Modèles statistiques et fréquentiels pour l'image - Master ID3D

Julie Digne
julie.digne@cnrs.fr

LIRIS - CNRS

04/09/2024

Organisational notes

- The course slides are available on my webpage on the day of the course: http://liris.cnrs.fr/julie.digne/cours/cours_image_stats.html

Organisational notes

- The course slides are available on my webpage on the day of the course: `http://liris.cnrs.fr/julie.digne/cours/cours_image_stats.html`
- Practical work ("TP"): in python with numpy and pillow.

Course schedule

- September 4th: Markovian model and texture synthesis.
- September 9th: Classification Methods and dimensionality reduction. (+TP)
- September 11th: Choosing a Model, Regression Problems, Considerations on Norms.
- September 16th: Image histograms and histogram specification, half-toning (+TP)
- September 18th: Patch-based image processing and editing (+TP)
- Exam on November 6th (to be confirmed)

Project and evaluation

The assignment is available on my webpage. Evaluation will be done through one-to-one interviews.

- September 9th - 1h30 session
- September 16th - 1h30 session
- September 18th - 2h session

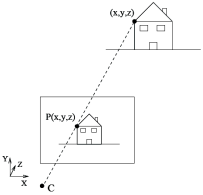
52	49	52	55	55	54	57	58	60	60
52	52	55	55	57	57	60	60	58	62
55	55	56	57	60	60	59	61	62	62
55	57	60	60	60	62	62	62	62	65
56	60	59	60	62	62	65	65	65	62
57	60	62	62	62	65	67	67	67	67
59	62	65	62	65	65	67	70	67	67
59	65	65	65	65	67	70	67	67	67
65	65	65	65	67	67	67	67	70	70
65	65	65	67	70	67	70	70	75	72
104	100	103	106	106	105	108	112	111	111
103	103	106	106	108	108	111	111	112	113
106	106	108	108	111	111	112	113	113	113
107	108	112	112	111	113	113	113	113	116
109	112	112	111	113	113	116	116	116	113
109	112	113	114	113	116	118	118	118	118
111	114	116	113	116	116	118	118	118	118
111	117	116	117	116	118	116	118	118	118
117	116	116	116	118	118	118	118	121	121
117	116	116	118	121	118	121	121	121	123
144	147	150	153	149	152	155	158	158	158
150	150	151	153	151	155	158	158	158	160
153	153	155	155	158	158	156	160	160	160
147	155	152	152	156	160	160	160	160	163
149	152	156	156	160	160	163	163	163	160
149	152	158	154	156	163	165	165	165	165
151	154	161	160	161	163	165	166	165	165
151	157	161	157	163	165	165	165	165	165
157	159	163	163	165	165	165	165	168	168
157	163	163	165	168	165	168	168	170	170

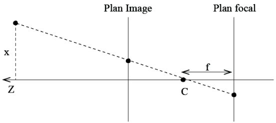
52	49	52	55	55	54	57	58	60	60
52	52	55	55	57	57	60	60	58	62
55	55	56	57	60	60	59	61	62	62
55	57	60	60	60	62	62	62	62	65
56	60	59	60	62	62	65	65	65	62
57	60	62	62	62	65	67	67	67	67
59	62	65	62	65	65	67	70	67	67
59	65	65	65	65	67	70	67	67	67
65	65	65	65	67	67	67	67	70	70
65	65	65	67	70	67	70	70	75	72
104	100	103	106	106	105	108	112	111	111
103	103	106	106	108	108	111	111	112	113
106	106	108	108	111	111	112	113	113	113
107	108	112	112	111	113	113	113	113	116
109	112	112	111	113	113	116	116	116	113
109	112	113	114	113	116	118	118	118	118
111	114	116	113	116	116	118	118	118	118
111	117	116	117	116	118	116	118	118	118
117	116	116	116	118	118	118	118	121	121
117	116	116	118	121	118	121	121	121	123
144	147	150	153	149	152	155	158	158	158
150	150	151	153	151	155	158	158	158	160
153	153	155	155	158	158	156	160	160	160
147	155	152	152	156	160	160	160	160	163
149	152	156	156	160	160	163	163	163	160
149	152	158	154	156	163	165	165	165	165
151	154	161	160	161	163	165	166	165	165
151	157	161	157	163	165	165	165	165	165
157	159	163	163	165	165	165	165	168	168
157	163	163	165	168	165	168	168	170	170

"The Eiffel tower" "A blue sky" …

Acquisition of digital images

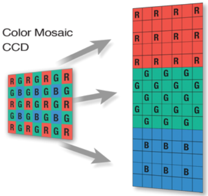
- Projection of a 3D scene on a 2D plane
- Numerically: only a table of numbers
- Black and white: $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$; $I(x, y) = i$
- Color: $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$; $I(x, y) = (r, g, b)$





Acquisition of digital images

- CCD Matrix (*Charged Coupled Device*): integrates the quantity of photons arriving at each cell
- Each pixel integrates a given color.



Bayer Pattern - demosaicking

What we'll see in this course

- Model an image as a *distribution* of colors
- Detect objects by *model regression* (least squares...)
- *Classify* objects by their similarities
- *Compare* textures, images

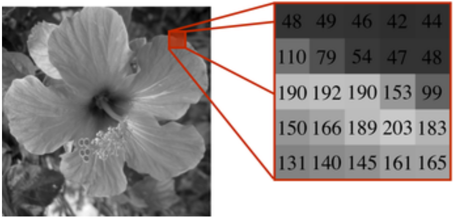
Plan

Grayscale image

Each pixel encodes a light intensity.

For an 8-bits image, a pixel can take 256 integer values ($0 \leq I(p) \leq et255$).

0 encodes a **black** pixel, 128 encodes a **gray** pixel and 255 for a white pixel.



Color Image

- A table (matrix) in which all pixels are triplets (R, V, B) corresponding to the color decomposition on the three primary colors red, green and blue.
- $(0, 0, 255)$ blue, $(0, 255, 0)$ green, $(255, 0, 0)$ red.

A color image

Color Image

- A table (matrix) in which all pixels are triplets (R, V, B) corresponding to the color decomposition on the three primary colors red, green and blue.
- $(0, 0, 255)$ blue, $(0, 255, 0)$ green, $(255, 0, 0)$ red.

Red channel

Color Image

- A table (matrix) in which all pixels are triplets (R, V, B) corresponding to the color decomposition on the three primary colors red, green and blue.
- $(0, 0, 255)$ blue, $(0, 255, 0)$ green, $(255, 0, 0)$ red.

green channel

Color Image

- A table (matrix) in which all pixels are triplets (R, V, B) corresponding to the color decomposition on the three primary colors red, green and blue.
- $(0, 0, 255)$ blue, $(0, 255, 0)$ green, $(255, 0, 0)$ red.

blue channel

The three channels are highly correlated..

From color values to gray scale values

Compute the image luminance using the channel values:

$$L = \frac{R + V + B}{3}$$

From color values to gray scale values

Compute the image luminance using the channel values:

$$L = \frac{R + V + B}{3}$$

Other (better?) color representations

HSV colorspace

- H indicates the color hue (red, yellow, green)
- Saturation S expresses the fact that the color is more or less pure
- Lightness value V indicates the luminosity of a pixel

Other (better?) color representations

HSV colorspace

- H indicates the color hue (red, yellow, green)
- Saturation S expresses the fact that the color is more or less pure
- Lightness value V indicates the luminosity of a pixel

Other (better?) color representations

HSV colorspace

- H indicates the color hue (red, yellow, green)
- Saturation S expresses the fact that the color is more or less pure
- Lightness value V indicates the luminosity of a pixel

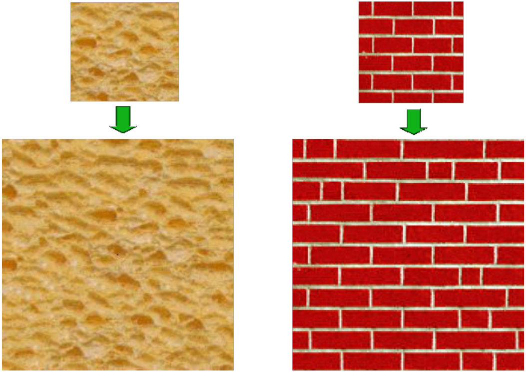
Other (better?) color representations

HSV colorspace

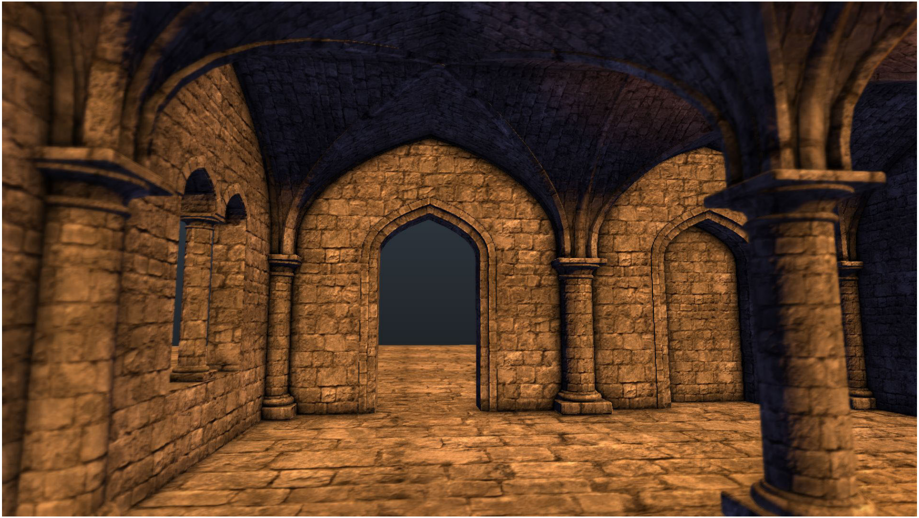
- H indicates the color hue (red, yellow, green)
- Saturation S expresses the fact that the color is more or less pure
- Lightness value V indicates the luminosity of a pixel

Plan

Texture Synthesis

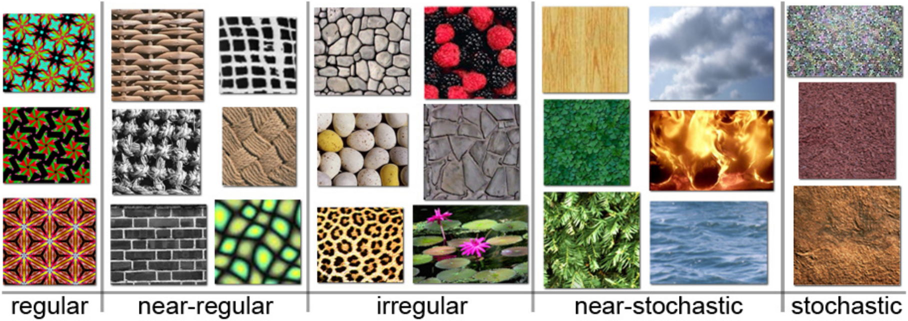


Goal

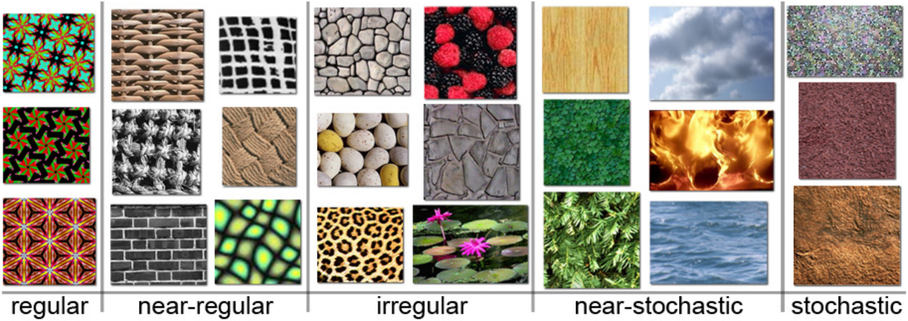


Copyright http://www.castlebuilder3d.com/

Textures are difficult



Textures are difficult



- Copy-pasting an image patch would work for regular textures but not for stochastic textures.
- Drawing pixel values from a probability distribution would work for stochastic textures but not for regular ones.

Plan

To begin with: Markov Chains

An example

Predicting the weather as “sunny”, “cloudy” or “rainy” for each day. The simplest approximation is to assume that the weather on day i only depends on the weather on day $i-1$.

To begin with: Markov Chains

An example

Predicting the weather as "sunny", "cloudy" or "rainy" for each day. The simplest approximation is to assume that the weather on day i only depends on the weather on day $i-1$.

Order

This is a first order Markov Chain

Wheather Markov Chain

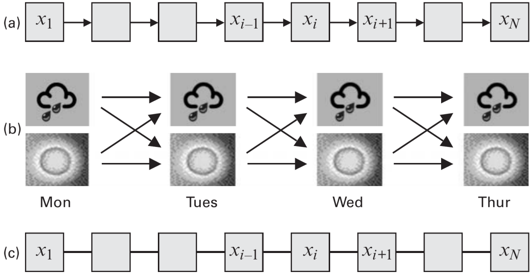


Image from Blake et al. 2011

Transition probabilities

Transition probabilities

The transition probability matrix between two states is

$$P(X_i = a | X_{i-1} = b) = M_i(a, b)$$

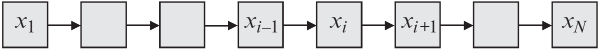
The Markov Chain is said to be stationary if the transition probabilities are independent of i, *i.e.*

$$M_i(a, b) = M_{i-1}(a, b) = M(a, b)$$

- One can devise Markov chains with higher orders (dependencies on $i-1$, $i-2$...): In that case, the value of X_i depends on a limited number of previous states $X_{i-1}, \cdots, X_{i-n}$.

Markov chains as graphs

- The chain can be represented as a graph:
  - Each node corresponds to one X_i
  - An edge exists between X_i and X_{i-1} to model $P(X_i|X_{i-1})$.

(a) 

[Blake 2011]

A graph corresponding to a markov chain of order 1

Markov Random Fields

Definition

A Markov Random Field (MRF) is defined as a probabilistic model over an undirected graph $(\mathcal{V}, \mathcal{E})$

$$P(x_i|(x_j)_{i \neq j}) = P(x_i|\{x_j|(i,j) \in \mathcal{E}\})$$

Markov Random Fields

Definition

A Markov Random Field (MRF) is defined as a probabilistic model over an undirected graph $(\mathcal{V}, \mathcal{E})$

$$P(x_i | (x_j)_{i \neq j}) = P(x_i | \{x_j | (i,j) \in \mathcal{E}\})$$

- Consequence: $P((x_i)_i) = \prod_{(i,j) \in \mathcal{E}} F_{i,j}(x_i, x_j)$

Modeling an image as a graph

Graph of an image

A graph can model the relationship between each pixel (or super-pixel) and its neighbors.

Modeling an image as a graph

Graph of an image

A graph can model the relationship between each pixel (or super-pixel) and its neighbors.

Markov Random Fields on graphs: Local Markov Property

The random variable at a node depends solely of the random variables in its neighborhood (Markov blanket).

Modeling an image as a graph

Graph of an image

A graph can model the relationship between each pixel (or super-pixel) and its neighbors.

Markov Random Fields on graphs: Local Markov Property

The random variable at a node depends solely of the random variables in its neighborhood (Markov blanket).

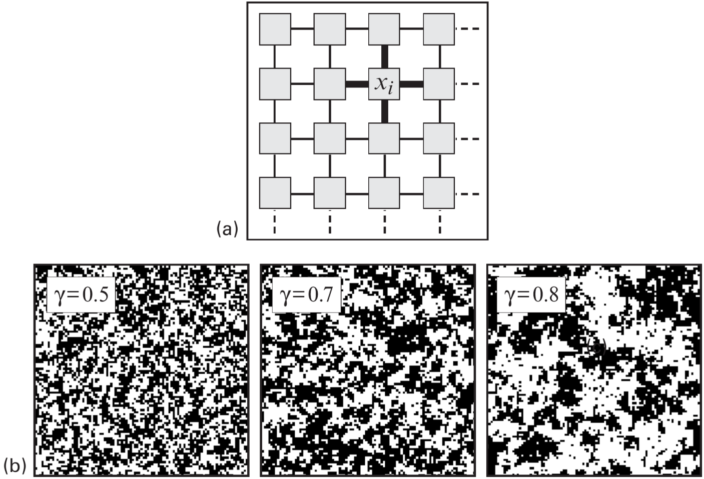
- From now on we will assume that the distribution is positive (in that case local Markov property $\Leftrightarrow$ global Markov property)

Energy

Energy

Since P is a positive distribution, we can rather rely on an energy $E(x)$ such that $P(x) = \prod_{c \in \mathcal{C}} \phi(x) = \exp(-E(x))$ and $E(x) = \sum_{c \in \mathcal{C}} \Phi_c(x_c)$.

Ising Model



[Blake 2011]

Ising Model

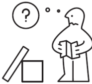
Ising Model

Each variable X_i takes values in $\{0, 1\}$. The cliques have size 2 (two neighboring pixels).

$$\phi_{ij} = \gamma |x_i - x_j|$$

where γ is a parameter of our method. The total energy is

$$E(x) = \sum_{(i,j) \in \mathcal{E}} \gamma |x_i - x_j|$$

- What does this γ model ?

Ising Model

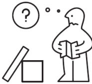
Ising Model

Each variable X_i takes values in $\{0, 1\}$. The cliques have size 2 (two neighboring pixels).

$$\phi_{ij} = \gamma |x_i - x_j|$$

where γ is a parameter of our method. The total energy is

$$E(x) = \sum_{(i,j) \in \mathcal{E}} \gamma |x_i - x_j|$$

- What does this γ model ? When two neighboring variables have different values the energy increases of amount γ

Gibbs Sampling

Sampling from a MRF

At each visit to a site i, x_i is sampled from the local conditional distribution $P(x_i|x_j, (i,j) \in \mathcal{E}, j \neq i)$. Start with random values for the pixels and traverse all sites in random order until convergence.

- local conditional distribution:

$$p = \frac{P(X_i = 1)}{P(X_i = 0)} = \frac{\exp -\gamma E(x_0, \cdots, X_i = 1, \cdots, x_n)}{\exp -\gamma E(x_0, \cdots, X_i = 0, \cdots, x_n)} = \exp -\gamma \Delta E$$

with ΔE the difference of energy between the two states.

Gibbs Sampling

Sampling from a MRF

At each visit to a site i, x_i is sampled from the local conditional distribution $P(x_i | x_j, (i,j) \in \mathcal{E}, j \neq i)$. Start with random values for the pixels and traverse all sites in random order until convergence.

- local conditional distribution:

$$p = \frac{P(X_i = 1)}{P(X_i = 0)} = \frac{\exp -\gamma E(x_0, \cdots, X_i = 1, \cdots, x_n)}{\exp -\gamma E(x_0, \cdots, X_i = 0, \cdots, x_n)} = \exp -\gamma \Delta E$$

  with ΔE the difference of energy between the two states.

- The process converges

Gibbs Sampling

Sampling from a MRF

At each visit to a site i, x_i is sampled from the local conditional distribution $P(x_i | x_j, (i,j) \in \mathcal{E}, j \neq i)$. Start with random values for the pixels and traverse all sites in random order until convergence.

- local conditional distribution:

$$p = \frac{P(X_i = 1)}{P(X_i = 0)} = \frac{\exp -\gamma E(x_0, \cdots, X_i = 1, \cdots, x_n)}{\exp -\gamma E(x_0, \cdots, X_i = 0, \cdots, x_n)} = \exp -\gamma \Delta E$$

  with ΔE the difference of energy between the two states.
- The process converges
- ... But can be extremely slow.

Plan

Texture synthesis as a MRF Problem

- A texture is modeled by a Markov Random Field

[Efros Leung 1999], **[Wei Levoy 2001]**

Texture synthesis as a MRF Problem

- A texture is modeled by a Markov Random Field
- Each pixel value depends on the pixel values of its neighbors

[Efros Leung 1999], [Wei Levoy 2001]

Texture synthesis as a MRF Problem

- A texture is modeled by a Markov Random Field
- Each pixel value depends on the pixel values of its neighbors
- The size of the neighborhood encodes how stochastic the texture is.

[Efros Leung 1999], **[Wei Levoy 2001]**

Synthesizing one pixel [Efros Leung 1999]

Synthesis

Let I_{smp} be the texture sample image, I_{real} be the infinite texture $I_{smp} \subset I_{real}$ and I the image being synthesized. Assume all pixels are known except p. Let $w(p)$ be its neighborhood, then:

$$P(I(p) = a|I) = P(I(p) = a|w(p))$$

- Let $d(w_1, w_2)$ be a distance between two patches (usually SSD of $SSD * G$)

Synthesizing one pixel [Efros Leung 1999]

Synthesis

Let I_{smp} be the texture sample image, I_{real} be the infinite texture $I_{smp} \subset I_{real}$ and I the image being synthesized. Assume all pixels are known except p. Let $w(p)$ be its neighborhood, then:

$$P(I(p) = a | I) = P(I(p) = a | w(p))$$

- Let $d(w_1, w_2)$ be a distance between two patches (usually SSD of $SSD * G$)
- $\Omega(p) = \{w \in I_{real} | d(w, w(p)) = 0\}$

Synthesizing one pixel [Efros Leung 1999]

Synthesis

Let I_{smp} be the texture sample image, I_{real} be the infinite texture $I_{smp} \subset I_{real}$ and I the image being synthesized. Assume all pixels are known except p. Let $w(p)$ be its neighborhood, then:

$$P(I(p) = a|I) = P(I(p) = a|w(p))$$

- Let $d(w_1, w_2)$ be a distance between two patches (usually SSD of $SSD * G$)
- $\Omega(p) = \{w \in I_{real} | d(w, w(p)) = 0\}$
- I_{real} is unavailable

Synthesizing one pixel

Heuristic

Replace $\Omega(p)$ by $\Omega'(p)$ containing patches that are close to $w(p)$. Let $w_{best} = argmin_{w \in I_{smp}} d(w, w(p))$ then:

$$\Omega'(p) = \{w \in I_{smp} | d(w, w(p)) \leq (1+\varepsilon) d(w_{best}, w(p))\}$$

Synthesizing one pixel

Heuristic

Replace $\Omega(p)$ by $\Omega'(p)$ containing patches that are close to $w(p)$. Let $w_{best} = argmin_{w \in I_{smp}} d(w, w(p))$ then:

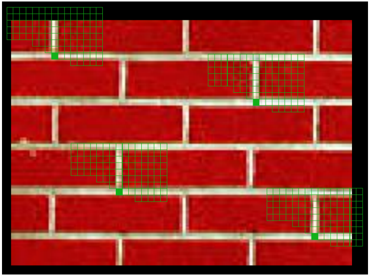
$$\Omega'(p) = \{w \in I_{smp} | d(w, w(p)) \leq (1+\varepsilon) d(w_{best}, w(p))\}$$

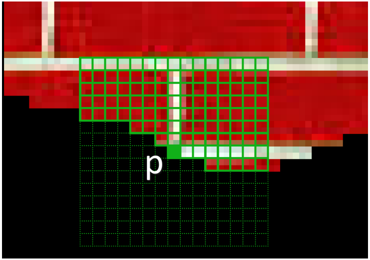
Finally

p is taken to be the average of the values of all center pixels in $\Omega'(p)$ (variation: the value of one uniformly drawn patch in $\Omega'(p)$.

Texture Synthesis Algorithm

- Assuming Markov property, compute $P(p|N(p))$
  - Search the input image for all similar neighborhoods $\Omega'(p)$
  - Pick one match at random





Texture synthesis Algorithm

1. Start from a $\ell \times \ell$ seed from the input

Texture synthesis Algorithm

1. Start from a $\ell \times \ell$ seed from the input
2. The new pixel p to fill is randomly picked among the ones that have the larger number of *filled* neighbors in their neighborhood.

Partial distance

Let $w(p) \in I$ and $w'(p') \in I_{smp}$ be two neighborhoods partially filled. $\tilde{N}$ is the set of all v such that $I(p+v)$ and $I_{smp}(p'+v)$ are defined. The distance between w and w' is

$$d(w, w') = \frac{\sum_{v \in \tilde{N}} \|I(p+v) - I_{smp}(p'+v)\|_2^2 G_\sigma(\|v\|)}{\sum_{v \in \tilde{N}} G_\sigma(\|v\|)}$$

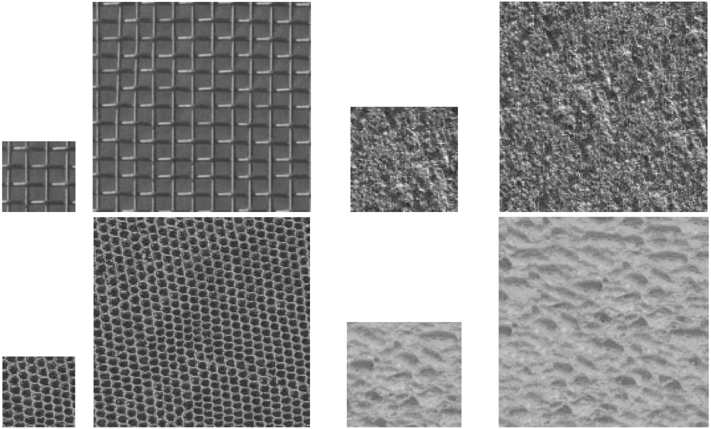
with G_σ a centered Gaussian with standard deviation σ.

Full Algorithm [Efros Leung 1999]

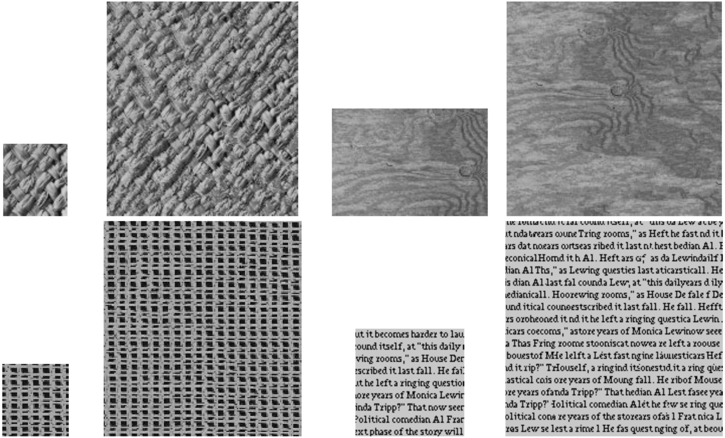
Input: image I_{smp}, output size, neighborhood size,

1. Initialize with $\ell \times \ell$ random seed of I_{smp}
2. While output I not filled
    1. Pick a pixel p not yet filled with a maximal number of filled neighbors
    2. Compute the distance of $w(p)$ to all patches of input I_{smp}
    3. Build $\Omega'(p)$
    4. Pick randomly one of the similar neighborhood $w(p')$ in $\Omega'(p)$
    5. Set $I(p) = I_{smp}(p')$ (= Fill p with central value)

Results

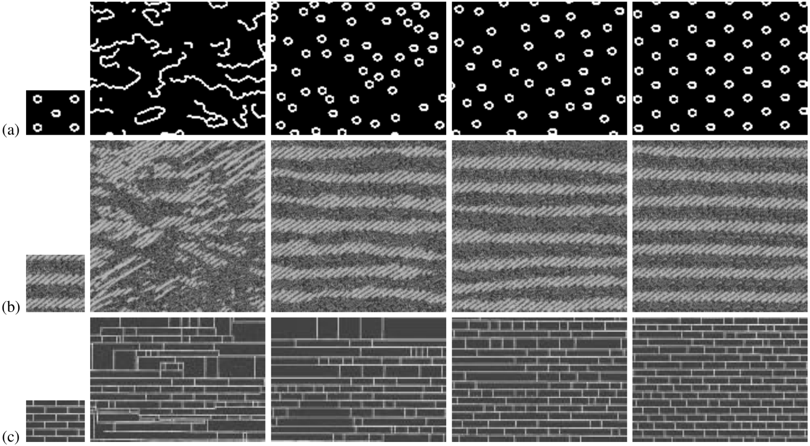


Results



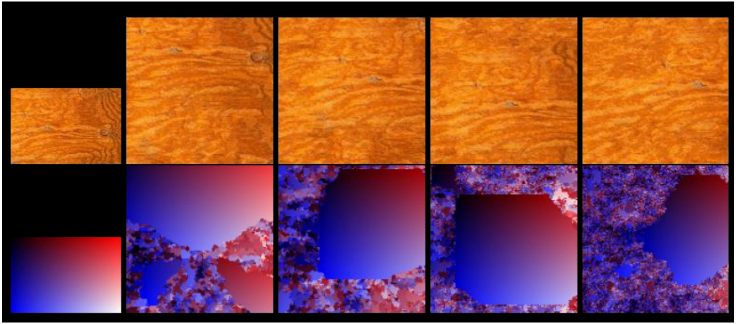
ut it becomes harder to lau
ound itself, at "this daily
ving rooms," as House Der
scribed it last fall. He fai
it he left a ringing questio
ore years of Monica Lewir
inda Tripp?" That now seer
Political comedian Al Frar
ext phase of the story will

Varying neighborhood size



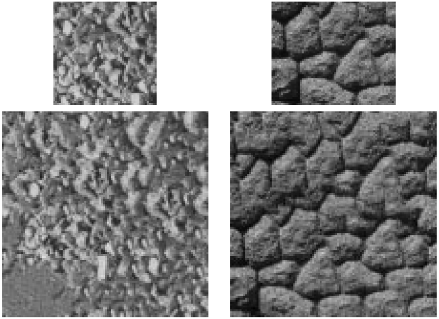
Patch sizes: $5, 11, 15, 23$

Varying epsilon ε

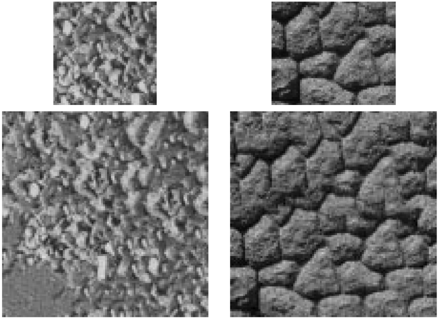


$$\varepsilon = 0.05, 0.1, 0.2, 0.5$$

Failure Cases

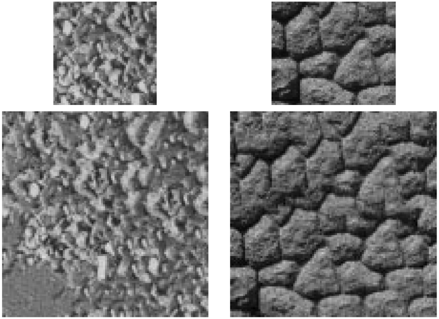


Failure Cases



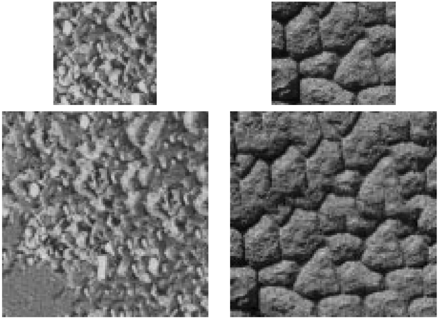
- Growing garbages

Failure Cases



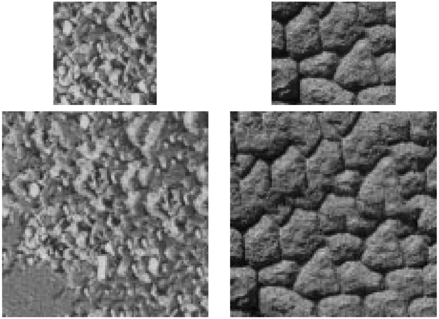
- Growing garbages (ε too large)

Failure Cases



- Growing garbages (ε too large)
- Verbatim Copy

Failure Cases

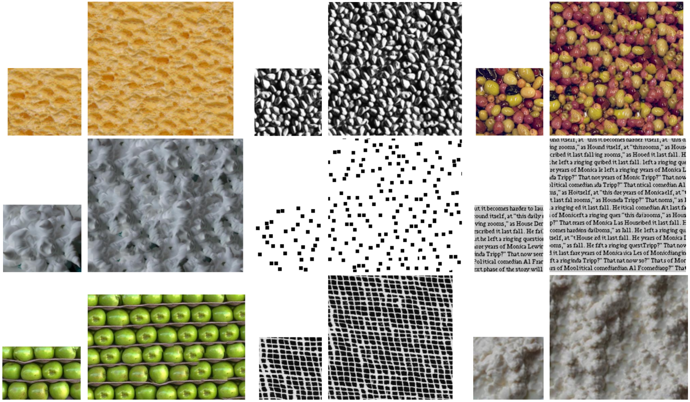


- Growing garbages (ε too large)
- Verbatim Copy (ε too small)

Plan

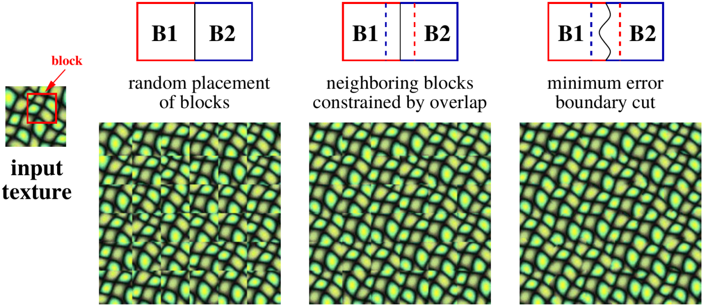
Image Quilting [Efros Freeman 2001]

- Focuses on boundary optimization between neighboring patches



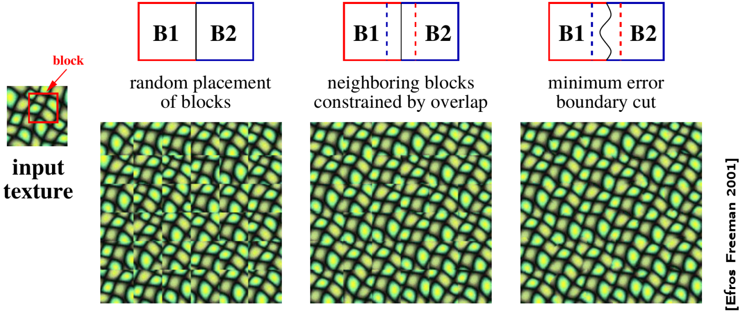
[Efros Freeman 2001]

Patch placement



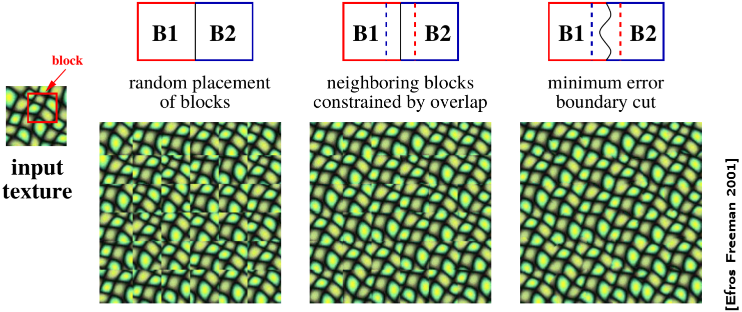
[Efros Freeman 2001]

Patch placement



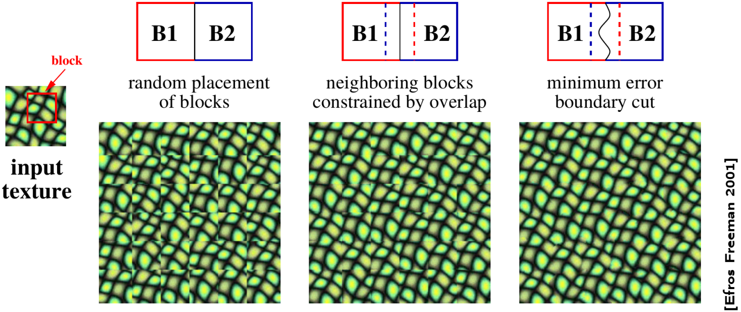
- Paste patches from the sample texture randomly (left)

Patch placement



- Paste patches from the sample texture randomly (left)
- Better: Paste patches from the sample texture randomly among those that fit approximation with their neighbors (middle)

Patch placement



- Paste patches from the sample texture randomly (left)
- Better: Paste patches from the sample texture randomly among those that fit approximation with their neighbors (middle)
- Optimize the boundary so that the patches blend *seamlessly* (right)

Overlapping criterion

- When a set of patches are placed, the next one should fit the set of already pasted patches
- The measure is similar to the partial distance as before.

Overlap error

Let B_1 and B_2 be two overlapping blocks, with overlaps B_1^{ov} and B_2^{ov}. The error image is then $e = \|B_1^{ov} - B_2^{ov}\|_2^2$ (e has the size of the overlap).

Boundary Optimization

- Before adding the patch: look for the optimal boundary cut.

Boundary Optimization

- Before adding the patch: look for the optimal boundary cut.
- A boundary cut is defined as a path of adjacent pixels that separate the two patches

Boundary Optimization

- Before adding the patch: look for the optimal boundary cut.
- A boundary cut is defined as a path of adjacent pixels that separate the two patches
- *Dynamic Programming* is used to look for a path optimally separating the patches

Boundary Optimization

- Before adding the patch: look for the optimal boundary cut.
- A boundary cut is defined as a path of adjacent pixels that separate the two patches
- *Dynamic Programming* is used to look for a path optimally separating the patches

Vertical Boundary Path

For each pixel $(i, j) \in e$ store:

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

When we reach the bottom we can take the minimum and roll back to get the path of pixels.

Boundary Optimization

- Before adding the patch: look for the optimal boundary cut.
- A boundary cut is defined as a path of adjacent pixels that separate the two patches
- *Dynamic Programming* is used to look for a path optimally separating the patches

Vertical Boundary Path

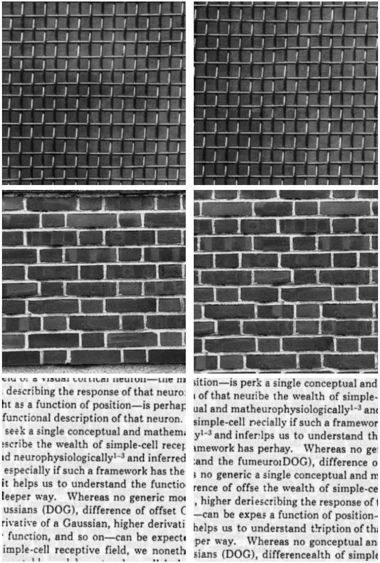
For each pixel $(i, j) \in e$ store:

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

When we reach the bottom we can take the minimum and roll back to get the path of pixels.

[Raad, Galerne 2017]

Results



[Efros Freeman 2001]

Results

[Efros Freeman 2001]

Results

[Efros Freeman 2001]

Plan

Graph Cuts

- Minimize energies by casting the problem as a min-cut problem

Graph Cuts

- Minimize energies by casting the problem as a min-cut problem
- Applications to segmentation, stereo, denoising energies

Problem Statement [Boykov Jolly 2001]

Given an image I, we look for a label A_p for each pixel p of an image. A_p can take values 0 or 1. There are two special sets of pixels O and B containing pixels that we know belong to class O or class B

Segmentation Energy

Segmentation Energy

$$E = \lambda \sum_p R_p(A_p) + \sum_{p,q \text{ neighbors}} B_{p,q}\delta_{p,q}$$

Where

- $R_p(A_p)$ encodes the probability for a pixel p to have label A_p
- $\delta(p,q) = 0$ if $A_p = A_q$ and 1 otherwise.
- $B(p,q)$ is the energy that two pixels have different classes given their color values.

Segmentation Energy

Segmentation Energy

$$E = \lambda \sum_{p} R_p(A_p) + \sum_{p,q \text{ neighbors}} B_{p,q}\delta_{p,q}$$

Where

- $R_p(A_p)$ encodes the probability for a pixel p to have label A_p
- $\delta(p,q) = 0$ if $A_p = A_q$ and 1 otherwise.
- $B(p,q)$ is the energy that two pixels have different classes given their color values.

Segmentation Energy

Segmentation Energy

$$E = \lambda \sum_{p} R_p(A_p) + \sum_{p,q \text{ neighbors}} B_{p,q}\delta_{p,q}$$

Where

- $R_p(A_p)$ encodes the probability for a pixel p to have label A_p
- $\delta(p,q) = 0$ if $A_p = A_q$ and 1 otherwise.
- $B(p,q)$ is the energy that two pixels have different classes given their color values.

- $R_p(bkg)$, $R_p(obj)$ class models

Segmentation Energy

Segmentation Energy

$$E = \lambda \sum_p R_p(A_p) + \sum_{p,q \text{ neighbors}} B_{p,q}\delta_{p,q}$$

Where

- $R_p(A_p)$ encodes the probability for a pixel p to have label A_p
- $\delta(p,q) = 0$ if $A_p = A_q$ and 1 otherwise.
- $B(p,q)$ is the energy that two pixels have different classes given their color values.

- $R_p(bkg)$, $R_p(obj)$ class models
- $B(p,q)$ likelihood for a boundary to cross edge p, q

Graph Construction

Graph Topology

The graph $G = (V, E)$ corresponding to the image is built as follows:

- All pixels are vertices of the graph, two special nodes S and T are also added to V
- Edges are added between nodes corresponding to neighboring pixels in the image
- Edges are also added between all pixels and S and all pixels and T

Graph Construction

Graph Topology

The graph $G = (V, E)$ corresponding to the image is built as follows:

- All pixels are vertices of the graph, two special nodes S and T are also added to V
- Edges are added between nodes corresponding to neighboring pixels in the image
- Edges are also added between all pixels and S and all pixels and T

- A neighborhood should be defined

Graph Construction

Graph Edge Weights

- Edge between pixels p, q: weight $B(p, q)$
- Edge between pixel p and node S: weight $\lambda R_p(bkg)$
- Edge between pixel p and node T: weight $\lambda R_p(obj)$

Graph Construction

Graph Edge Weights

- Edge between pixels p, q: weight $B(p, q)$
- Edge between pixel p and node S: weight $\lambda R_p(bkg)$
- Edge between pixel p and node T: weight $\lambda R_p(obj)$

- $K = 1 + max_p \sum_{q \in \mathcal{N}(p)} B(p, q)$

Graph Construction

Graph Edge Weights

- Edge between pixels p, q: weight $B(p, q)$
- Edge between pixel p and node S: weight $\lambda R_p(bkg)$
- Edge between pixel p and node T: weight $\lambda R_p(obj)$

- $K = 1 + max_p \sum_{q \in \mathcal{N}(p)} B(p, q)$
- If $p \in O$: edge p, S has weight K, edge p, T has weight 0

Graph Construction

Graph Edge Weights

- Edge between pixels p, q: weight $B(p, q)$
- Edge between pixel p and node S: weight $\lambda R_p(bkg)$
- Edge between pixel p and node T: weight $\lambda R_p(obj)$

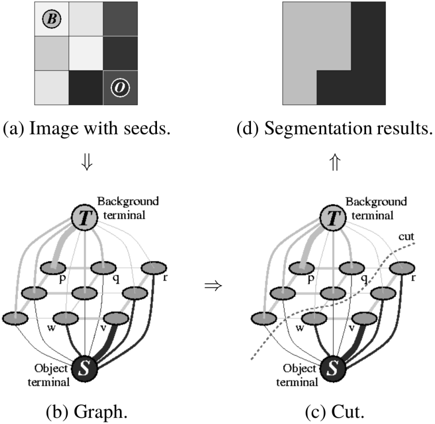
- $K = 1 + max_p \sum_{q \in \mathcal{N}(p)} B(p, q)$
- If $p \in O$: edge p, S has weight K, edge p, T has weight 0
- If $p \in B$: edge p, S has weight 0, edge p, T has weight K

Equivalence to a min cut problem

Energy minimization

The minimium of the segmentation energy is obtained by finding the minimum cost cut on graph G separating S from T

Toy example



[Boykov and Jolly 2001]

- Some hard constraints are created using seeds $p \in O$ or $p \in B$

Proof

Lemma

The minimum cut $\hat{C}$ on graph G is feasible *ie*

- $\hat{C}$ severs exactly one t-link for each p (either $p - S$ or $p - T$)
- if $(p, q) \in \hat{C}$, then one of them is linked to S and the other is linked to T after the cut
- if $p \in O$, then $(p, T) \in \hat{C}$
- if $p \in B$, then $(p, S) \in \hat{C}$

- Proof: Bear in mind that $\hat{C}$ is minimal.

Link with the segmentation

Correspondence

Any feasible cut C corresponds to a segmentation $A(C)$ such that:

$$A_p(C) = \begin{cases} obj \text{ if } (p, T) \in C \\ bkg \text{ if } (p, S) \in C \end{cases}$$

Optimal segmentation

Theorem

Among all segmentation A satisfying $A_p = obj$ if $p \in O$ and $A_p = bkg$ if $p \in B$, the one defined by the minimal cut $\hat{C}$ minimizes the segmentation energy.

Optimal segmentation

Theorem

Among all segmentation A satisfying $A_p = obj$ if $p \in O$ and $A_p = bkg$ if $p \in B$, the one defined by the minimal cut $\hat{C}$ minimizes the segmentation energy.

- Proof: link the energy with the cost of the cut.

What are $R_p(A_p)$ and $B(p, q)$?

Regional term

The regional term is a log-likelihood term

$$R_p(A_p) = \begin{cases} -\log P(I_p|O) \text{ if } A_p = obj \\ -\log P(I_p|B) \text{ if } A_p = bkg \end{cases}$$

What are $R_p(A_p)$ and $B(p,q)$?

Regional term

The regional term is a log-likelihood term

$$R_p(A_p) = \begin{cases} -\log P(I_p|O) \text{ if } A_p = obj \\ -\log P(I_p|B) \text{ if } A_p = bkg \end{cases}$$

Boundary penalty

$$B(p,q) = \frac{1}{dist(p,q)} \exp - \frac{\|I_p - I_q\|_2^2}{2\sigma^2}$$

How do we compute the log-likelihoods?

MRF formulation

The sets O and B give histograms of pixel values for the object and background yielding in turn $P(I_p|O)$ and $P(I_p|B)$

Result

(a) Original image

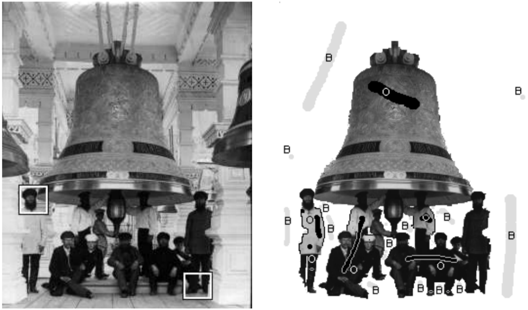
(b) Result for $\lambda = 7$—43

(c) Result for $\lambda = 0$

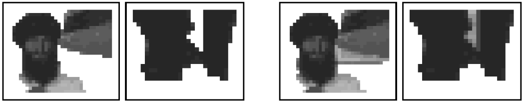
(d) Result for $\lambda = 60$

[Boykov and Jolly 2001]

Result



(a) Original B&W photo (b) Segmentation results



(c) Details of segmentation with regional term (d) Details of segmentation without regional term

[Boykov and Jolly 2001]

Plan

Texture Synthesis using Graph Cuts

Kwatra et al. 2004

Principle

Idea

Copy irregularly shaped patches on the image and arrange the boundaries between the copied patches

Candidate patch selection

- A candidate rectangular patch (or patch offset) is selected by performing a comparison between the candidate patch and the pixels already in the output image.

Candidate patch selection

- A candidate rectangular patch (or patch offset) is selected by performing a comparison between the candidate patch and the pixels already in the output image.
- An optimal (irregularly shaped) portion of this rectangle is computed and only these pixels are copied to the output image.

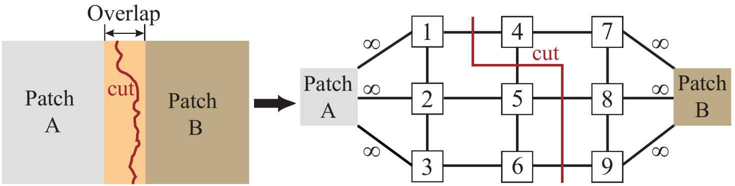
Candidate patch selection

- A candidate rectangular patch (or patch offset) is selected by performing a comparison between the candidate patch and the pixels already in the output image.
- An optimal (irregularly shaped) portion of this rectangle is computed and only these pixels are copied to the output image. This is where we use graphcuts

Matching quality

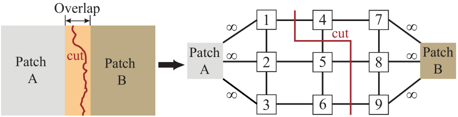
s and t two adjacent pixel positions in two copied patches overlap region. $A(s)$ and $B(s)$ pixel colors at s in the two patches. *Matching quality cost* M between s and t:

$$M(s,t,A,B) = \|A(s) - B(s)\| + \|A(t) - B(t)\|$$



[Kwatra et al. 2004]

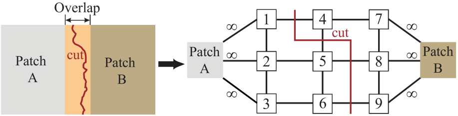
Graph Cut between two patches



[Kwatra et al. 2004]

- Goal Find a minimal path separating A from B

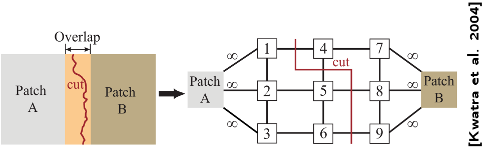
Graph Cut between two patches



[Kwatra et al. 2004]

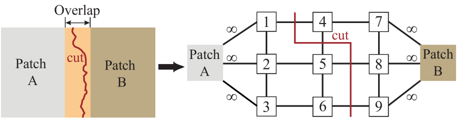
- Goal Find a minimal path separating A from B
- Connect neighboring pixels by an edge with weight $M(s, t, A, B)$

Graph Cut between two patches



- Goal Find a minimal path separating A from B
- Connect neighboring pixels by an edge with weight $M(s,t,A,B)$
- Add two terminal nodes corresponding to A and B

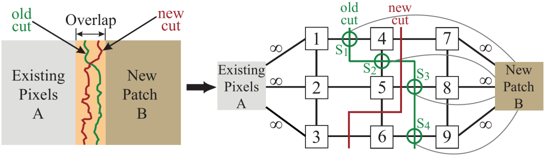
Graph Cut between two patches



[Kwatra et al. 2004]

- Goal Find a minimal path separating A from B
- Connect neighboring pixels by an edge with weight $M(s, t, A, B)$
- Add two terminal nodes corresponding to A and B
- min-cut algorithm yields the best boundary

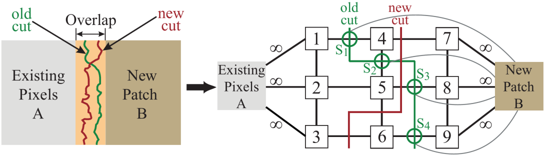
Between more than 2 patches



[Kwatra et al. 2004]

- Assume we have already copy-pasted several patches yielding existing pixel values

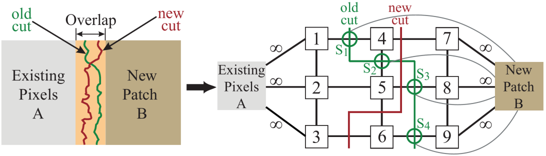
Between more than 2 patches



[Kwatra et al. 2004]

- Assume we have already copy-pasted several patches yielding existing pixel values
- Copying a new patch B

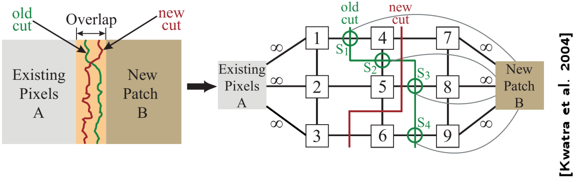
Between more than 2 patches



[Kwatra et al. 2004]

- Assume we have already copy-pasted several patches yielding existing pixel values
- Copying a new patch B
- Graph cuts used to find the new seam

Between more than 2 patches

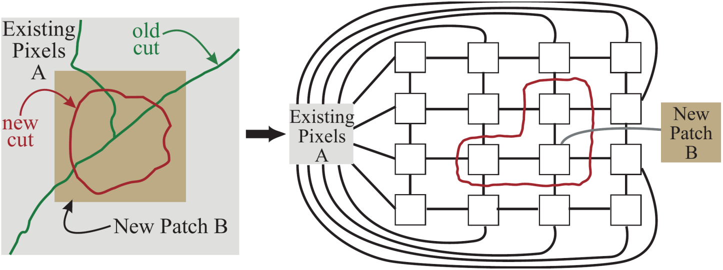


- Assume we have already copy-pasted several patches yielding existing pixel values
- Copying a new patch B
- Graph cuts used to find the new seam

Multiple Seams

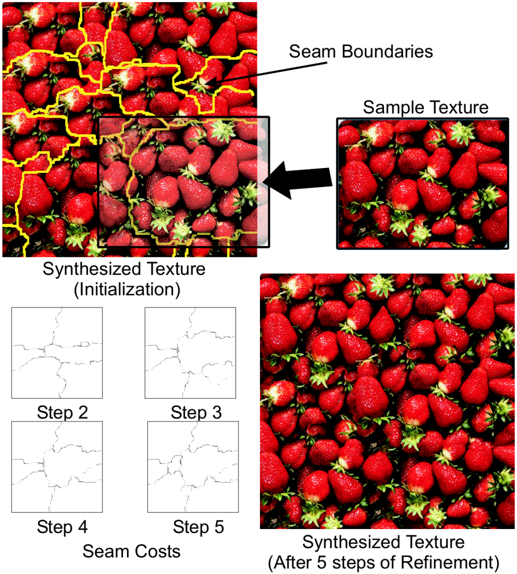
Each pixel s keeps track of the patch A_s it originated from, then the weights on graph edges between two neighboring pixels s and p originating from patches A_s, A_p is simply: $M(s, p, A_s, A_p)$

Surrounded regions



[Kwatra et al. 2004]

Algorithm


Seam Boundaries
Sample Texture
Synthesized Texture
(Initialization)
Step 2
Step 3
Step 4
Step 5
Seam Costs
Synthesized Texture
(After 5 steps of Refinement)

Patch placement

Three strategies are possible:

- Random patch placement

Patch placement

Three strategies are possible:

- Random patch placement
- Entire patch matching

Patch placement

Three strategies are possible:

- Random patch placement
- Entire patch matching
- Subpatch matching

Random patch placement

- The new patch (entire sample texture) is translated to a random offset location. The graph cut algorithm selects a piece of this patch to lay down into the out- put image and the process is repeated

Random patch placement

- The new patch (entire sample texture) is translated to a random offset location. The graph cut algorithm selects a piece of this patch to lay down into the out- put image and the process is repeated

Pros & Cons

Fastest synthesis method, good result for random textures.

Entire patch matching

- Search for translations of the input image that match well with the currently synthesized output.

Entire patch matching

- Search for translations of the input image that match well with the currently synthesized output.

Pros & Cons

Best results for structured and semi-structured textures.

Sub-patch matching

- First pick a small sub-patch in the output texture.
- Look for a sub-patch in the input texture that matches this output-sub-patch well.

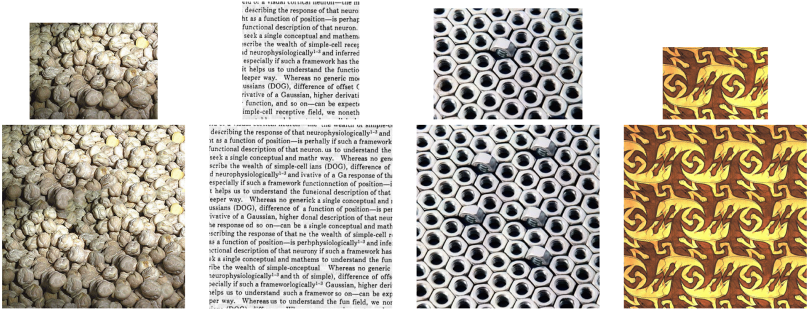
Sub-patch matching

- First pick a small sub-patch in the output texture.
- Look for a sub-patch in the input texture that matches this output-sub-patch well.

Pros & Cons

most general method.

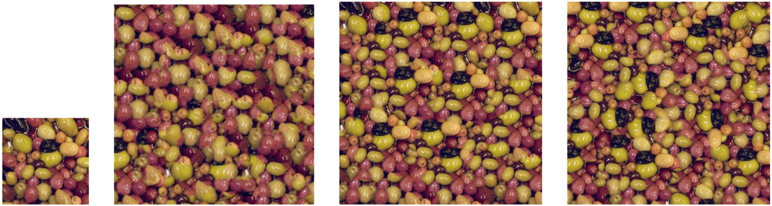
Results



[Kwatra et al. 2004]

Results

Input Image Quilting Graph cut



Input Image Quilting Graph cut Rotation & Mirroring

[Kwatra et al. 2004]

Results

[Kwatra et al. 2004]

Conclusion

- Other methods for Texture Synthesis: Gabor Noise, variational methods ... A vast literature on the subject exists
- Now: Machine learning methods (Gatys et al. 2015 and so on) beyond the scope of this course.

[Zhou et al. 2018]