CPSC 425: Computer Vision

Lecture 15: Corner Detection (cont), Texture

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little and Fred Tung )
Menu for Today (February 12, 2021)

— Texture

Readings:

— Today's Lecture: Forsyth & Ponce (2nd ed.) 5.3, 6.1, 6.3
— Next Lecture: Forsyth & Ponce (2nd ed.) 3.1-3.3

— Assignment 2: Face Detection in a Scaled Representation is due today
— Assignment 3: Texture Synthesis is out soon
<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="template1.png" alt="Template" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="template2.png" alt="Template" /></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td><img src="imageL.png" alt="Image" /></td>
<td></td>
</tr>
</tbody>
</table>
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image</th>
<th>Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td><img src="image1" alt="Image" /></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td><img src="image3" alt="Image" /></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td><img src="image4" alt="Image" /></td>
</tr>
</tbody>
</table>

$(x_0, y_0)$
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image</th>
<th>Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>(x₀, y₀)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>(xₗ, yₗ)</td>
<td></td>
</tr>
</tbody>
</table>

JUDYBATS
### Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image0.png" alt="Image" /></td>
<td>$(x_0, y_0)$</td>
</tr>
</tbody>
</table>
| 1     | ![Image](image1.png) | $x_L = x_0 s^L$  
               $y_L = y_0 s^L$ |
| ...   | ![Image](imageL.png) | $(x_L, y_L)$ |
## Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid(s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image01" alt="Image" /></td>
<td>((x_0, y_0) \pm (w_0/2, h_0/2))</td>
</tr>
</tbody>
</table>
| 1     | ![Image](image02) | \(x_L = x_0 s^L\)  \(w_L = w_0 s^L\)  
\(y_L = y_0 s^L\)  \(h_L = h_0 s^L\) |
| \(L\) | ![Image](image03) | \((x_L, y_L) \pm (w_L/2, h_L/2)\) |
### Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image1.jpg" alt="Judybats Image" /></td>
<td><img src="image2.jpg" alt="Template Image" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image3.jpg" alt="Judybats Image" /></td>
<td><img src="image4.jpg" alt="Template Image" /></td>
</tr>
<tr>
<td>...</td>
<td><img src="image5.jpg" alt="Judybats Image" /></td>
<td><img src="image6.jpg" alt="Template Image" /></td>
</tr>
<tr>
<td>L</td>
<td><img src="image7.jpg" alt="Judybats Image" /></td>
<td><img src="image8.jpg" alt="Template Image" /></td>
</tr>
</tbody>
</table>

(x_L, y_L)
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>![Image 0]</td>
<td><img src="https://example.com/template1.png" alt="Template" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>((x_0, y_0))</td>
</tr>
<tr>
<td>1</td>
<td>![Image 1]</td>
<td>[x_0 = x_L \times \frac{1}{s^L}]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[y_0 = y_L \times \frac{1}{s^L}]</td>
</tr>
<tr>
<td>...</td>
<td>![Image L]</td>
<td><img src="https://example.com/templateL.png" alt="Template" /></td>
</tr>
<tr>
<td>L</td>
<td>![Image L]</td>
<td>((x_L, y_L))</td>
</tr>
</tbody>
</table>
## Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid(s)</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image0.png" alt="Image" /></td>
<td>$(x_0, y_0) \pm (w_0/2, h_0/2)$</td>
</tr>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Image" /></td>
<td>$x_0 = x_L \times \frac{1}{s^L}$, $w_0 = w \times \frac{1}{s^L}$, $y_0 = y_L \times \frac{1}{s^L}$, $h_0 = h \times \frac{1}{s^L}$</td>
</tr>
<tr>
<td>...</td>
<td><img src="image2.png" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td><img src="imageL.png" alt="Image" /></td>
<td>$(x_L, y_L) \pm (w_L/2, h_L/2)$</td>
</tr>
</tbody>
</table>
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template Pyramid (1/s)</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>JUDYBATS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Template matching

Both allow search over scale
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template</th>
<th>Template Pyramid (1/s)</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Image of JUDYBATS</td>
<td>&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Image of JUDYBATS</td>
<td>&gt;</td>
<td>Faster</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>Image of JUDYBATS</td>
<td>&gt;</td>
<td>Slow</td>
<td></td>
</tr>
</tbody>
</table>

Both allow **search over scale**
Template matching

<table>
<thead>
<tr>
<th>Level</th>
<th>Image Pyramid (s)</th>
<th>Template Pyramid (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Image Pyramid" /></td>
<td><img src="image" alt="Template Pyramid" /></td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Image Pyramid" /></td>
<td><img src="image" alt="Template Pyramid" /></td>
</tr>
<tr>
<td>L</td>
<td><img src="image" alt="Image Pyramid" /></td>
<td><img src="image" alt="Template Pyramid" /></td>
</tr>
</tbody>
</table>

Note: This does not search over scales
Template matching

Level 0

Image Pyramid(s)

Template Pyramid(s)
Today’s “fun” Example: Face Detection
Today’s “fun” Example: Face Detection
Today’s “fun” Example: Face Detection

https://www.youtube.com/watch?v=gWjBleSfZBk
Lecture 14 re-cap

Find local maxima in both **position** and **scale**

---

**Slide Credit**: Ioannis (Yannis) Gkioulekas (CMU)
Formally ... 

Highest response when the signal has the same **characteristic scale** as the filter

---

Slide Credit: Ioannis (Yannis) Gkioulkas (CMU)
Characteristic Scale

characteristic scale - the scale that produces peak filter response
Applying **Laplacian** Filter at Different **Scales**

![Laplacian Filter Examples](image)

**Slide Credit:** Ioannis (Yannis) Gkioulkas (CMU)
Local max in space and scale

Optional subtitle

cross-scale maximum

local maximum

local maximum

local maximum
Texture

What is **texture**?

Texture is widespread, easy to recognize, but hard to define

Views of large numbers of small objects are often considered textures
— e.g. grass, foliage, pebbles, hair

Patterned surface markings are considered textures
— e.g. patterns on wood

*Figure Credit:* Alexei Efros and Thomas Leung
(Functional) **Definition:**

**Texture** is detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements.
(Functional) Definition:

**Texture** is detail in an image that is at a scale too small to be resolved into its constituent elements and at a scale large enough to be apparent in the spatial distribution of image measurements.

Sometimes, textures are thought of as patterns composed of repeated instances of one (or more) identifiable elements, called **textons**.

— e.g. bricks in a wall, spots on a cheetah
Uses of **Texture**

Texture can be a strong cue to **object identity** if the object has distinctive material properties.

Texture can be a strong cue to an **object’s shape** based on the deformation of the texture from point to point.

— Estimating surface orientation or shape from texture is known as “**shape from texture**"
Texture

We will look at two main questions:

1. How do we represent texture?
   → Texture **analysis**

2. How do we generate new examples of a texture?
   → Texture **synthesis**

We begin with texture synthesis to set up **Assignment 3**
Texture *Synthesis*

Why might we want to synthesize texture?

1. To fill holes in images (*inpainting*)
   — Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.
   — We need to find something to put in place of the pixels that were removed
   — We synthesize regions of texture that fit in and look convincing
Texture Synthesis

Why might we want to synthesize texture?

1. To fill holes in images (inpainting)
   — Art directors might want to remove telephone wires. Restorers might want to remove scratches or marks.
   — We need to find something to put in place of the pixels that were removed
   — We synthesize regions of texture that fit in and look convincing

2. To produce large quantities of texture for computer graphics
   — Good textures make object models look more realistic
Texture Synthesis

Szeliski, Fig. 10.49
Texture Synthesis

Bush campaign digitally altered TV ad
President Bush’s campaign acknowledged Thursday that it had digitally altered a photo that appeared in a national cable television commercial. In the photo, a handful of soldiers were multiplied many times.

Photo Credit: Associated Press
Texture Synthesis

Cover of “The Economist,” June 19, 2010

Photo Credit (right): Reuters/Larry Downing
Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish
Assignment 3 Preview: Texture Synthesis

Task: Make donkey vanish

Method: Fill-in regions using texture from the white box
Assignment 3 Preview: Texture Synthesis

**Task:** Make donkey vanish

**Method:** Fill-in regions using texture from the white box
**Texture Synthesis**

**Objective**: Generate new examples of a texture. We take a “data-driven” approach.

**Idea**: Use an image of the texture as the source of a probability model.
- Draw samples directly from the actual texture.
- Can account for more types of structure.
- Very simple to implement.
- Success depends on choosing a correct “distance”.
Texture Synthesis by Non-parametric Sampling

Alexei Efros and Thomas Leung
UC Berkeley

Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt
Efros and Leung

- Wood
- Granite
Efros and Leung

- white bread
- brick wall
Like **Copying**, But not Just Repetition
Efros and Leung: Synthesizing One Pixel

— What is **conditional** probability distribution of $p$, given the neighbourhood window?
— What is **conditional** probability distribution of \( p \), given the neighbourhood window?

— Directly search the input image for all such neighbourhoods to produce a **histogram** for \( p \)
— What is **conditional** probability distribution of \( p \), given the neighbourhood window?

— Directly search the input image for all such neighbourhoods to produce a **histogram** for \( p \)

— To **synthesize** \( p \), pick one match at random
— Since the sample image is finite, an exact neighbourhood match might not be present
Efros and Leung: Synthesizing One Pixel

— Since the sample image is finite, an exact neighbourhood match might not be present

— Find the **best match** using SSD (sum of squared differences) error, weighted by Gaussian to emphasize local structure, and take all samples within some distance from that match
Efros and Leung: Synthesizing Many Pixels

For multiple pixels, "grow" the texture in layers
— In the case of hole-filling, start from the edges of the hole

For an interactive demo, see


(written by Julieta Martinez, a previous CPSC 425 TA)
Randomness Parameter

Texturing a Sphere

Sample image

2D

3D

Slide Credit: [link](http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt)
Efros and Leung: More Synthesis Results

![Image of synthesis results with varying window sizes](image)

Forsyth & Ponce (2nd ed.) Figure 6.12
Efros and Leung: Image Extrapolation

Slide Credit: http://graphics.cs.cmu.edu/people/efros/research/NPS/efros-iccv99.ppt
“Big Data” Meets Inpainting

“Big Data” enables surprisingly simple non-parametric, matching-based techniques to solve complex problems in computer graphics and vision.

Suppose instead of a single image, you had a massive database of a million images. What could you do?
“Big Data” Meets Inpainting

Original Image

Input

Figure Credit: Hays and Efros 2007
“Big Data” Meets Inpainting

Input | Scene Matches | Output

Figure Credit: Hays and Efros 2007
Effectiveness of “Big Data”
Effectiveness of "Big Data"

10 nearest neighbors from a collection of 20,000 images

Figure Credit: Hays and Efros 2007
Effectiveness of “Big Data”

10 nearest neighbors from a collection of 2 million images

Figure Credit: Hays and Efros 2007
“Big Data” Meets Inpainting

Figure Credit: Hays and Efros 2007
“Big Data” Meets Inpainting

**Algorithm** sketch (Hays and Efros 2007):

1. Create a short list of a few hundred “best matching" images based on global image statistics

2. Find patches in the short list that match the context surrounding the image region we want to fill

3. Blend the match into the original image

Purely **data-driven**, requires no manual labeling of images
“Big Data” Meets Inpainting

Original Image

Input

Figure Credit: Hays and Efros 2007
“Big Data” Meets Inpainting

Figure Credit: Hays and Efros 2007
“Big Data” Meets Inpainting

Figure Credit: Hays and Efros 2007
Goal of Texture **Synthesis**

Given a finite sample of some texture, the goal is to synthesize other samples from that same texture

— The sample needs to be "large enough"

**Credit**: Bill Freeman
Goal of Texture Analysis

Compare textures and decide if they’re made of the same “stuff”

Credit: Bill Freeman
Definition of Texture (Re-Cap)

Recall that texture is easy to recognize but hard to define
— A functional definition was presented last class

We need representations that differ in ways that are easy to observe when two textures are significantly different.

Recall that textures can often be thought of as patterns composed of repeated instances of one (or more) identifiable elements, called textons
— e.g. bricks in a wall, spots on a cheetah
Texture Segmentation

Question: Is texture a property of a point or a property of a region?
Texture **Segmentation**

**Question:** Is texture a property of a point or a property of a region?

**Answer:** We need a region to have a texture.
Texture **Segmentation**

**Question**: Is texture a property of a point or a property of a region?

**Answer**: We need a region to have a texture.

There is a “chicken–and–egg” problem. Texture segmentation can be done by detecting boundaries between regions of the same (or similar) texture. Texture boundaries can be detected using standard edge detection techniques applied to the texture measures determined at each point.
Texture Segmentation

Question: Is texture a property of a point or a property of a region?

Answer: We need a region to have a texture.

There is a “chicken–and–egg” problem. Texture segmentation can be done by detecting boundaries between regions of the same (or similar) texture. Texture boundaries can be detected using standard edge detection techniques applied to the texture measures determined at each point.

We compromise! Typically one uses a local window to estimate texture properties and assigns those texture properties as point properties of the window’s center row and column.
Question: How many degrees of freedom are there to texture?

(Mathematical) Answer: Infinitely many

(Perceptual Psychology) Answer: There are perceptual constraints. But, there is no clear notion of a “texture channel” like, for example, there is for an RGB colour channel.
Texture **Representation**

**Observation**: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties.

**Idea**: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region.
Texture Representation

**Observation:** Textures are made up of generic sub-elements, repeated over a region with similar statistical properties.

**Idea:** Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region.

**Question:** What filters should we use?

**Answer:** Human vision suggests spots and oriented edge filters at a variety of different orientations and scales.
Texture Representation

**Observation**: Textures are made up of generic sub-elements, repeated over a region with similar statistical properties.

**Idea**: Find the sub-elements with filters, then represent each point in the image with a summary of the pattern of sub-elements in the local region.

**Question**: What filters should we use?

**Answer**: Human vision suggests spots and oriented edge filters at a variety of different orientations and scales.

**Question**: How do we “summarize”?

**Answer**: Compute the mean or maximum of each filter response over the region — Other statistics can also be useful.
Human Texture Perception

Fig. 1 Top row. Textures consisting of Xs within a texture composed of Ls. The micropatterns are placed at random orientations on a randomly perturbed lattice. a. The bars of the Xs have the same length as the bars of the Ls. b. The bars of the Ls have been lengthened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is enhanced. c. The bars of the Ls have been shortened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is impaired. Bottom row: the responses of a size-tuned mechanism. d. response to image a; e. response to image b; f. response to image c.

Credit: Bergen and Adelson, Nature, 1988
Texture Representation

Figure Credit: Leung and Malik, 2001
Texture Representation

original image

derivative filter responses, squared

<table>
<thead>
<tr>
<th>Win. #1</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows

Slide Credit: Trevor Darrell
Texture Representation

original image

derivative filter responses, squared

<table>
<thead>
<tr>
<th>Win. #</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>#2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>#9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows

Slide Credit: Trevor Darrell
Texture Representation

Far: dissimilar textures
Close: similar textures

<table>
<thead>
<tr>
<th>Window</th>
<th>(\text{mean } \frac{d}{dx}\text{ value})</th>
<th>(\text{mean } \frac{d}{dy}\text{ value})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Statistics to summarize patterns in small windows

Slide Credit: Trevor Darrell
Texture Representation

Chi-square

\{ 0.1 \}

\{ 0.8 \}
Spots and Bars (Fine Scale)

Forsyth & Ponce (1st ed.) Figures 9.3–9.4
Spots and Bars (Coarse Scale)

Forsyth & Ponce (1st ed.) Figures 9.3 and 9.5
Comparison of Results

Forsyth & Ponce (1st ed.) Figures 9.4–9.5
A Short **Exercise**: Match the texture to the response

![Filters and Mean abs responses]

Slide Credit: James Hays
A Short **Exercise**: Match the texture to the response

Slide Credit: James Hays
Summary

**Texture** representation is hard
— difficult to define, to analyze
— texture synthesis appears more tractable

Objective of texture **synthesis** is to generate new examples of a texture
— Efros and Leung: Draw samples directly from the texture to generate one pixel at a time. A “data-driven" approach.

Approaches to texture embed assumptions related to human perception