CSCE 448/748 - Computational Photography

Texture Synthesis

Nima Kalantari

Many slides from Alexei A. Efros

Texture

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

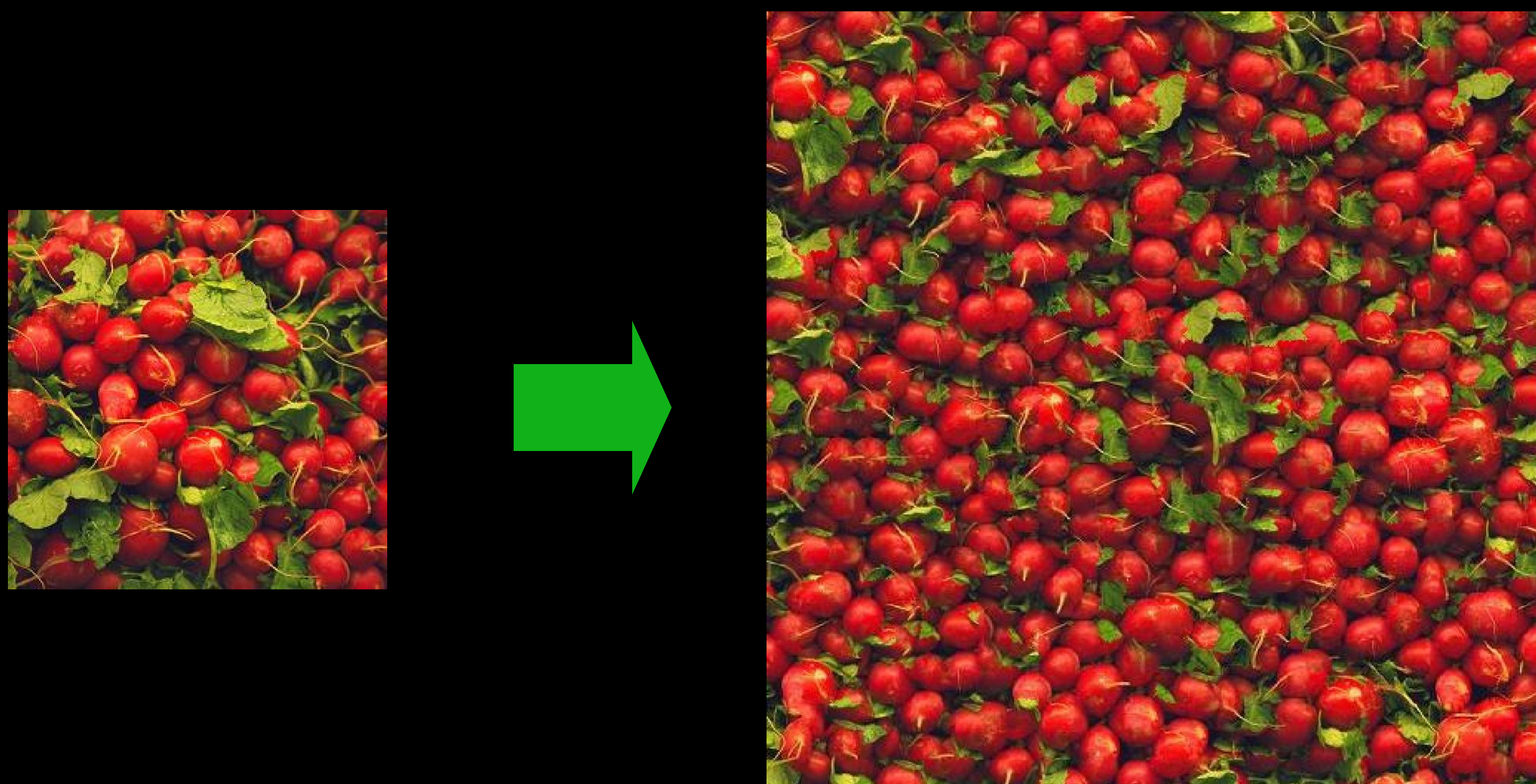

radishes


rocks

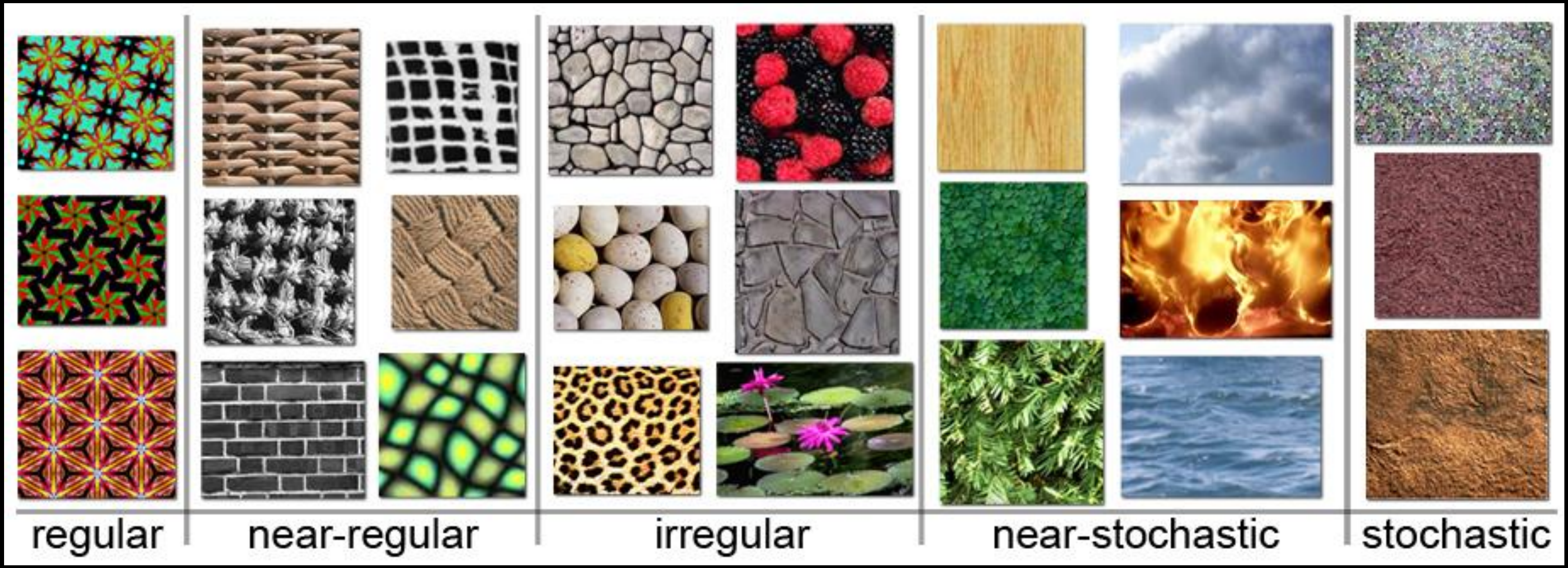

yogurt

Texture Synthesis

- Goal of Texture Synthesis: create new samples of a given texture
- Many applications: hole-filling, texturing surfaces



The Challenge



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

Weather Forecasting

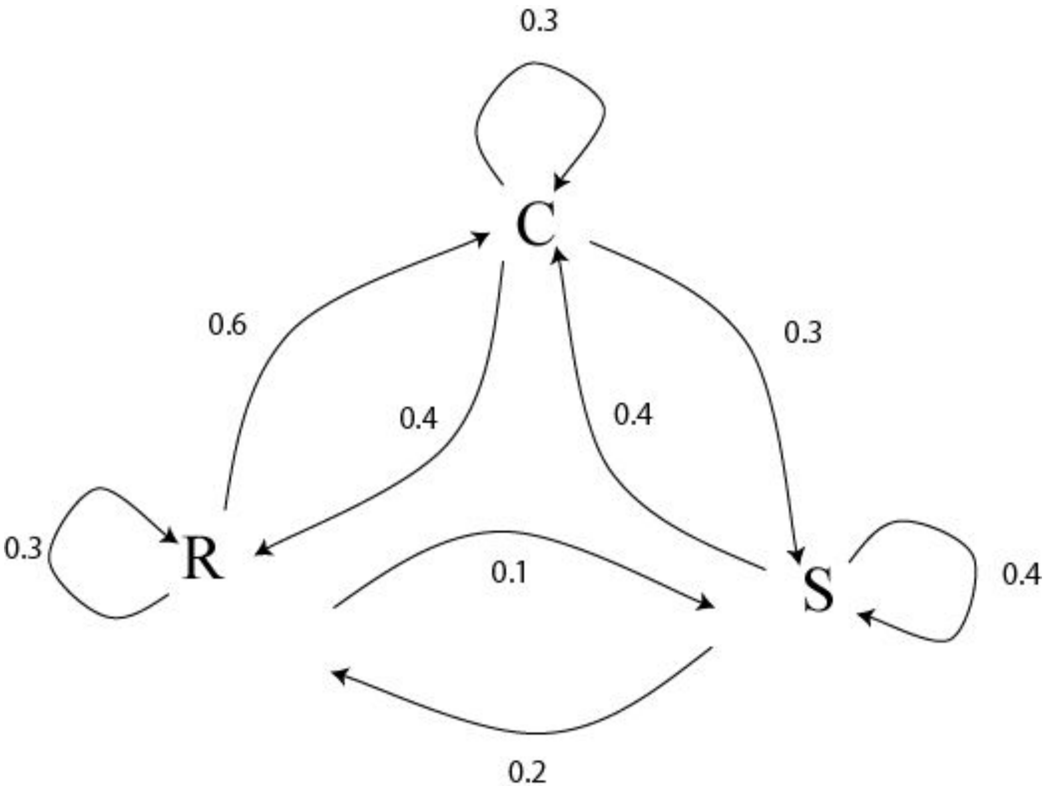
Let's predict weather:

- Given today's weather only, we want to know tomorrow's
- Suppose weather can only be {**S**unny, **C**loudy, **R**aining}

The "Weather Channel" algorithm:

- Over a long period of time, record:
  - How often S followed by R
  - How often S followed by S
  - Etc.
- Compute percentages for each state:
  - P(R|S), P(S|S), etc.
- Predict the state with highest probability!
- It's a Markov Chain

Markov Chain



$$\begin{array}{cc} & \text{Tomorrow} \\ & \begin{array}{ccc} \mathrm{R} & \mathrm{C} & \mathrm{S} \end{array} \\ \text{Today} \begin{array}{c} \mathrm{R} \\ \mathrm{C} \\ \mathrm{S} \end{array} & \begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{pmatrix} \end{array}$$

What if we know today and yesterday’s weather?

Second Order Markov Chain

Tomorrow

$$
\begin{array}{c}
\\ \text{Today} \\ \\
\end{array}
\begin{array}{c}
\\ R \\ C \\ S
\end{array}
\begin{array}{c}
\begin{array}{ccc} R & C & S \end{array} \\
\begin{pmatrix}
0.3 & 0.6 & 0.1 \\
0.4 & 0.3 & 0.3 \\
0.2 & 0.4 & 0.4
\end{pmatrix}
\end{array}
$$

$$
\text{Observation}
\begin{array}{c}
R,R \\ R,C \\ R,S \\ C,R \\ C,C \\ C,S \\ S,R \\ S,C \\ S,S
\end{array}
\begin{pmatrix}
0.3 & 0.6 & 0.1 \\
0.4 & 0.3 & 0.3 \\
0.2 & 0.4 & 0.4 \\
0.3 & 0.6 & 0.1 \\
0.4 & 0.3 & 0.3 \\
0.2 & 0.4 & 0.4 \\
0.3 & 0.6 & 0.1 \\
0.4 & 0.3 & 0.3 \\
0.2 & 0.4 & 0.4
\end{pmatrix}
$$

Markov Chains

Markov Chain

- a *sequence* of random variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
- $\mathbf{x}_t$ is the **state** of the model at time t

$$\boxed{\mathbf{x}_1} \rightarrow \boxed{\mathbf{x}_2} \rightarrow \boxed{\mathbf{x}_3} \rightarrow \boxed{\mathbf{x}_4} \rightarrow \boxed{\mathbf{x}_5}$$

- **Markov assumption**: each state is dependent only on the previous one
  - dependency given by a **conditional probability**:

$$p(\mathbf{x}_t | \mathbf{x}_{t-1})$$

- The above is actually a *first-order* Markov chain
- An *N'th-order* Markov chain:

$$p(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-N})$$

Text synthesis

Create plausible looking poetry, love letters, term papers, etc.

Most basic algorithm

1. Build probability histogram
   - find all blocks of N consecutive words/letters in training documents
   - compute probability of occurance $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-(n-1)})$
2. Given words $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}$
   - compute $\mathbf{x}_k$ by sampling from $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-(n-1)})$

Markov Chain Example: Text

"A dog is a man's best friend. It's a dog eat dog world out there."

$p(\mathbf{x}_t|\mathbf{x}_{t-1})$

$\mathbf{x}_{t-1}$ \ $\mathbf{x}_t$	a	dog	is	man's	best	friend	it's	eat	world	out	there	.
a		**2/3**		**1/3**								
dog			**1/3**					**1/3**	**1/3**			
is	**1**											
man's					**1**							
best						**1**						
friend												**1**
it's	**1**											
eat		**1**										
world										**1**		
out											**1**	
there												**1**
.							**1**					

Mark V. Shaney (Bell Labs)

Results:

- *“As I've commented before, really relating to someone involves standing next to impossible.”*
- *“One morning I shot an elephant in my arms and kissed him.”*
- *“I spent an interesting evening recently with a grain of salt”*

Markov Random Field

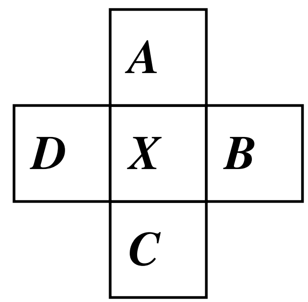
A Markov random field (MRF)

- generalization of Markov chains to two or more dimensions.

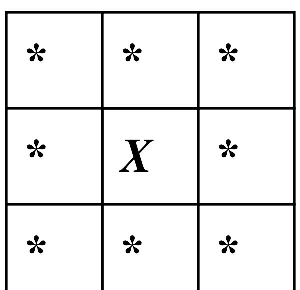
First-order MRF:

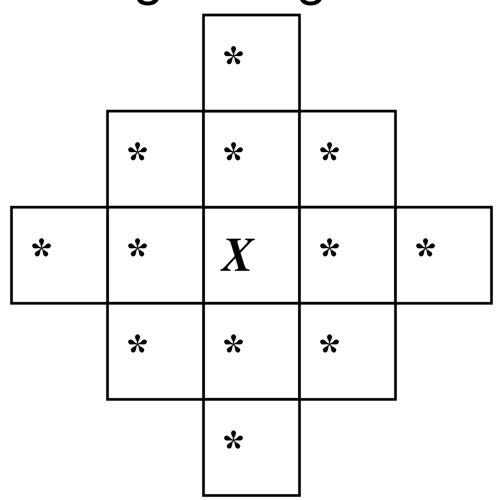
- probability that pixel *X* takes a certain value given the values of neighbors *A*, *B*, *C*, and *D*:

$$P(\mathbf{X}|\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$$

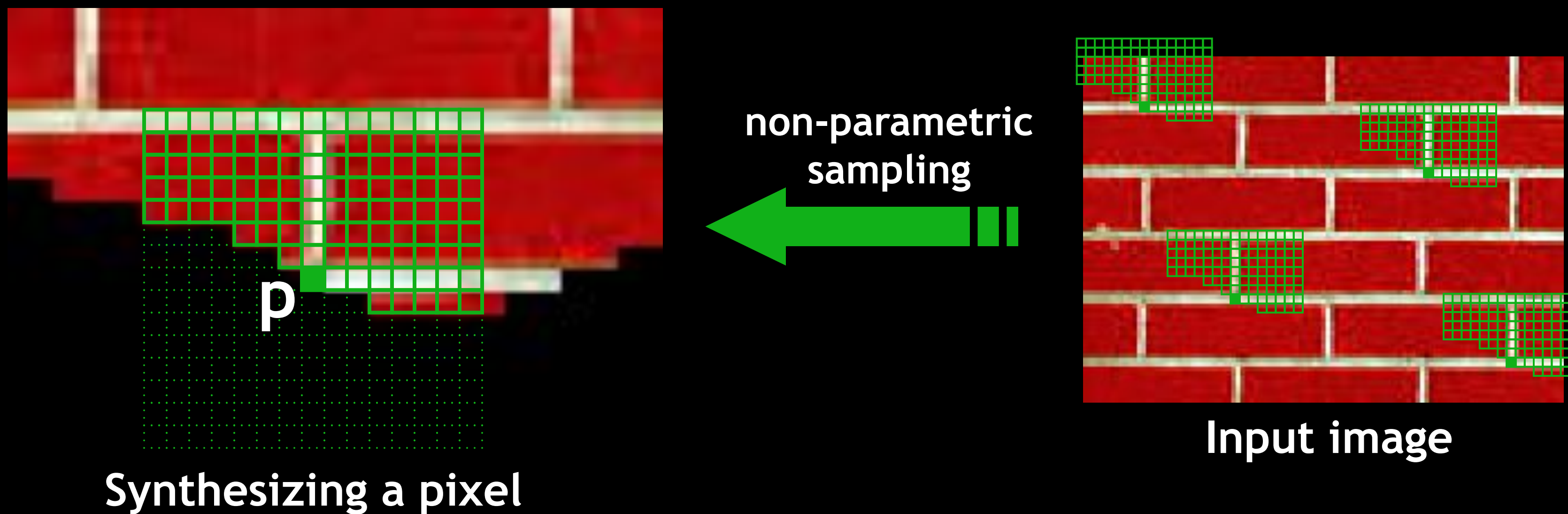


- Higher order MRF's have larger neighborhoods



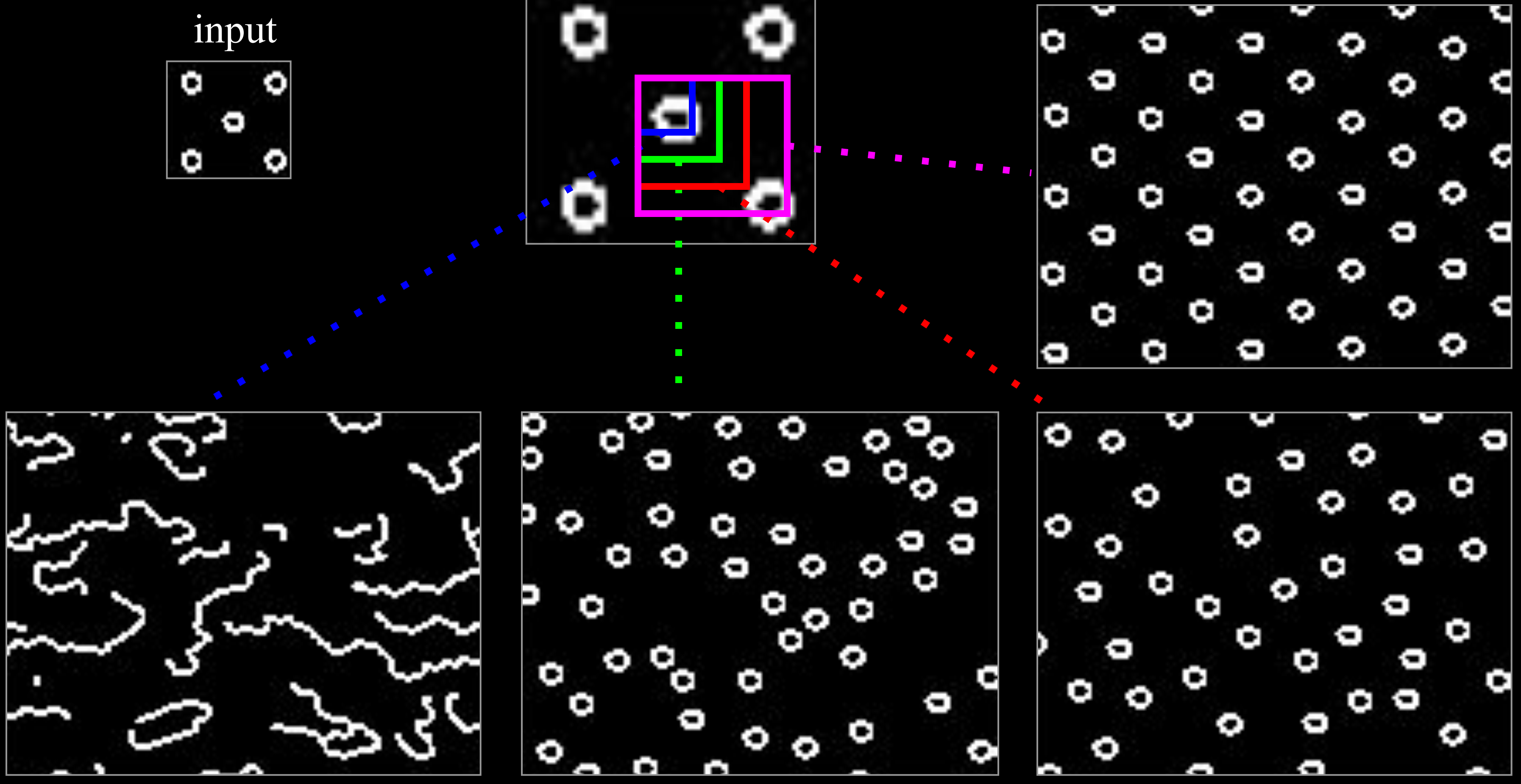


Efros & Leung Algorithm

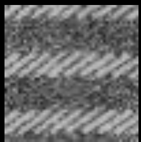


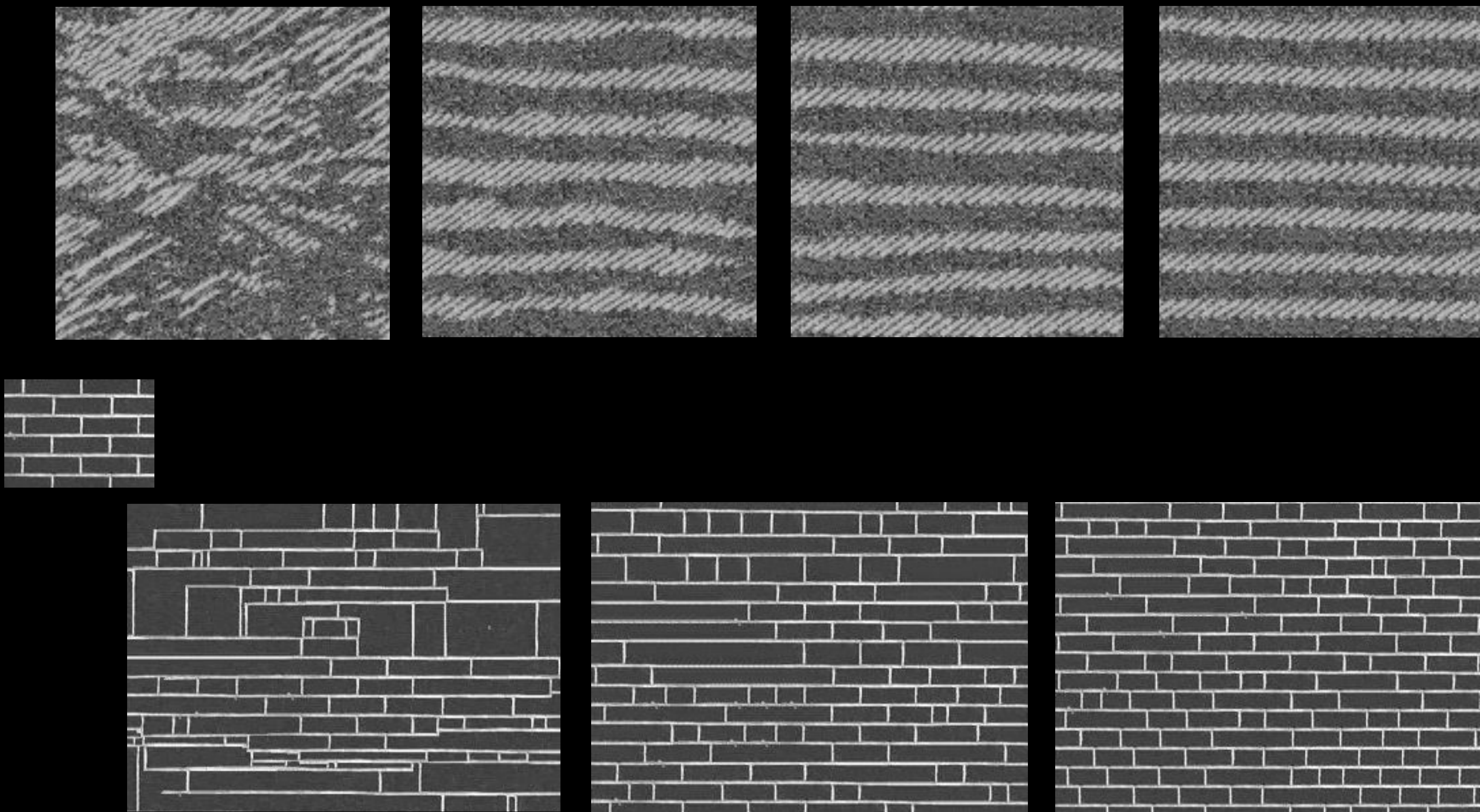
- Assuming Markov property, compute P(**p**|N(**p**))
  - Building explicit probability tables infeasible
  - Instead, we *search the input image* for all similar neighborhoods — that's our pdf for **p**
  - To sample from this pdf, just pick one match at random

Neighborhood Window



Varying Window Size

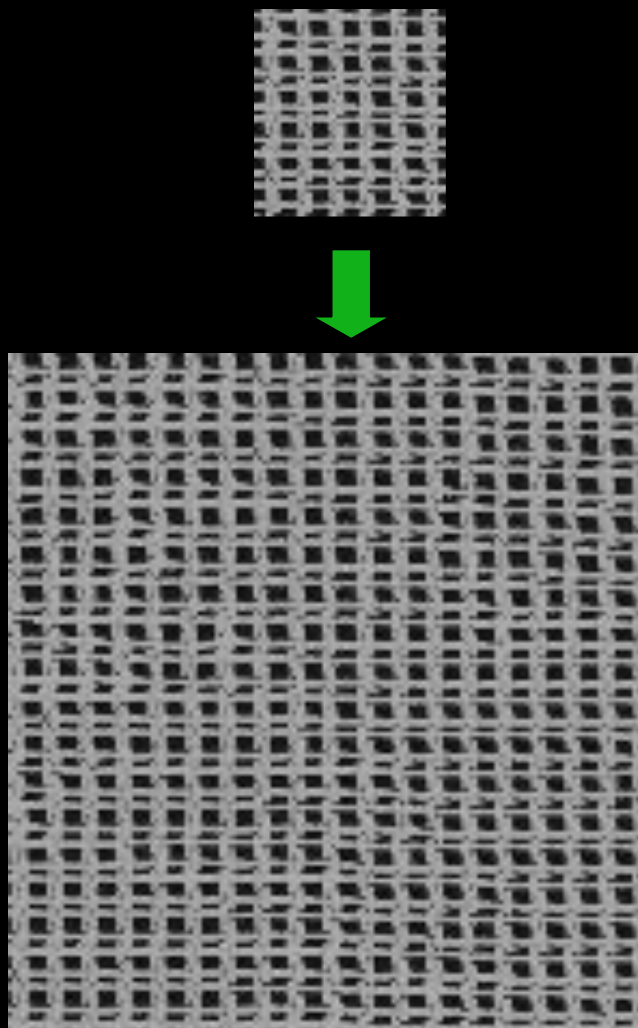




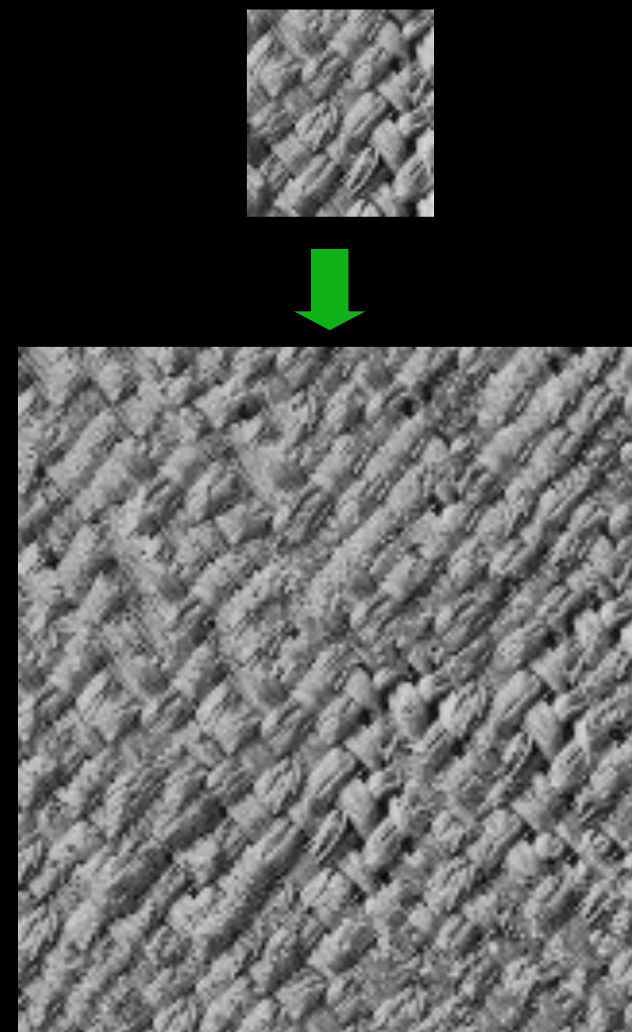
Increasing window size

Synthesis Results

french canvas



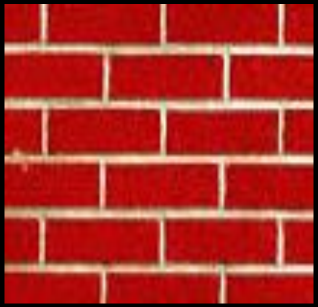
rafia weave

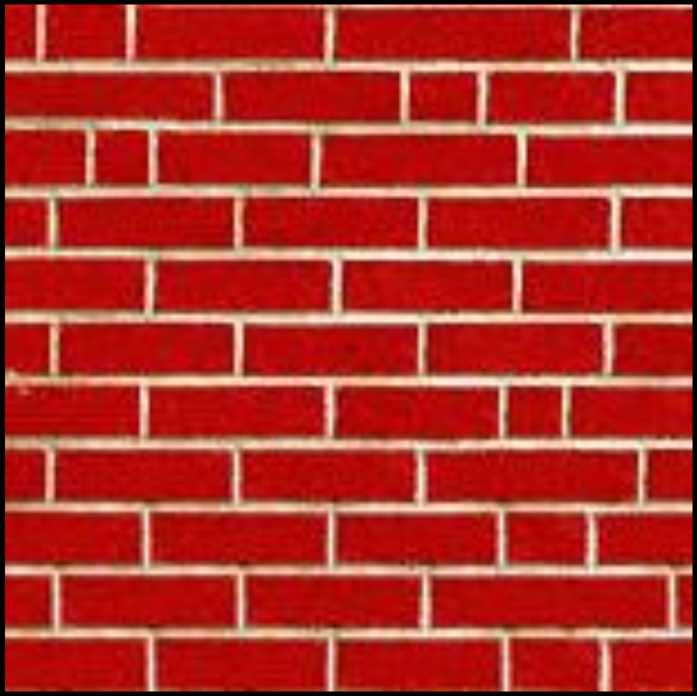


More Results

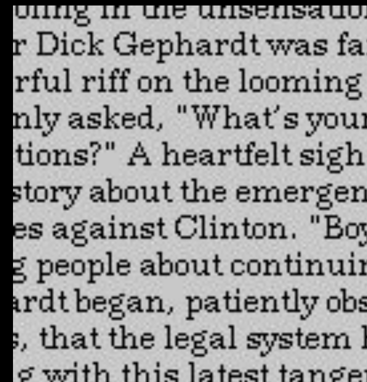
white bread

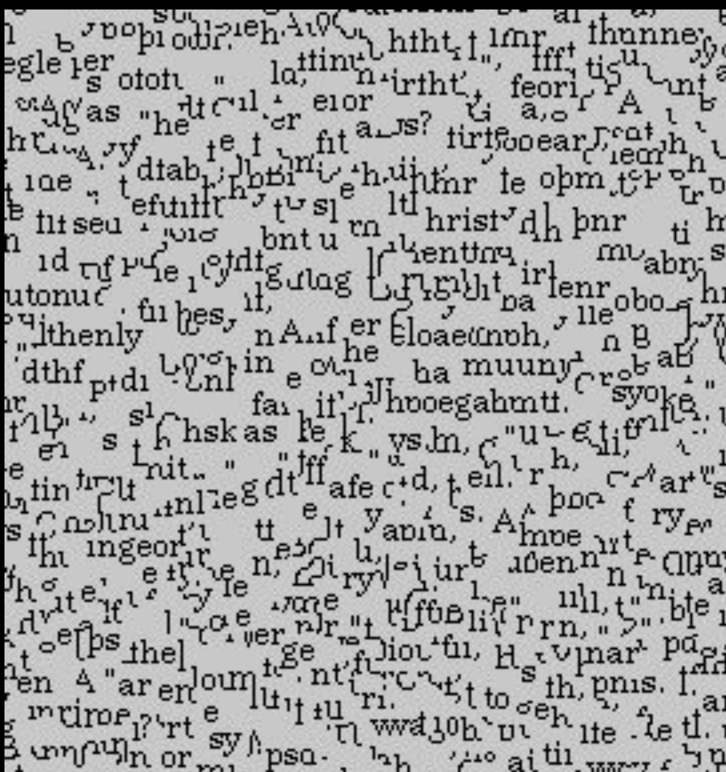
brick wall

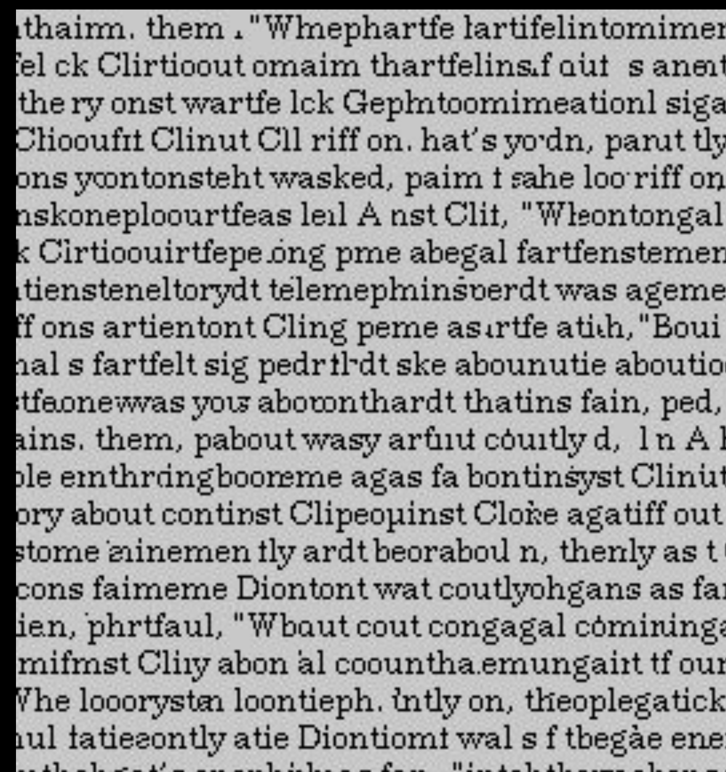




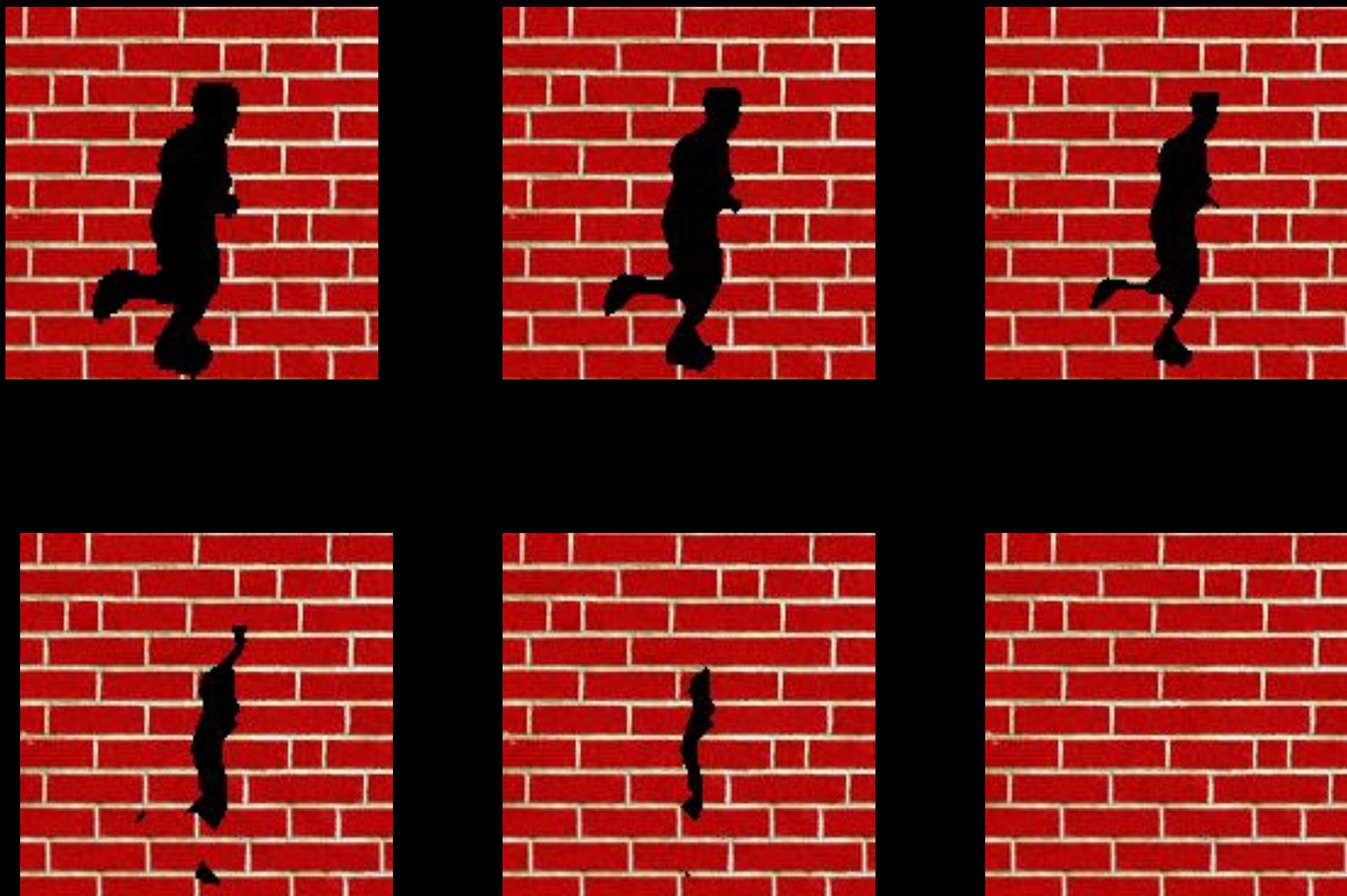
Homage to Shannon







Hole Filling

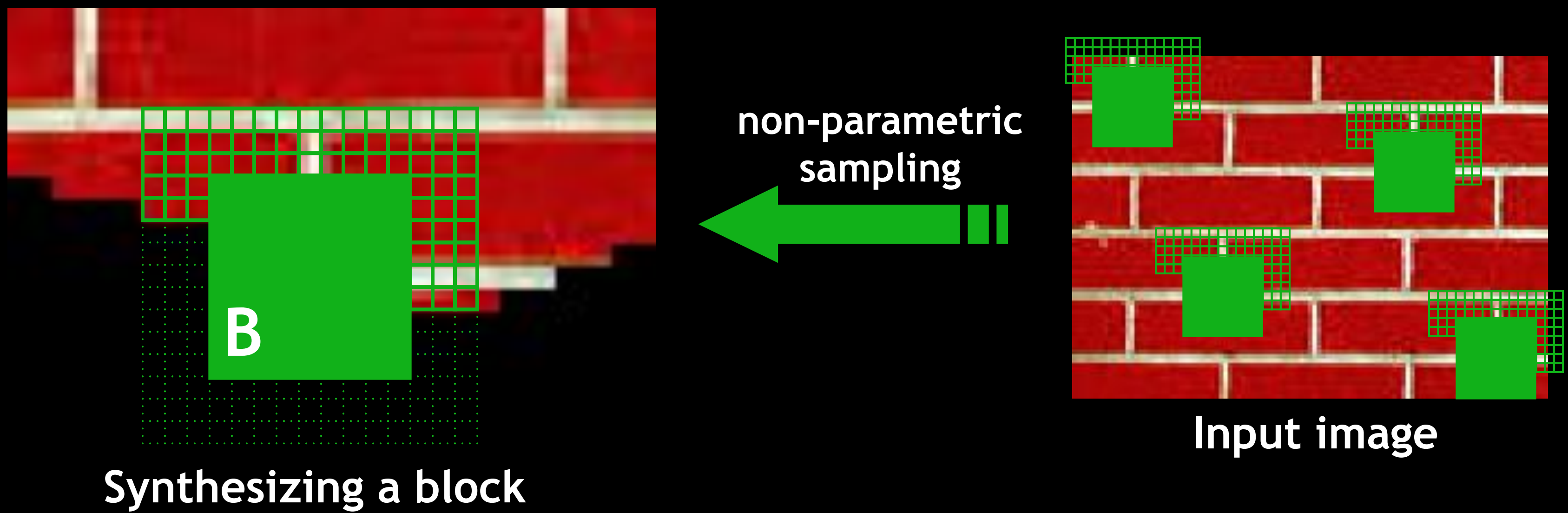


Extrapolation

Summary

- The Efros & Leung algorithm
  - Very simple
  - Surprisingly good results
  - …but very slow

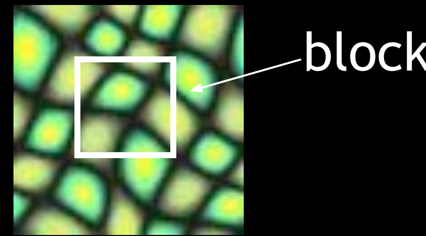
Image Quilting [Efros & Freeman]



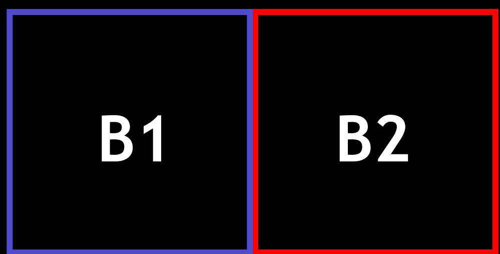
- Observation: neighbor pixels are highly correlated

Idea: unit of synthesis = block

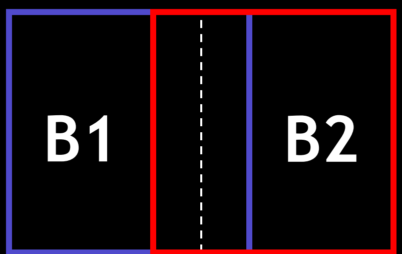
- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once



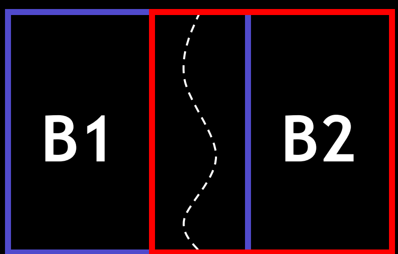
Input texture



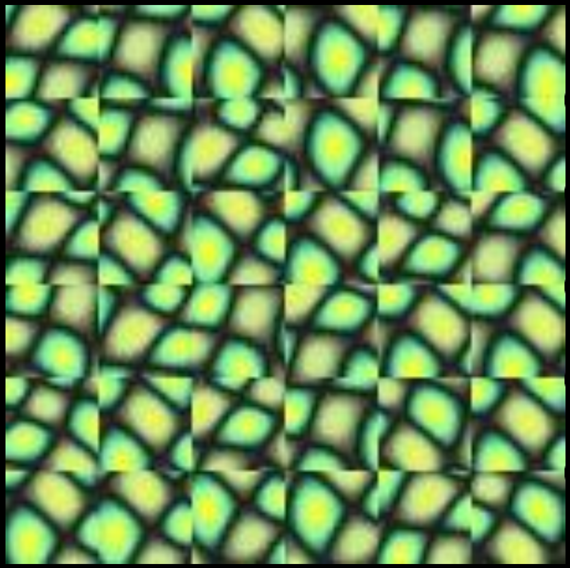
Random placement of blocks

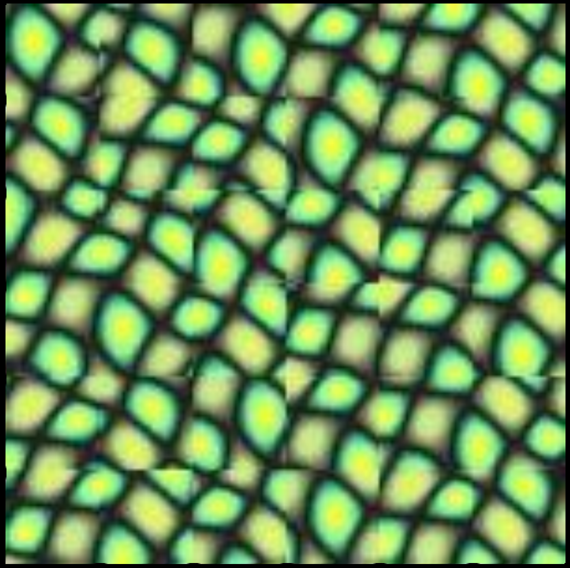


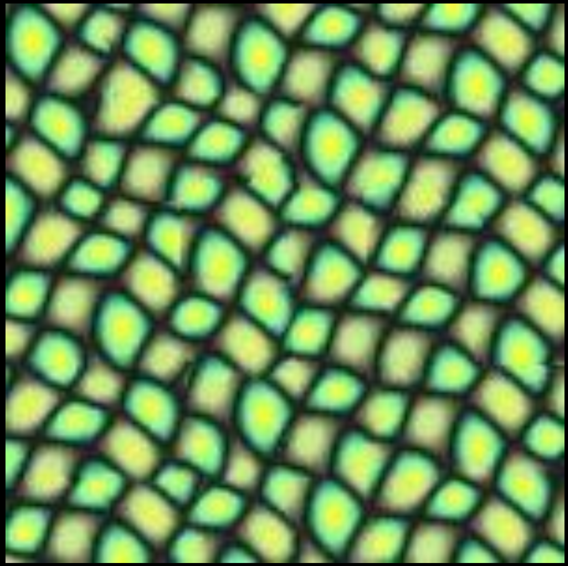
Neighboring blocks constrained by overlap



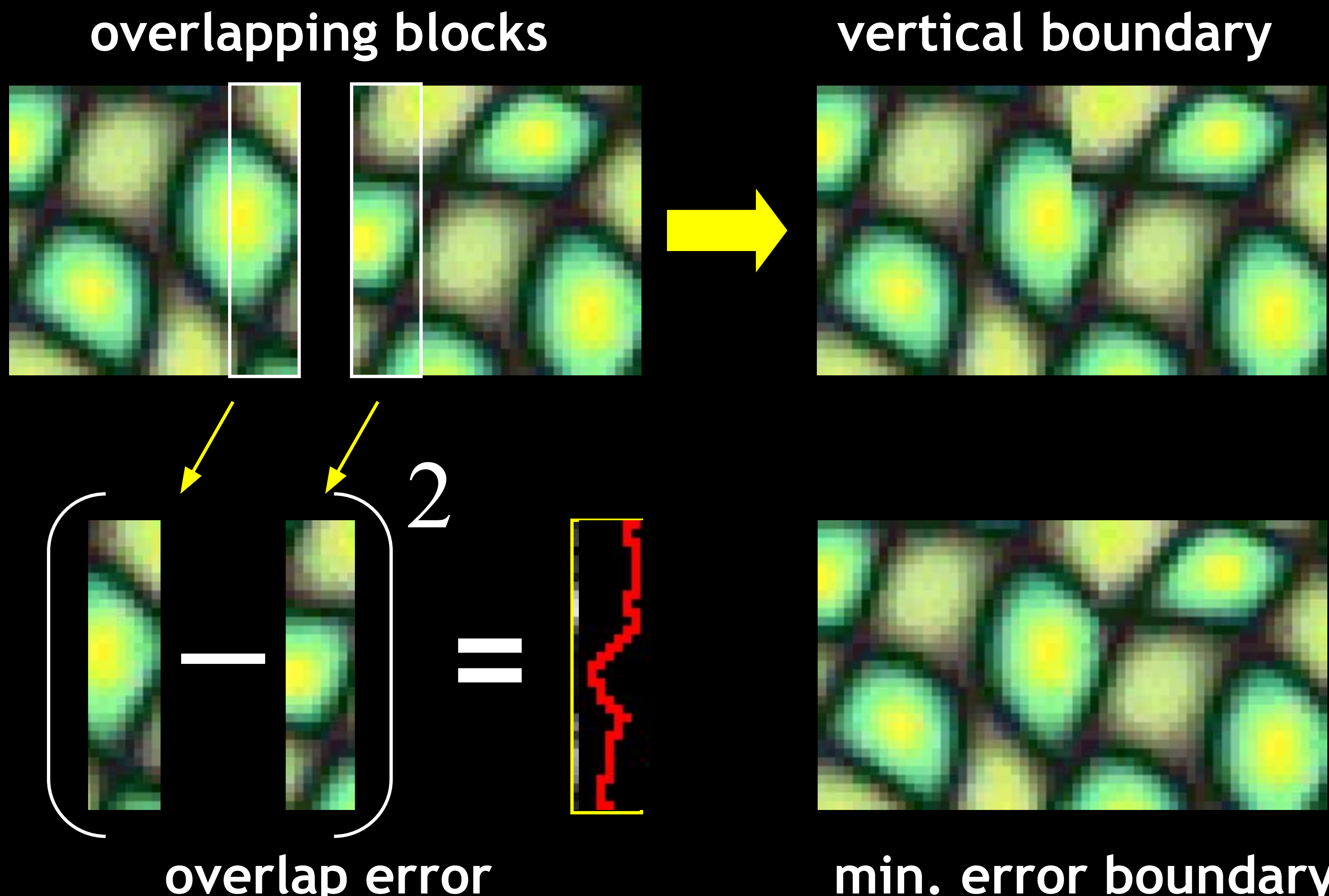
Minimal error boundary cut







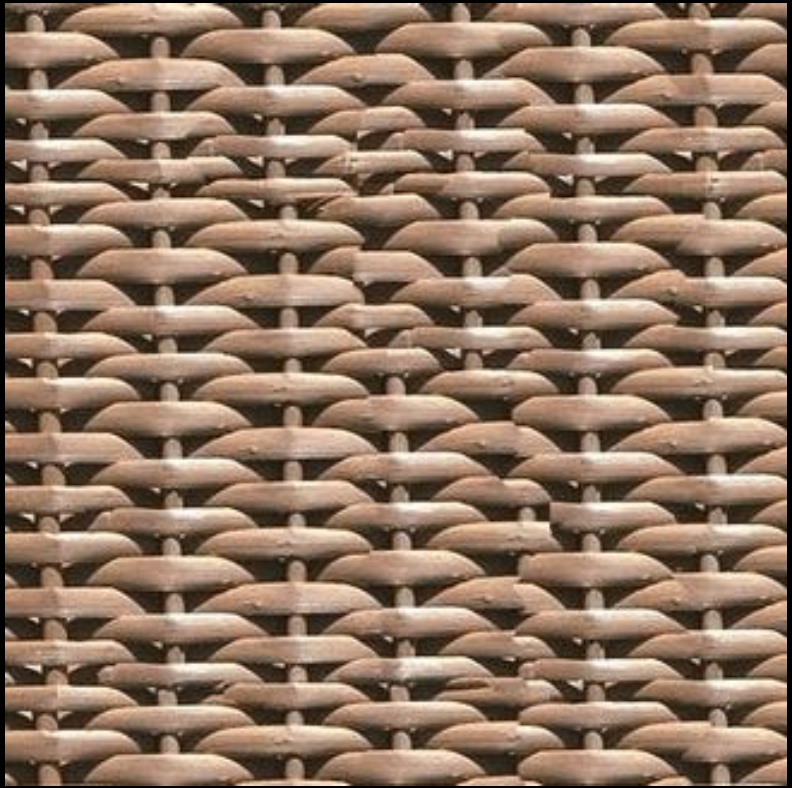
Minimal error boundary

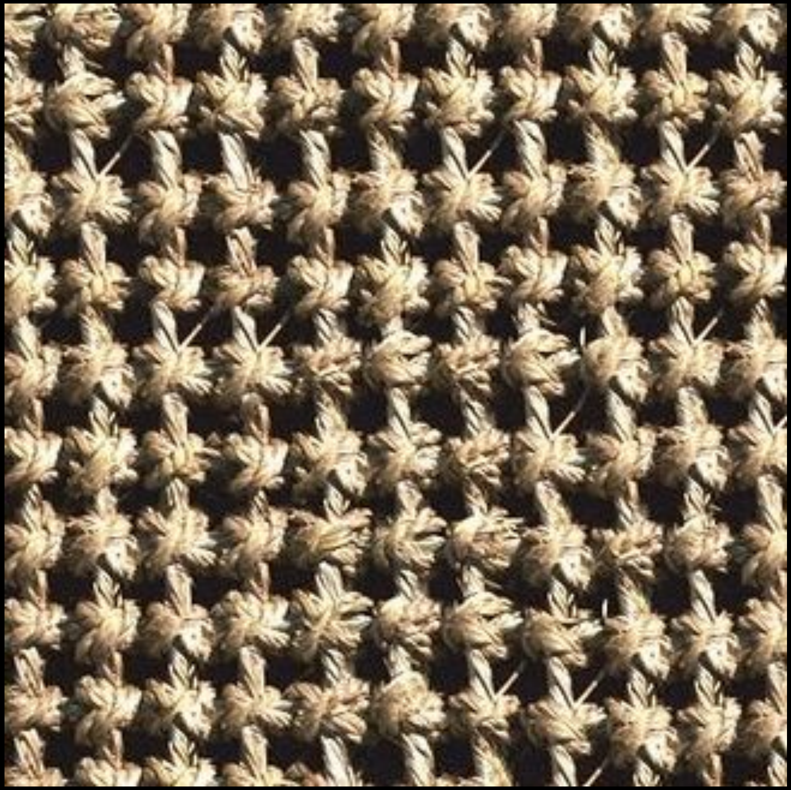


Our Philosophy

- 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


College Inn





input image

Portilla & Simoncelli

Xu, Guo & Shum

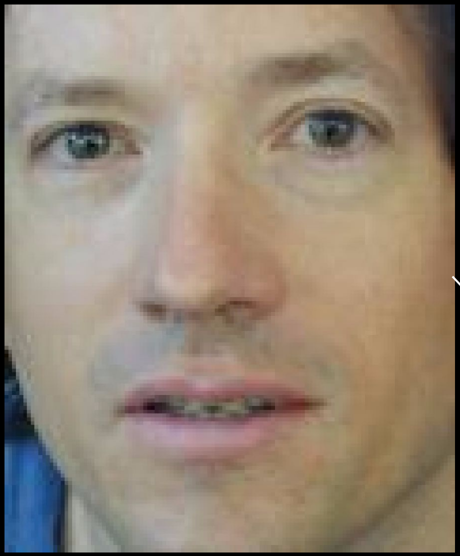
Wei & Levoy

Our algorithm

Application: Texture Transfer

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

Texture Transfer

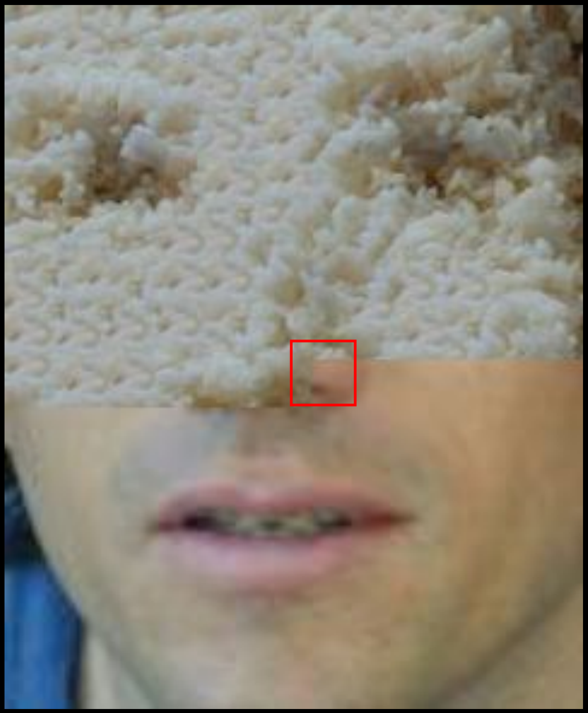


Constraint

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"