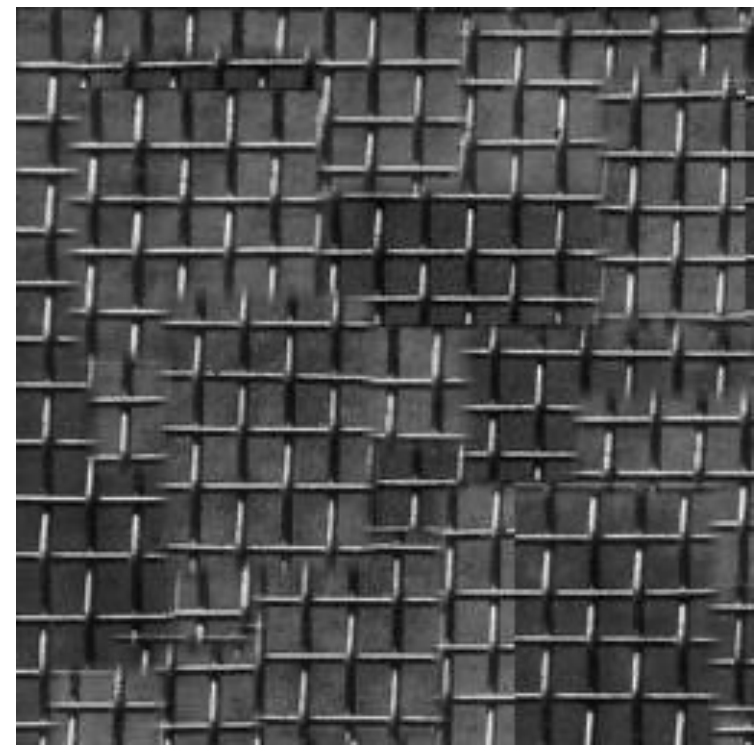




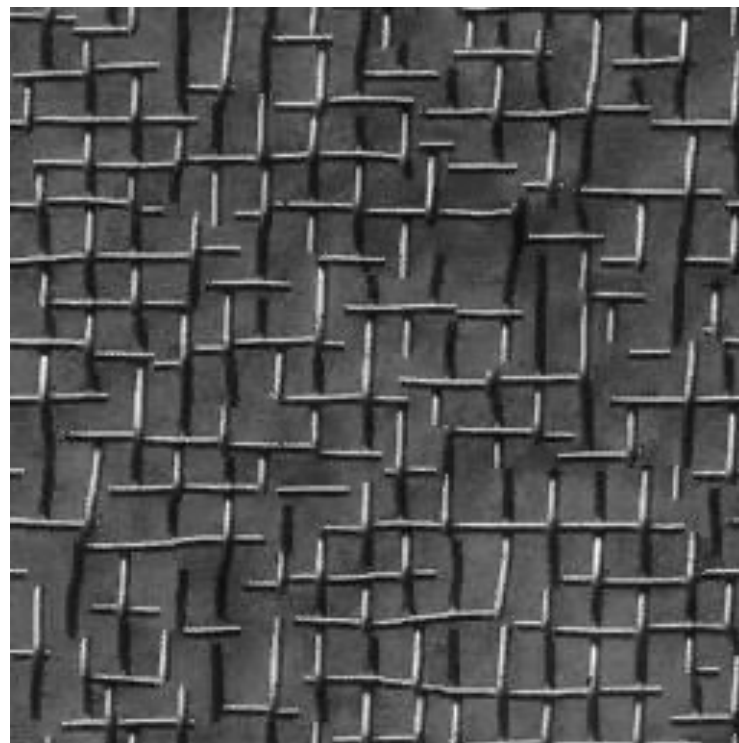
input image



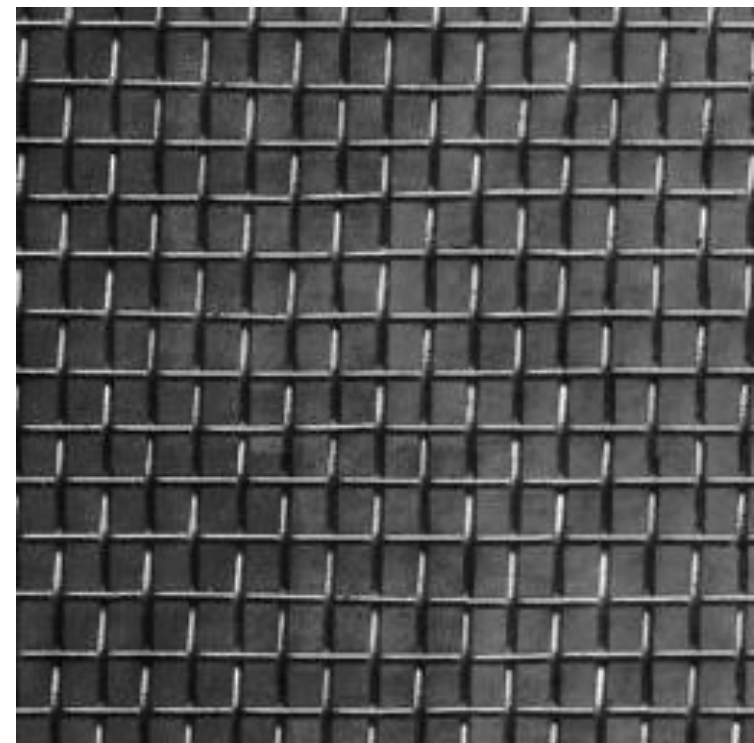
Portilla & Simoncelli



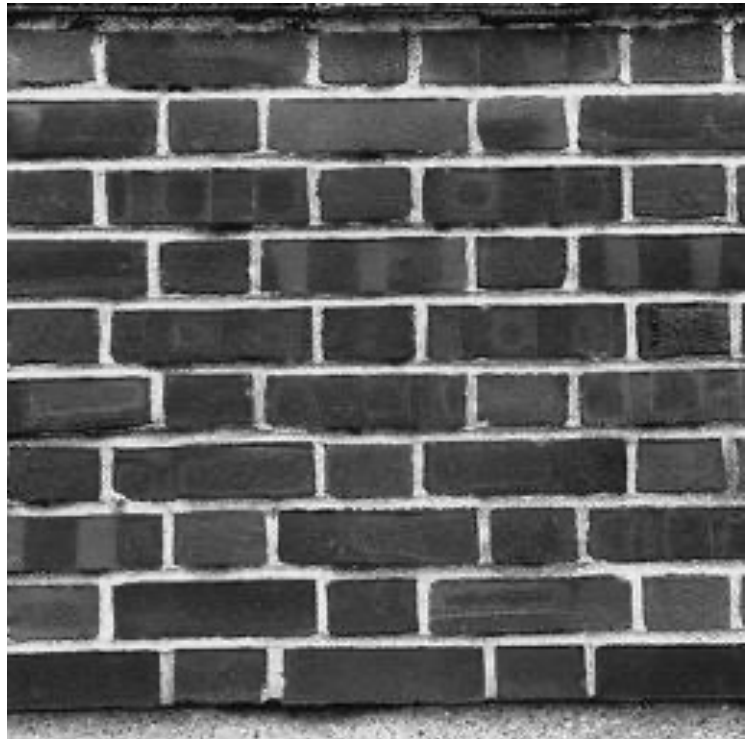
Xu, Guo & Shum



Wei & Levoy



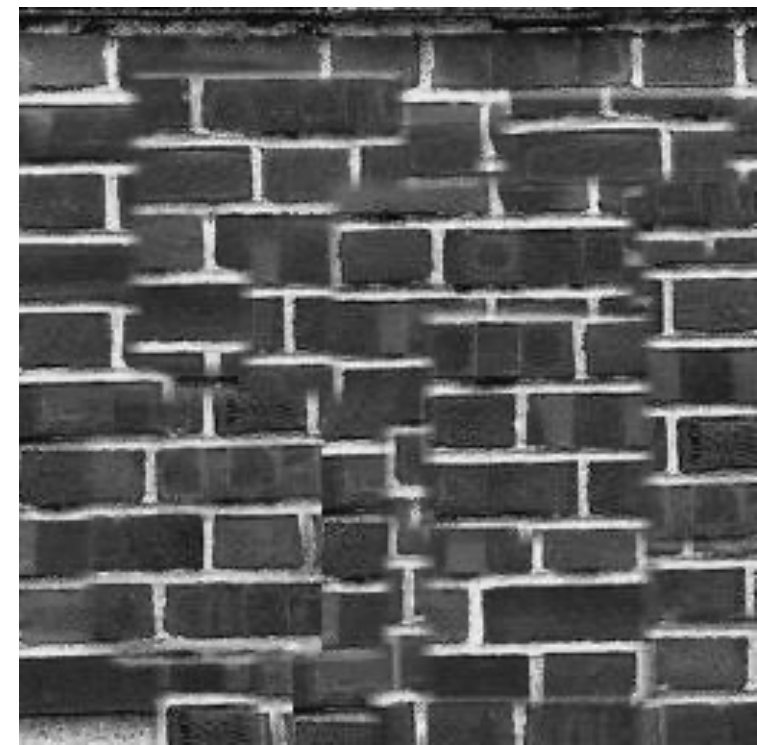
Efros and Freeman



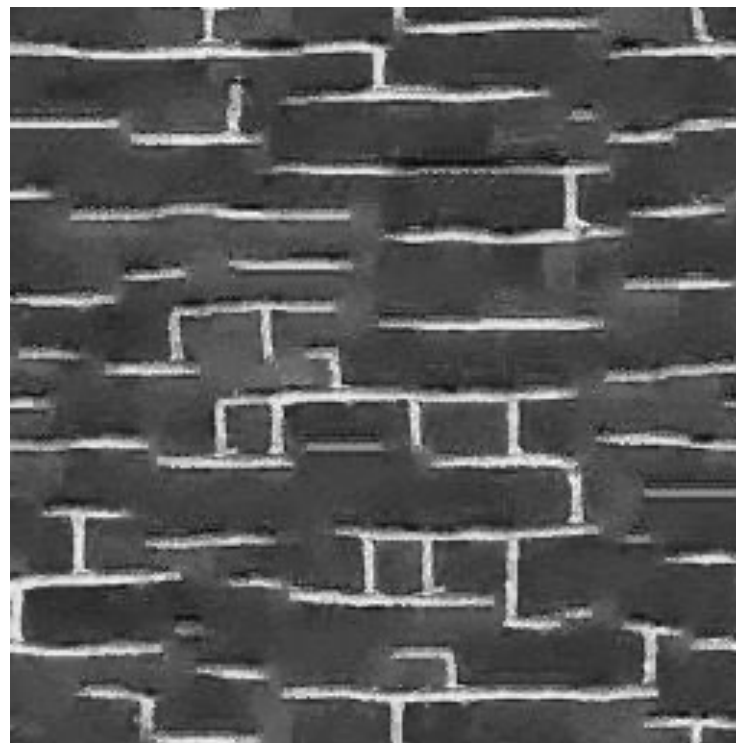
input image



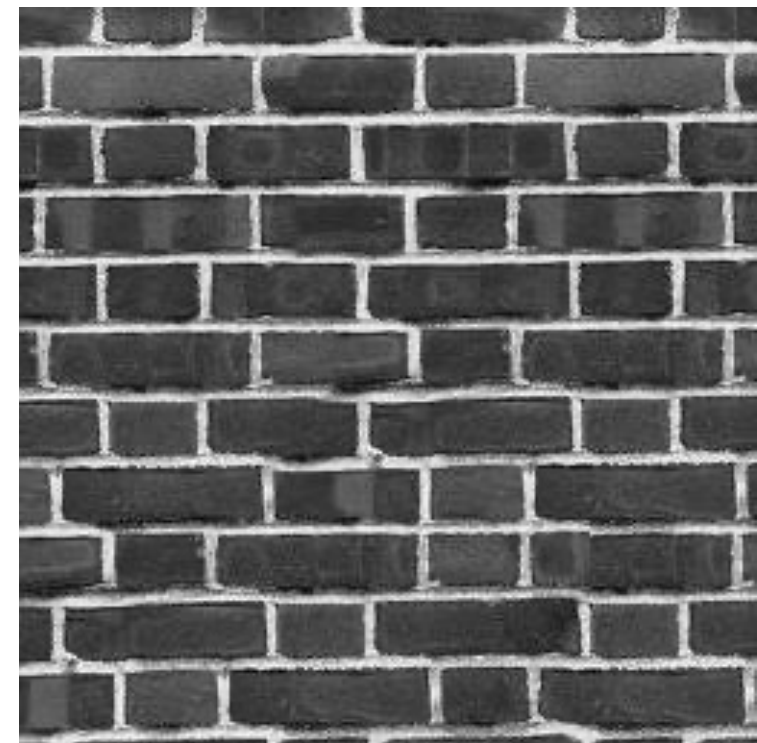
Portilla & Simoncelli



Xu, Guo & Shum



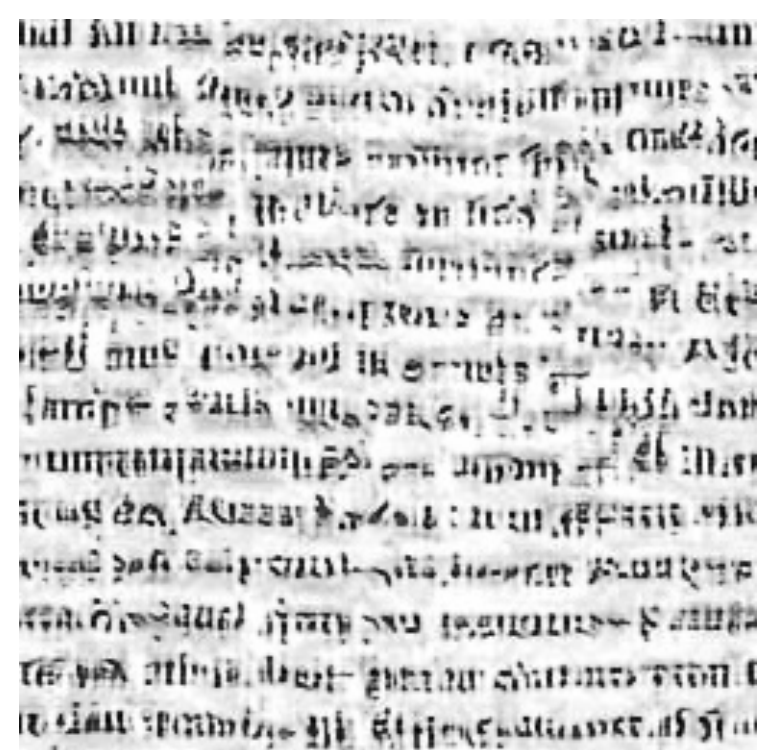
Wei & Levoy



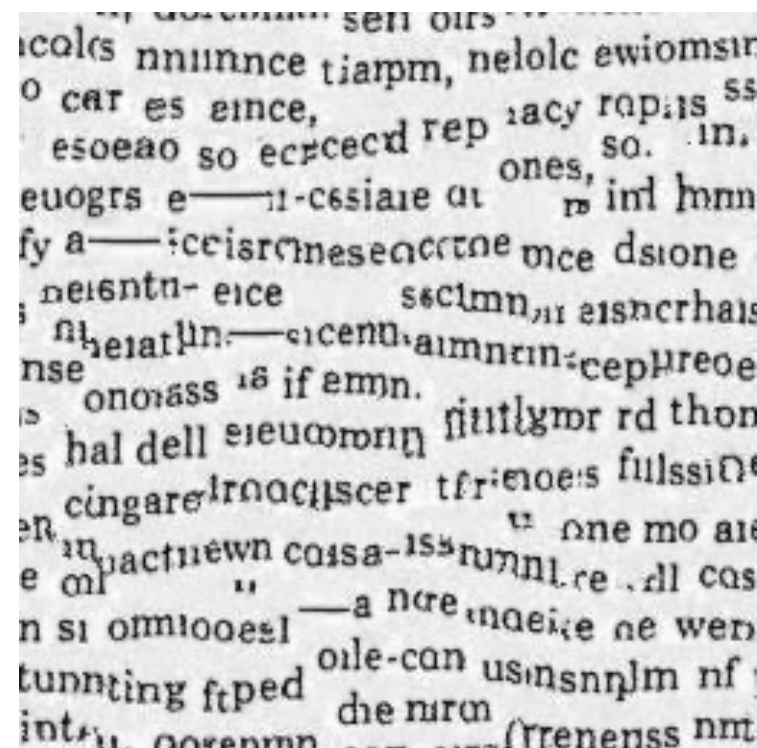
Efros and Freeman

field of a visual cortical neuron—the in-
describing the response of that neuron
ht as a function of position—is perhaps
functional description of that neuron.
seek a single conceptual and mathemati-
describe the wealth of simple-cell recep-
and neurophysiologically¹⁻³ and inferred
especially if such a framework has the
it helps us to understand the function
eeper way. Whereas no generic model
ussians (DOG), difference of offset C
rivative of a Gaussian, higher derivati-
function, and so on—can be expected
imple-cell receptive field, we nonetheless

input image



Portilla & Simoncelli



Wei & Levoy

field of a visual cortical neuron—the in-
des and so on—can be expected
ht as a function of position—is perhaps
functional description of that neuron.
seek a single conceptual and mathemati-
describe the wealth of simple-cell recep-
and neurophysiologically¹⁻³ and inferred
especially if such a framework has the
it helps us to understand the function
eeper way. Whereas no generic model
ussians (DOG), difference of offset C
rivative of a Gaussian, higher derivati-
function, and so on—can be expected
imple-cell receptive field, we nonetheless

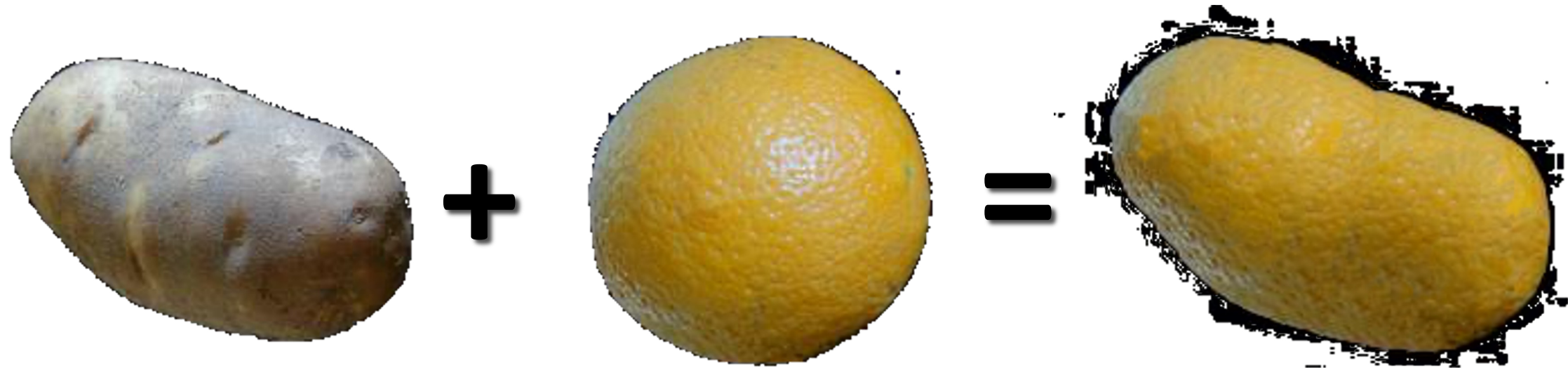
Xu, Guo & Shum

position—is perhaps a single conceptual and
of that neuron—the wealth of simple-
ual and mathematically¹⁻³ and
simple-cell receptive field, we nonetheless
y¹⁻³ and inferred helps us to understand the
ramework has perhaps. Whereas no ge-
and the function (DOG), difference of
s no generic a single conceptual and m-
rence of offset the wealth of simple-ce-
, higher derivatives describing the response of
—can be expected as a function of position—
helps us to understand the description of the
per way. Whereas no generic conceptual and
ussians (DOG), difference of simple

Efros and Freeman

Application: Texture Transfer

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

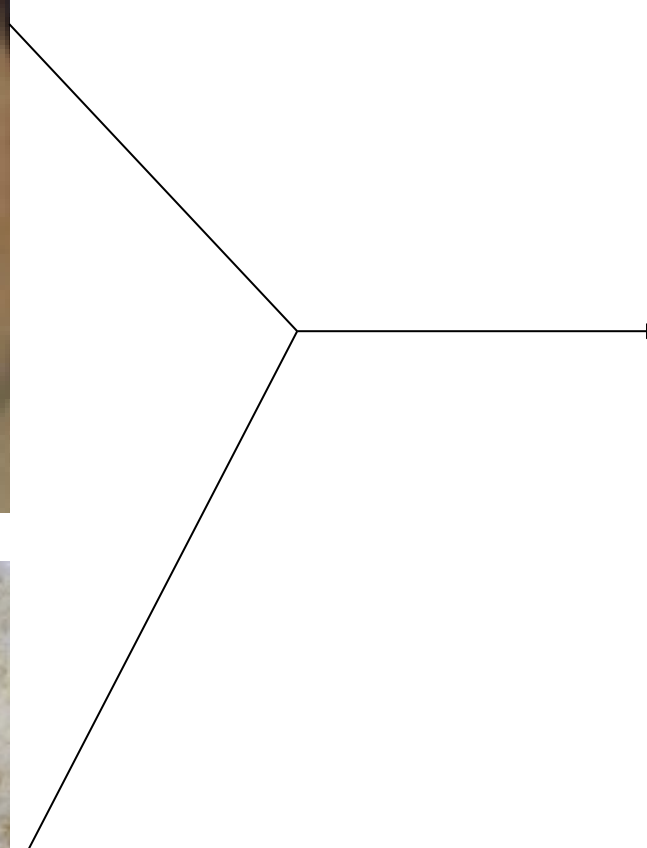


Texture Transfer

Constraint



Texture sample



Texture Transfer

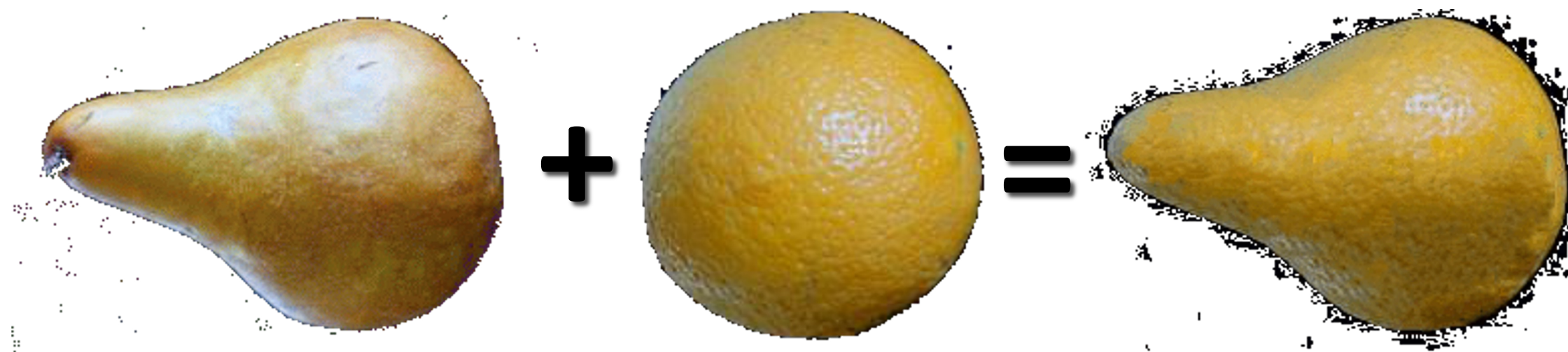
- Take the texture from one image and “paint” it onto another object



Same as texture synthesis, except an additional constraint:

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

Texture Transfer



Texture Transfer

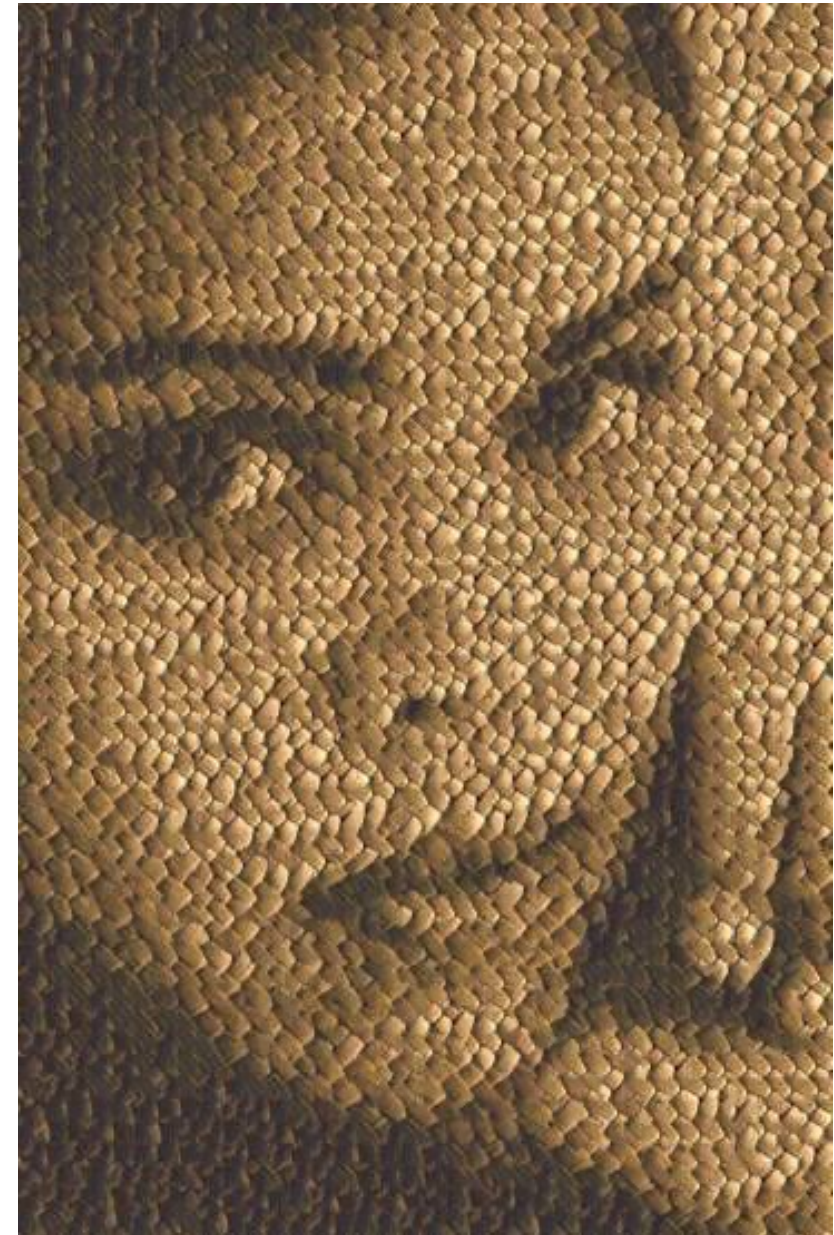


Image Analogies

Aaron Hertzmann^{1,2}

Chuck Jacobs²

Nuria Oliver²

Brian Curless³

David Salesin^{2,3}

¹**New York University**

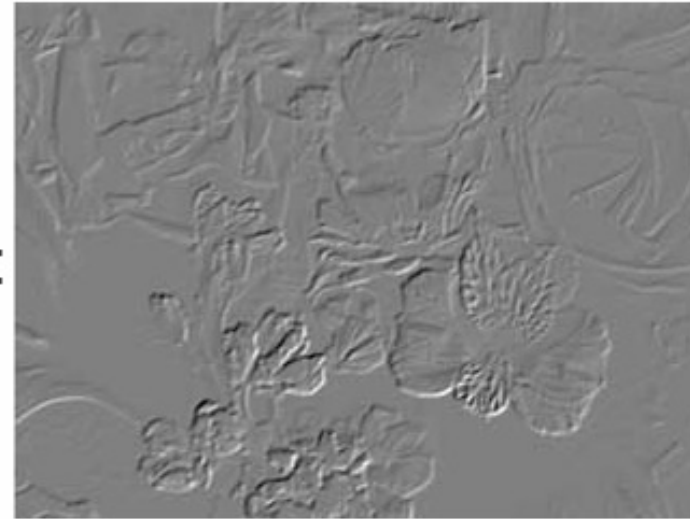
²**Microsoft Research**

³**University of Washington**

Edge Filter



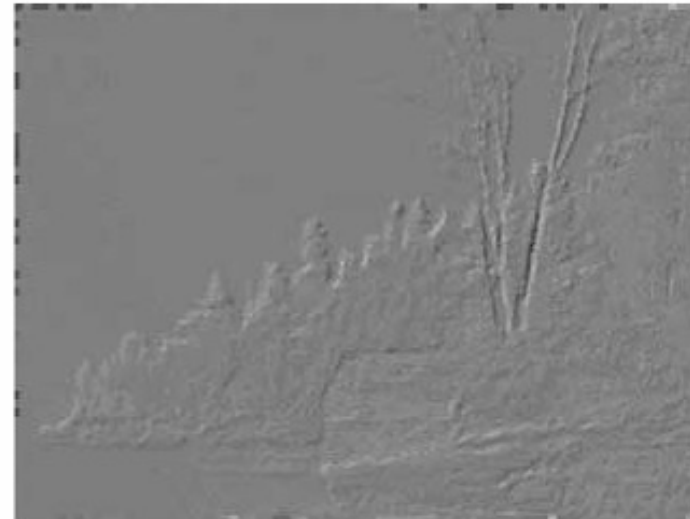
Unfiltered source (A)



Filtered source (A')



Unfiltered target (B)



Filtered target (B')

Artistic Filters



A



A'



B



B'

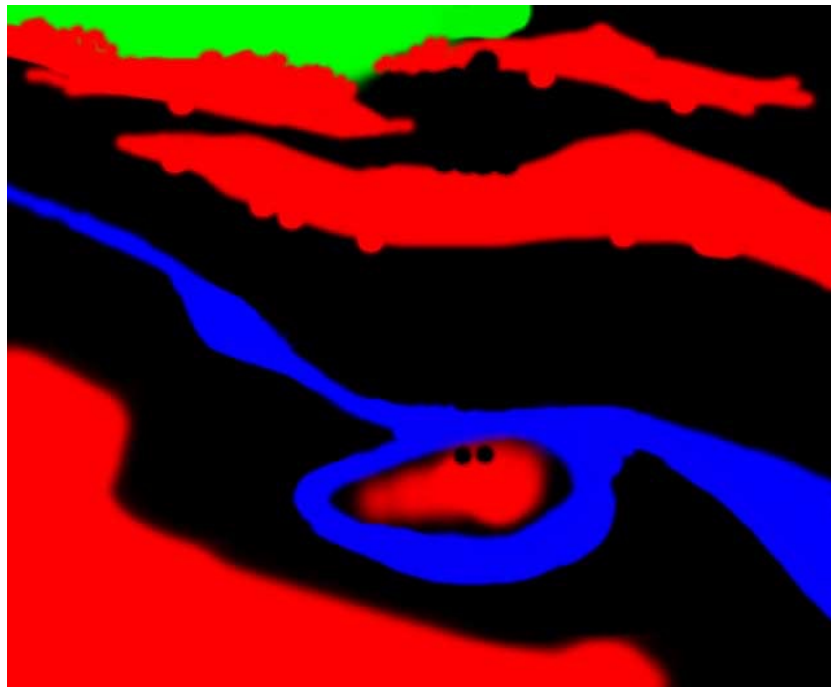
Texture-by-numbers



A



A'

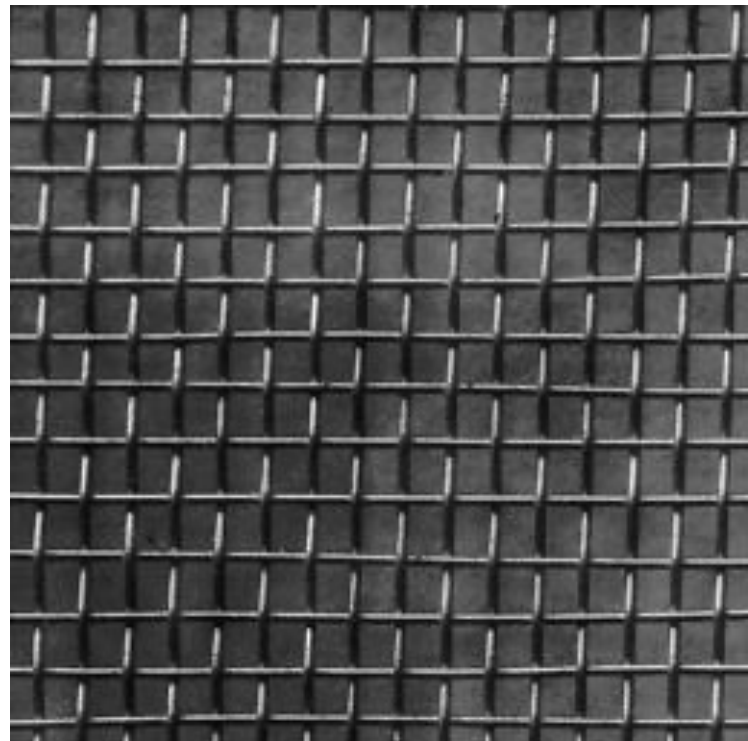


B

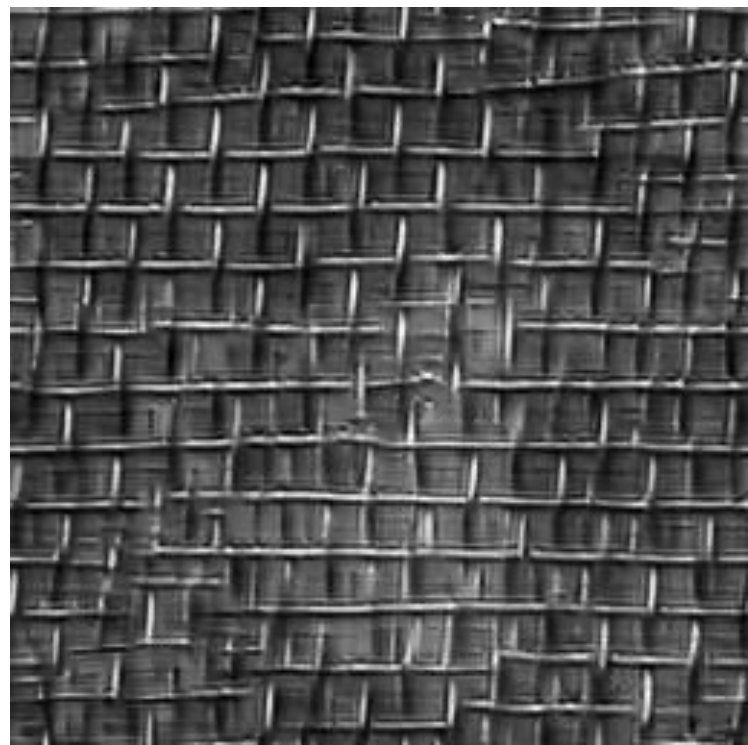


B'

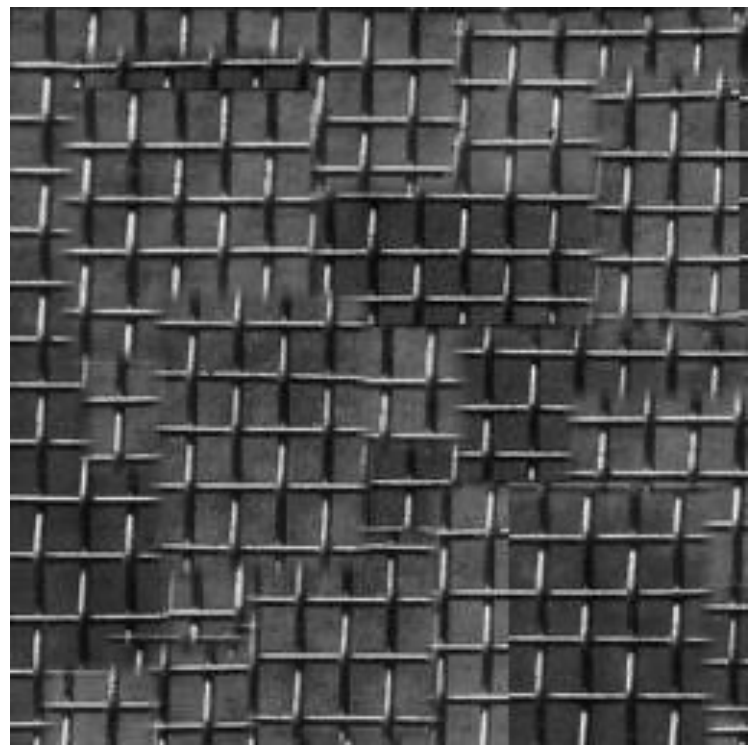
Parametric Texture Synthesis



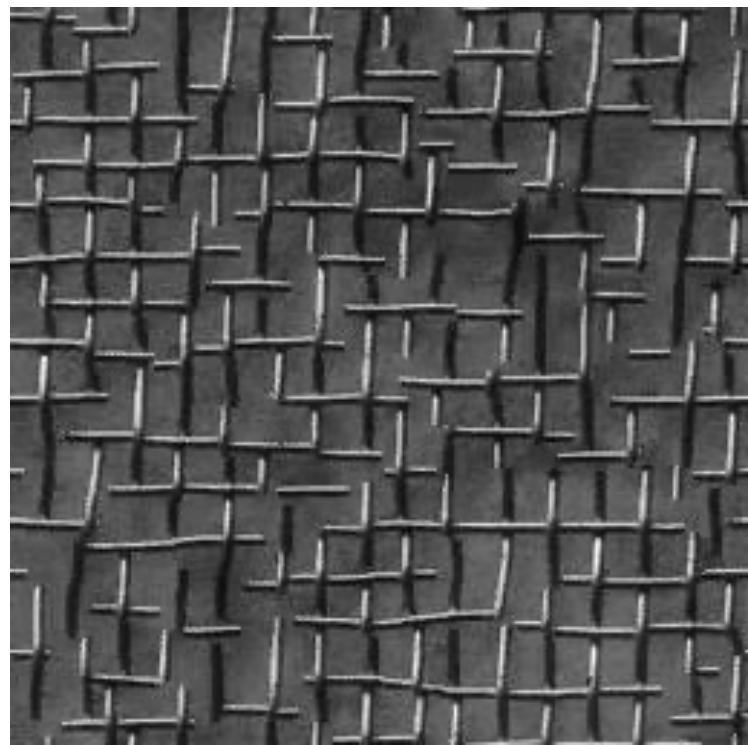
input image



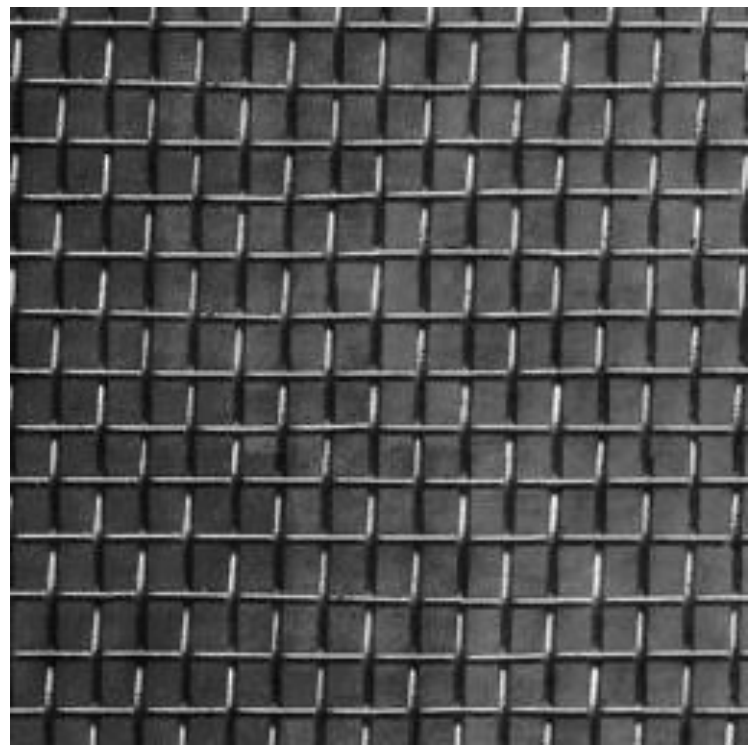
Portilla & Simoncelli



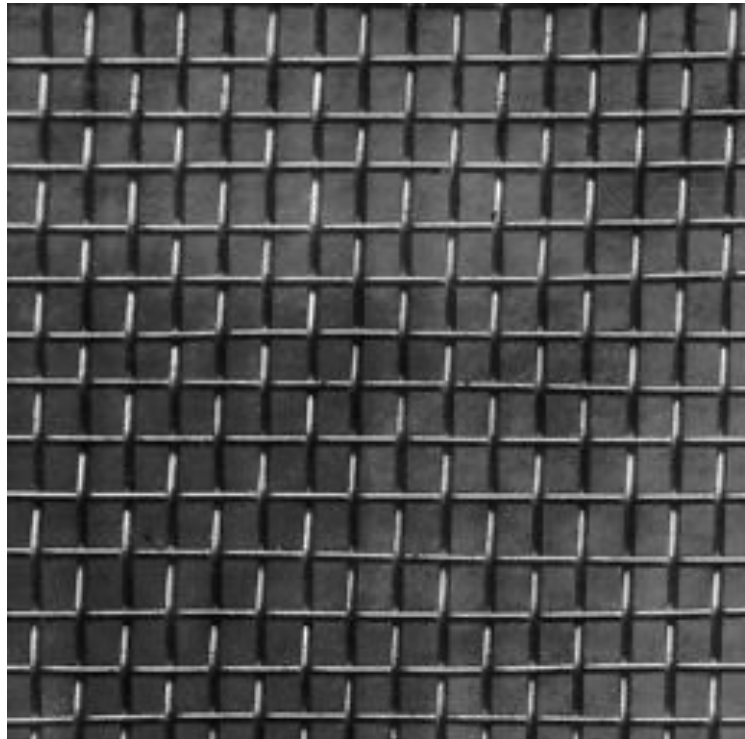
Xu, Guo & Shum



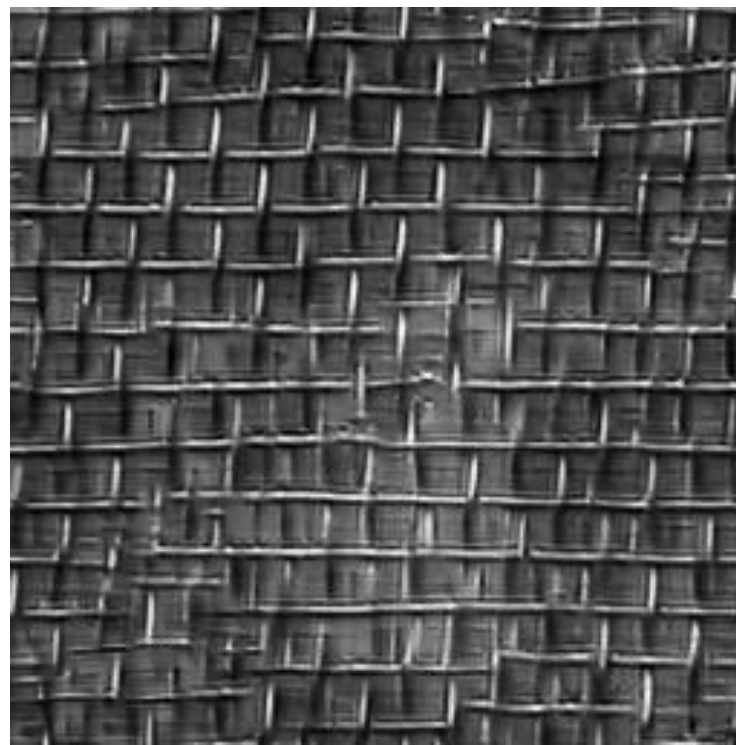
Wei & Levoy



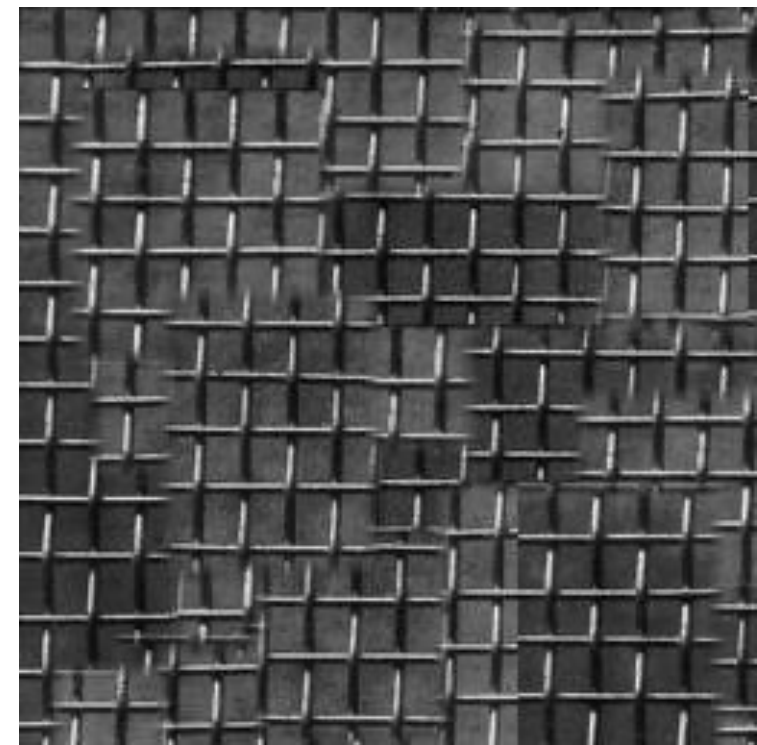
Efros and Freeman



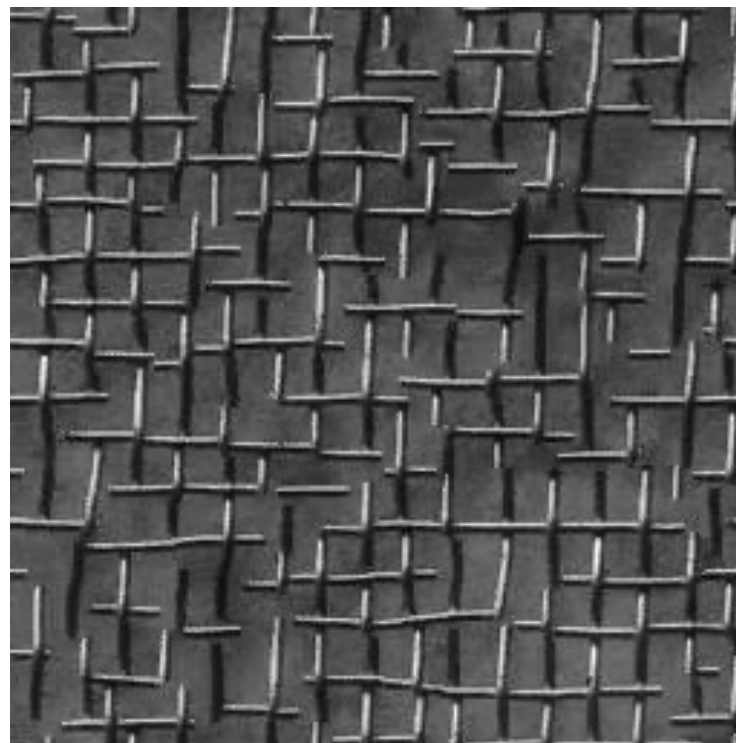
input image



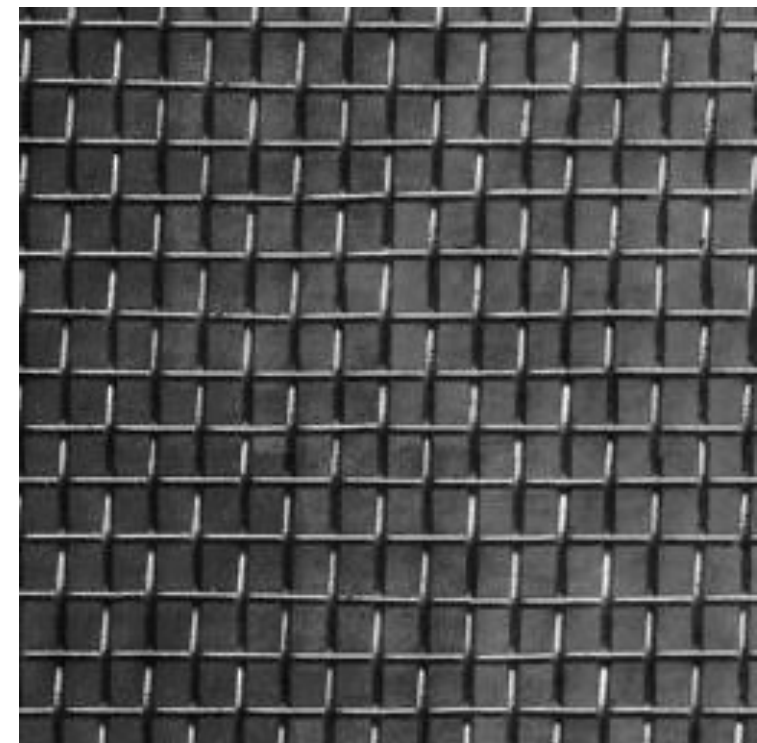
Portilla & Simoncelli



Xu, Guo & Shum

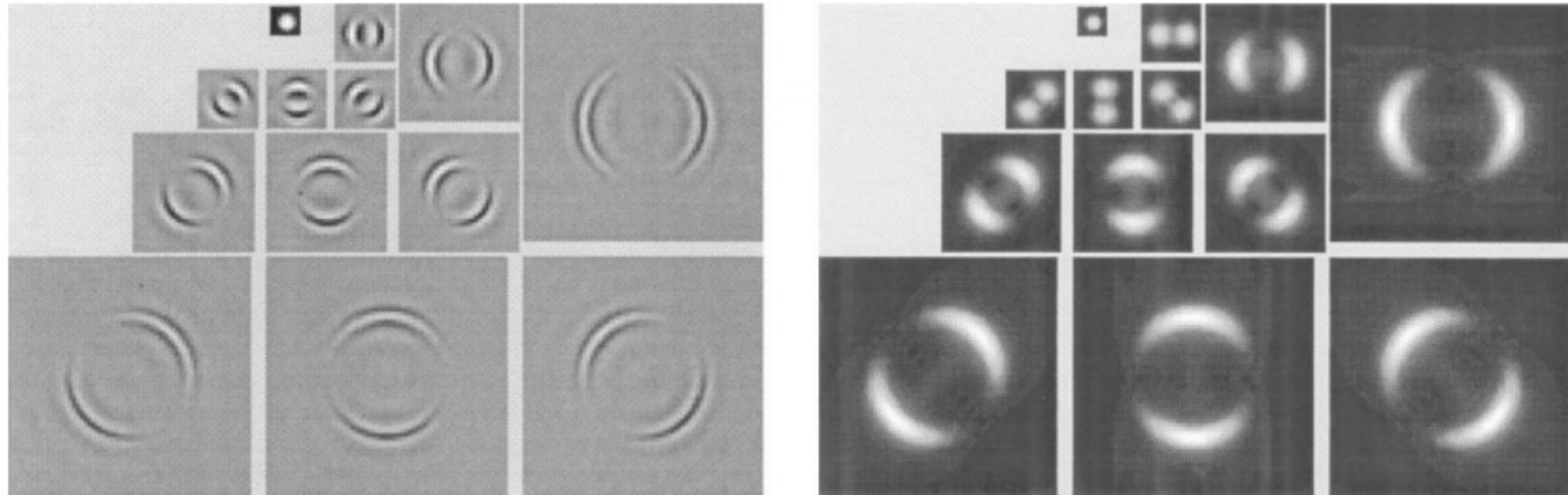


Wei & Levoy



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

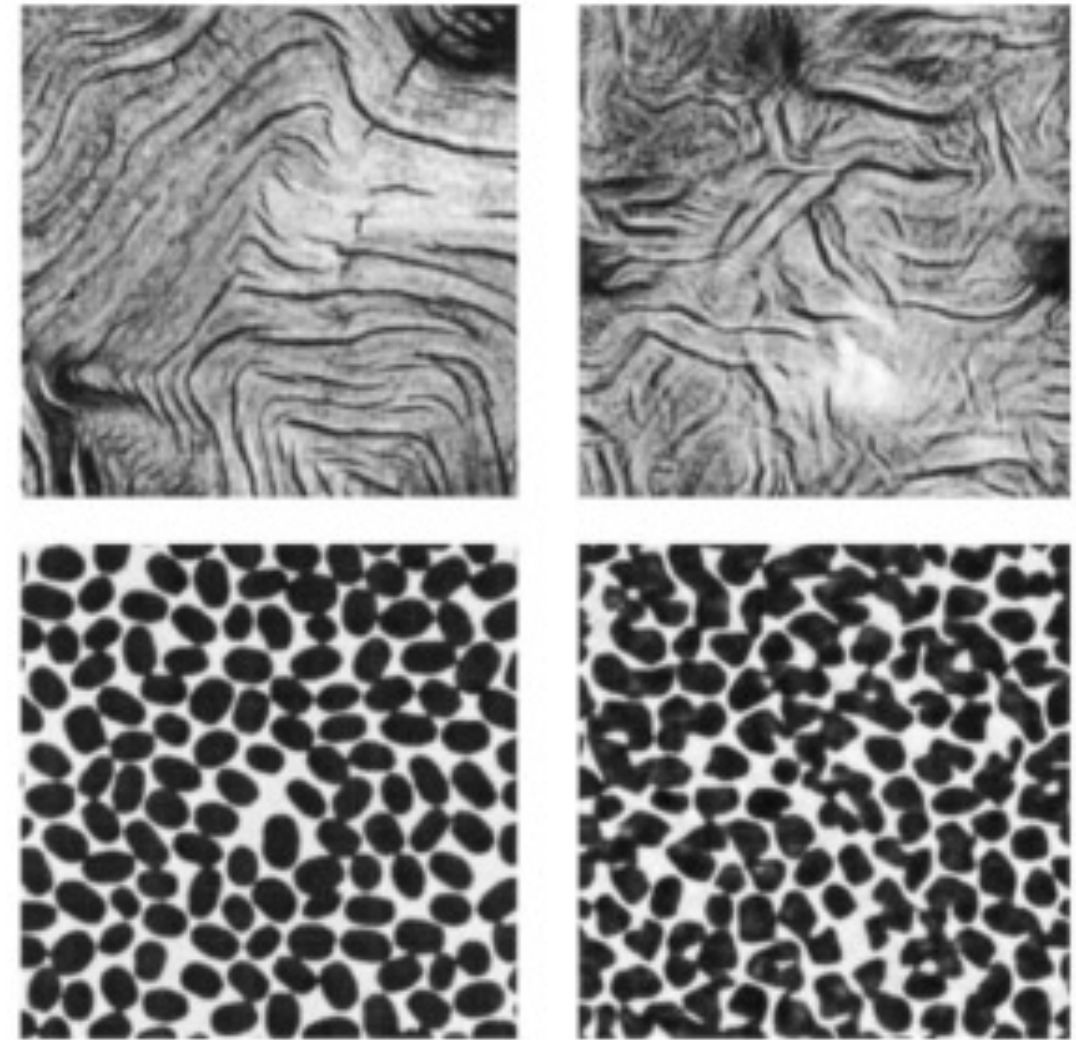
Parametric Texture Synthesis

Objective function

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

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

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

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

h, w: pixel locations index

c: channel index

H, W: height and width of feature map

C: the number of total channels

Style Reconstruction (Style Loss)

$$\left| \text{Gram} \left(\overset{\hat{y}}{\text{img}} \right) - \text{Gram} \left(\overset{y}{\text{img}} \right) \right|$$

optimized output style image

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

Style Loss

$$\arg \min_{\hat{y}} \sum_j^M \overset{\text{weight}}{\lambda_j} \left\| \overset{(j)\text{-th layer}}{\text{Gram}^{(j)}(\hat{y})} - \text{Gram}^{(j)}(y) \right\|^2$$

Portilla & Simoncelli



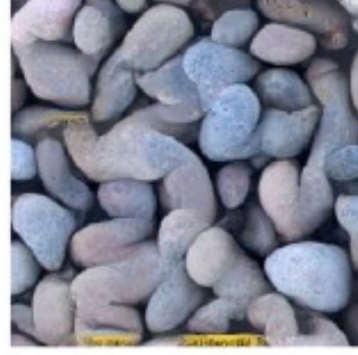
original



pool4



pool3



A ~1k parameters



~10k parameters



~177k parameters



~852k parameters

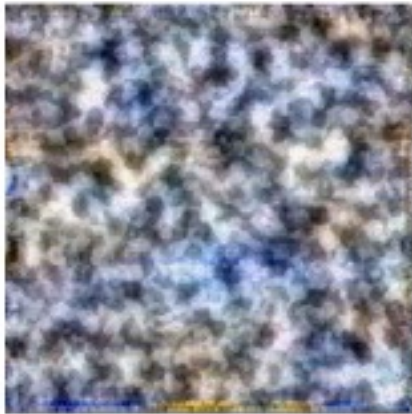


original



Number of parameters

B conv1



conv2



conv3



conv4



conv5

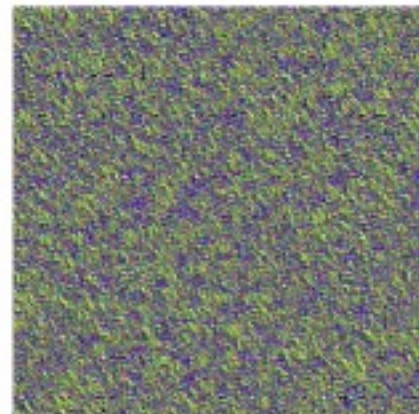


Different layers

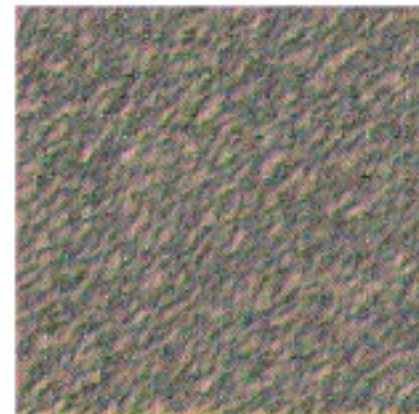
C conv1_1



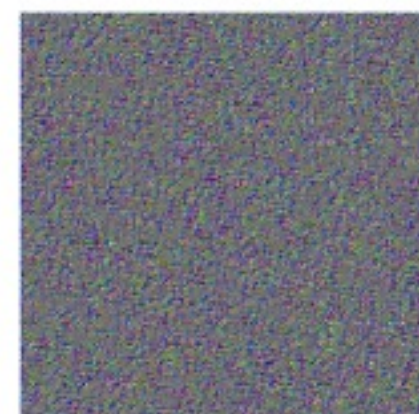
pool1



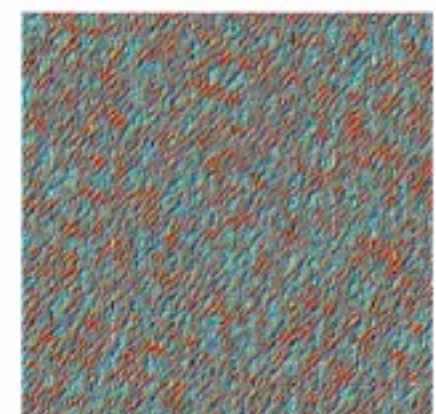
pool2



pool3



pool4



The same network architecture with random weights

Neural Style Transfer



content image

+



style image

=



output result

Content Reconstruction (Perceptual Loss)

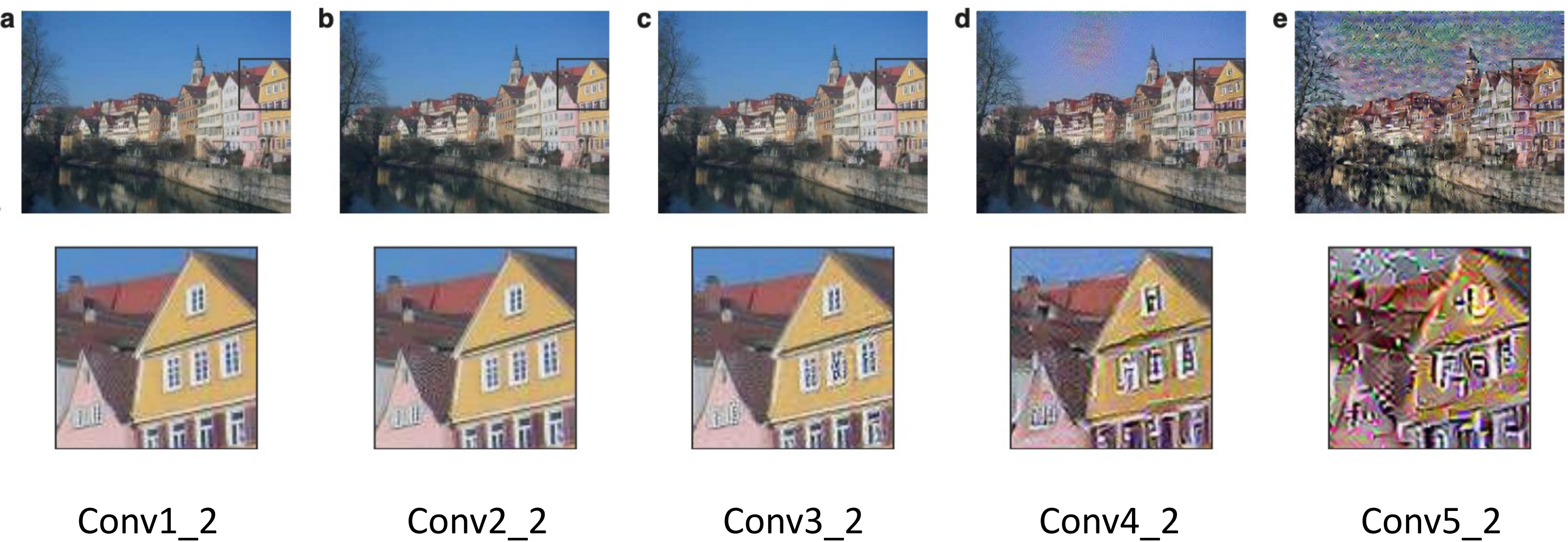
$$\left| F\left(\overset{\hat{y}}{\text{optimized output}}\right) - F\left(\overset{x}{\text{content image}}\right) \right|$$

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

Content Loss

$$\arg \min_{\hat{y}} \sum_i^N \overset{\text{weight}}{\lambda_i} \left\| \overset{\text{(i)-th layer}}{F^{(i)}}(\hat{y}) - F^{(i)}(x) \right\|_1$$

Content Reconstruction (Perceptual Loss)



Neural Style Transfer

$$|\text{Gram}(\overset{\hat{y}}{\text{optimized output}}) - \text{Gram}(\overset{y}{\text{style image}})|$$

$$+ |\text{F}(\overset{\hat{y}}{\text{optimized output}}) - \text{F}(\overset{x}{\text{content image}})|$$

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



Different Initializations



Fast Neural Style Transfer

- Optimization-based method

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

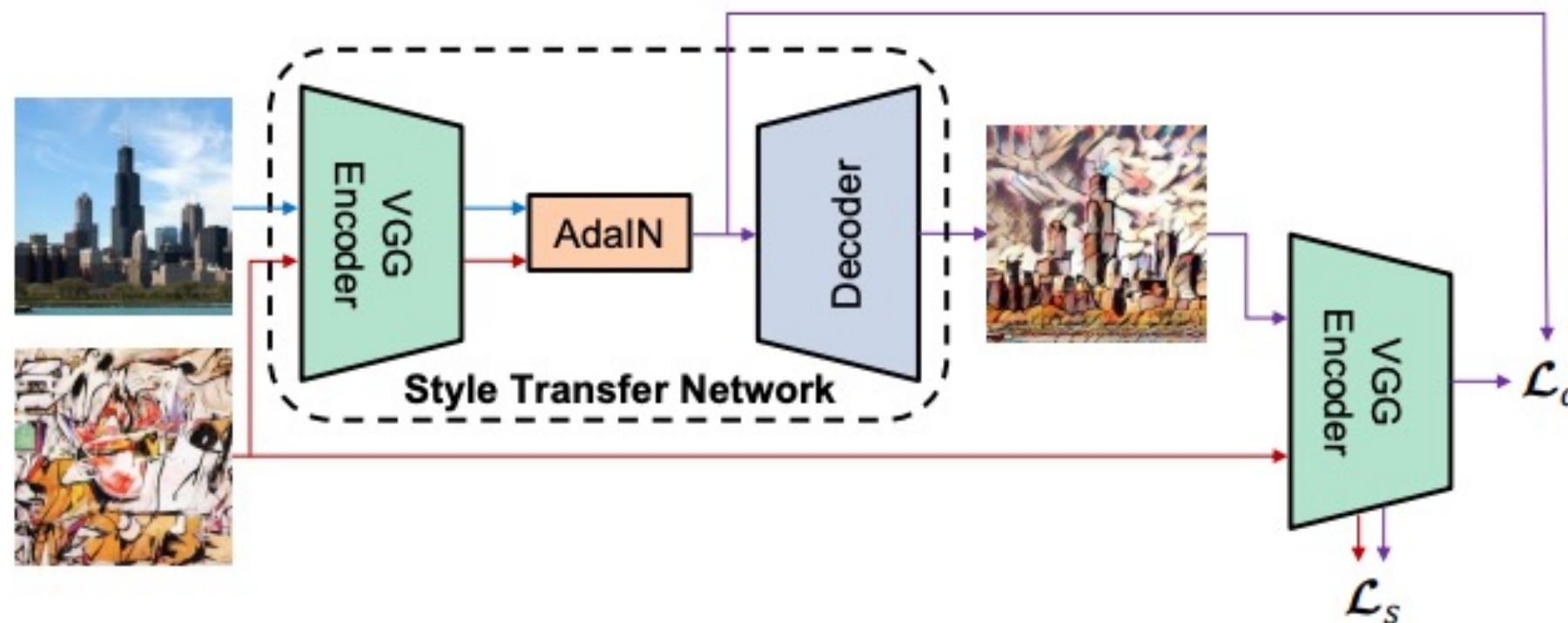
- Feedforward network

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

Arbitrary Style Transfer with AdaIN

- Feedforward network with any style

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



Arbitrary Style Transfer with AdaIN



Neural Style Transfer vs. Image-to-Image Translation

Input



Style Image I



Style image II



Entire collection



CycleGAN

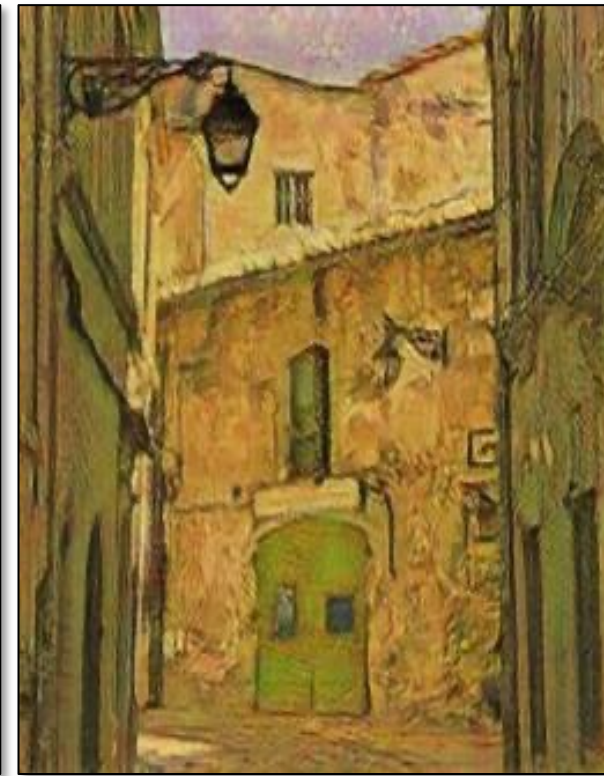


Photo → Van Gogh

Input



Style image I



Style image II



Entire collection



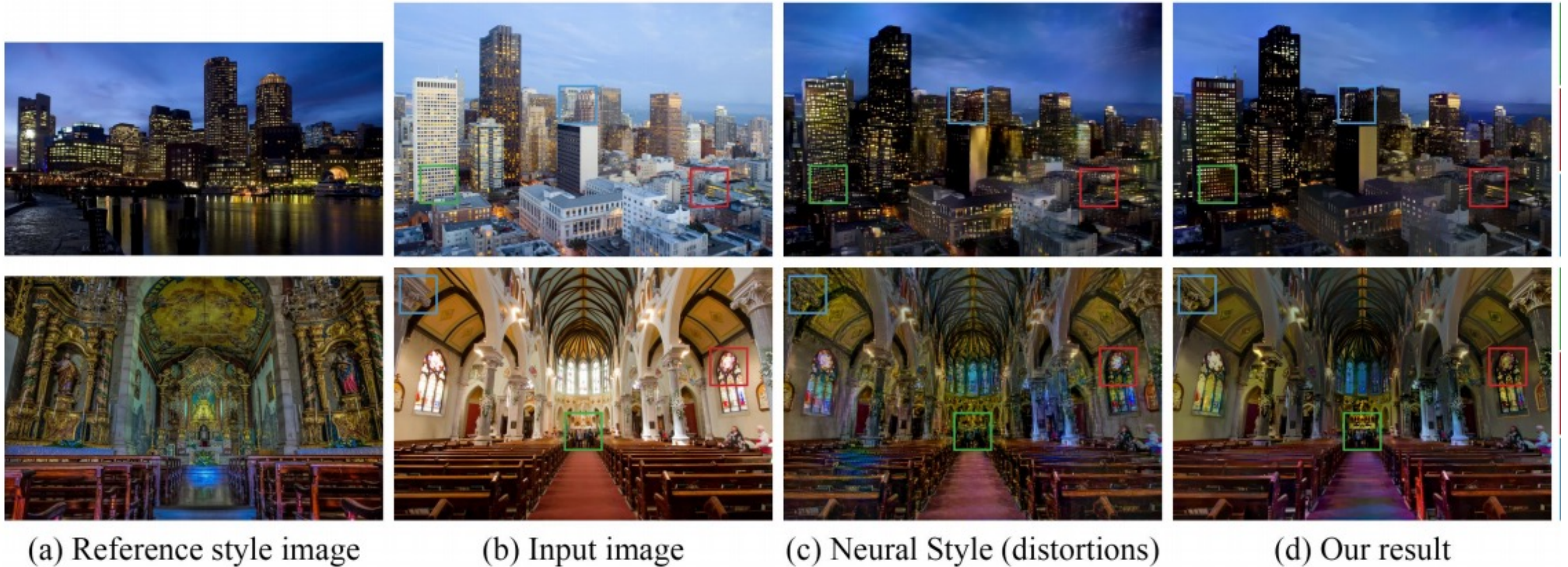
CycleGAN



horse → zebra

Photo Style Transfer

Deep Photo Style Transfer



Local color transfer? (hard to transfer texture)