Lecture 9: Template Matching (cont.) and Scaled Representations

( unless otherwise stated slides are taken or adopted from Bob Woodham, Jim Little, Leon Sigal, and Fred Tung )

Image Credit: https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html
Menu for Today (September 27, 2021)

Topics:

— Template Matching
— Normalized Correlation
— Scaled Representations
— Image Derivatives

Readings:

— Today’s Lecture: Forsyth & Ponce (2nd ed.) 4.5 - 4.7
— Next Lecture: Forsyth & Ponce (2nd ed.) 5.1 - 5.2

Reminders:

— Assignment 1: Image Filtering and Hybrid Images due TODAY
— Assignment 2: Face Detection in a Scaled Representation out soon
Today’s “fun” Example: Müller-Lyer Illusion
Today’s “fun” Example: Müller-Lyer Illusion
Today’s “fun” Example: Müller-Lyer Illusion
Salt crystal formation
Lecture 8: Re-cap

In the continuous case, images are functions of two spatial variables, x and y.

The discrete case is obtained from the continuous case via sampling (i.e. tessellation, quantization).

If a signal is **bandlimited** then it is possible to design a sampling strategy such that the sampled signal captures the underlying continuous signal exactly.

Adequate sampling may not always be practical. In such cases there is a trade-off between “things missing” and “artifacts”.

— Different applications make the trade-off differently
Lecture 8: Re-cap

“Color” is **not** an objective physical property of light (electromagnetic radiation).
Instead, light is characterized by its wavelength.

Color Filter Arrays (CFAs) allow capturing of mosaiced color information; the layout of the mosaic is called **Bayer** pattern.

**Demosaicing** is the process of taking the RAW image and interpolating missing color pixels per channel.
Lecture 8: Re-cap

How can we find a part of one image that matches another?

or,

How can we find instances of a pattern in an image?
Lecture 8: Re-cap

How can we find a part of one image that matches another?

or,

How can we find instances of a pattern in an image?

**Key Idea:** Use the pattern as a **template**
Template Matching

A toy example

Template (mask)

Scene

Slide Credit: Kristen Grauman
Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.
— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.
Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

<table>
<thead>
<tr>
<th>Image Patch 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Image Patch 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Template</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

16
Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.
— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

\[
\begin{array}{c}
\text{Image Patch 1} \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array} \\
\text{Image Patch 2} \\
\begin{array}{ccc}
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{Template} \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{Result} \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\end{array}
\]

In this example, the dot product of the template and the first image patch results in a value of 0, indicating no match. The dot product with the second image patch results in a value of 2, indicating a match.
**Template Matching**

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

<table>
<thead>
<tr>
<th>Image Patch 1</th>
<th>Template</th>
<th>Image Patch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>0 1 1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad \quad \quad
\begin{array}{ccc}
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad \quad \quad
\begin{array}{ccc}
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\quad \quad \quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad \quad \quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad = \quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad = \ 3
\]

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
\end{array}
\quad \quad \quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\quad = \ 1
\]
## Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

<table>
<thead>
<tr>
<th>Image Patch 1</th>
<th>Template</th>
<th>Image Patch 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 0</td>
<td>1 0 1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 1 0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 1</td>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 1</td>
<td>0 0 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

×255

\[
\begin{align*}
\text{Result} & = 3 \\
\text{Result} & = 1 \times 255
\end{align*}
\]
**Template Matching**

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.

— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

Image Patch 1

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Template

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Element multiply

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
\end{bmatrix} \times 255 \quad = \quad 3
\]

The dot product may be large simply because the image region is bright.

We need to normalize the result in some way.

Image Patch 2

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \times 255 \quad = \quad 1 \times 255
\]
Template Matching

Let \( \mathbf{a} \) and \( \mathbf{b} \) be vectors. Let \( \theta \) be the angle between them. We know

\[
\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \, ||\mathbf{b}||} = \frac{\mathbf{a} \cdot \mathbf{b}}{\sqrt{(\mathbf{a} \cdot \mathbf{a})(\mathbf{b} \cdot \mathbf{b})}} = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \, ||\mathbf{b}||}
\]

where \( \cdot \) is dot product and \( || \) is vector magnitude

Correlation is a dot product.

Correlation measures similarity between the filter and each local image region

**Normalized correlation** varies between \(-1\) and 1

Normalized correlation attains the value 1 when the filter and image region are identical (up to a scale factor).  

21
Template Matching

We can think of convolution/correlation as comparing a template (the filter) with each local image patch.

— Consider the filter and image patch as vectors.
— Applying a filter at an image location can be interpreted as computing the dot product between the filter and the local image patch.

\[
\begin{array}{c|c|c|c}
\text{Image} & \text{Patch} 1 & \text{Template} & \text{Patch} 2 \\
\hline
1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\quad
\begin{array}{c|c|c|c}
& \times 1/3 & \times 1/3 & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\quad
\begin{array}{c|c|c|c}
& \times 1/3 & \times 1/3 & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\frac{1}{3} \times \frac{1}{3^3} = \frac{1}{3^3}
\]

\[
\frac{1}{3} \times \frac{1}{3} = \frac{1}{3}
\]
Template Matching

Assuming template is all positive, what does this tell us about correlation map?
Template Matching

Assuming template is all positive, what does this tell us about correlation map?

Detected template

Correlation map

\[
\frac{a}{|a|} \cdot \frac{b}{|b|} = ?
\]

Slide Credit: Kristen Grauman
Template Matching

Assuming template is all positive, what does this tell us about correlation map?
Template Matching

Assuming template is all positive, what does this tell us about correlation map?

\[
\frac{a}{|a|} \frac{b}{|b|} = ?
\]
**Template Matching**

Assuming template is all positive, what does this tell us about correlation map?

Detected template

Correlation map

\[ \frac{a}{|a|} \cdot \frac{b}{|b|} = ? \]

*Slide Credit: Kristen Grauman*
Template Matching

Detection can be done by comparing correlation map score to a threshold

What happens if the threshold is relatively low?

Slide Credit: Kristen Grauman
Template Matching

Detection can be done by comparing correlation map score to a threshold

What happens if the threshold is relatively low?

Slide Credit: Kristen Grauman
Template Matching

Detection can be done by comparing correlation map score to a threshold

What happens if the threshold is very high (e.g., 0.99)?

Slide Credit: Kristen Grauman
Template Matching

Detection can be done by comparing correlation map score to a threshold

What happens if the threshold is very high (e.g., 0.99)?

Slide Credit: Kristen Grauman
Linear filtering the entire image computes the entire set of dot products, one for each possible alignment of filter and image

Important Insight:
— filters look like the pattern they are intended to find
— filters find patterns they look like

Linear filtering is sometimes referred to as template matching
Example 1:

Example 1:

Example 1:

Example 1:

Example 1:

Example 1:

Template (left), image (middle), normalized correlation (right)

Note peak value at the true position of the hand

Template Matching

When might template matching fail?
Template Matching

When might template matching fail?

— Different scales
When might **template matching** fail?

— Different scales

— Different orientation
Template Matching

When might template matching fail?

- Different scales
- Different orientation
- Lighting conditions
Template Matching

When might template matching fail?

— Different scales
— Different orientation
— Lighting conditions
— Left vs. Right hand
When might **template matching fail**?

- Different scales
- Different orientation
- Lighting conditions
- Left vs. Right hand
- Partial Occlusions
When might template matching fail?

- Different scales
- Different orientation
- Lighting conditions
- Left vs. Right hand
- Partial Occlusions
- Different Perspective
- Motion / blur
Template Matching Summary

**Good News:**
- works well in presence of noise
- relatively easy to compute

**Bad News:**
- sensitive to (spatial) scale change
- sensitive to 2D rotation

**More Bad News:**
When imaging 3D worlds:
- sensitive to viewing direction and pose
- sensitive to conditions of illumination
Scaled Representations

**Problem:** Make template matching robust to changes in 2D (spatial) scale.

**Key Idea(s):** Build a scaled representation: the Gaussian image pyramid

**Alternatives:**
- use multiple sizes for each given template
- ignore the issue of 2D (spatial) scale

**Theory:** Sampling theory allows us to build image pyramids in a principled way

**“Gotchas:”**
- template matching remains sensitive to 2D orientation, 3D pose and illumination
Scaled Representations: Goals

to find **template matches** at all scales

— template size constant, image scale varies

— finding hands or faces when we don’t know what size they are in the image
Scaled Representations: Goals

to find template matches at all scales
— template size constant, image scale varies
— finding hands or faces when we don’t know what size they are in the image

efficient search for image–to–image correspondences
— look first at coarse scales, refine at finer scales
— much less cost (but may miss best match)
Scaled Representations: Goals

to find **template matches** at all scales
— template size constant, image scale varies
— finding hands or faces when we don’t know what size they are in the image

efficient search for image–to–image correspondences
— look first at coarse scales, refine at finer scales
— much less cost (but may miss best match)

to examine all **levels of detail**
— find edges with different amounts of blur
— find textures with different spatial frequencies (i.e., different levels of detail)
Shrinking the Image

We can’t shrink an image simply by taking every second pixel

Why?
Shrinking the Image

We can’t shrink an image simply by taking every second pixel. If we do, characteristic **artifacts** appear:
— small phenomena can look bigger
— fast phenomena can look slower

Common **examples** include:
— checkerboard patterns misrepresented in video games
— striped shirts look funny on colour television
— wagon wheels roll the wrong way in movies
Shrinking the Image

Forsyth & Ponce (2nd ed.) Figure 4.12-4.14 (top rows)
Template Matching: Sub-sample with Gaussian Pre-filtering

Apply a smoothing filter first, then throw away half the rows and columns.

Gaussian filter delete even rows delete even columns

Gaussian filter delete even rows delete even columns

1/2

1/4

1/8

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Template Matching: Sub-sample with Gaussian Pre-filtering

1/2

1/4 (2x zoom)

1/8 (4x zoom)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Template Matching: Sub-sample with NO Pre-filtering

1/2  1/4 (2x zoom)  1/8 (4x zoom)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Gaussian Pre-filtering

**Question**: How much smoothing is needed to avoid aliasing?
Gaussian Pre-filtering

**Question:** How much smoothing is needed to avoid aliasing?

**Answer:** Smoothing should be sufficient to ensure that the resulting image is band limited “enough” to ensure we can sample every other pixel.

**Practically:** For every image reduction of 0.5, smooth by $\sigma = 1$
An image pyramid is a collection of representations of an image. Typically, each layer of the pyramid is half the width and half the height of the previous layer.

In a Gaussian pyramid, each layer is smoothed by a Gaussian filter and resampled to get the next layer.
Gaussian Pyramid

Again, let $\otimes$ denote convolution

Create each level from previous one
— smooth and (re)sample

Smooth with Gaussian, taking advantage of the fact that

$$G_{\sigma_1}(x) \otimes G_{\sigma_2}(x) = G_{\sqrt{\sigma_1^2 + \sigma_2^2}}(x)$$
Example 2: Gaussian Pyramid

Forsyth & Ponce (2nd ed.) Figure 4.17

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 2: Gaussian Pyramid

What happens to the details?

Forsyth & Ponce (2nd ed.) Figure 4.17

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 2: Gaussian Pyramid

What happens to the details?
— They get smoothed out as we move to higher levels

What is preserved at the higher levels?

Forsyth & Ponce (2nd ed.) Figure 4.17

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 2: Gaussian Pyramid

What happens to the details?
— They get smoothed out as we move to higher levels

What is preserved at the higher levels?
— Mostly large uniform regions in the original image

How would you reconstruct the original image from the image at the upper level?

Forsyth & Ponce (2nd ed.) Figure 4.17

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
Example 2: Gaussian Pyramid

What happens to the details?
— They get smoothed out as we move to higher levels

What is preserved at the higher levels?
— Mostly large uniform regions in the original image

How would you reconstruct the original image from the image at the upper level?
— That’s not possible

Forsyth & Ponce (2nd ed.) Figure 4.17

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)
From Template Matching to **Local Feature Detection**

We’ll now shift from global template matching to **local feature detection**.

Consider the problem of finding images of an elephant using a template.
From Template Matching to **Local Feature Detection**

We’ll now shift from global template matching to **local feature detection**

Consider the problem of finding images of an elephant using a template.

An elephant looks different from different viewpoints:
- from above (as in an aerial photograph or satellite image)
- head on
- sideways (i.e., in profile)
- rear on

What happens if parts of an elephant are obscured from view by trees, rocks, other elephants?
From Template Matching to **Local Feature Detection**

Find the chair in this image

This is a chair

Output of normalized correlation

*Slide Credit:* Li Fei-Fei, Rob Fergus, and Antonio Torralba
From Template Matching to **Local Feature Detection**

Find the chair in this image

Pretty much garbage
Simple template matching is not going to make it

**Slide Credit:** Li Fei-Fei, Rob Fergus, and Antonio Torralba
A difficult chair image
From Template Matching to **Local Feature Detection**

— Move from global template matching to **local template matching**

— Local template matching also called local **feature detection**

— Obvious local features to detect are **edges** and **corners**
Estimating **Derivatives**

Recall, for a 2D (continuous) function, $f(x,y)$

\[
\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}
\]

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution
Recall, for a 2D (continuous) function, $f(x,y)$

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution

A (discrete) approximation is

$$\frac{\partial f}{\partial x} \approx \frac{F(X + 1, y) - F(x, y)}{\Delta x}$$
Estimating Derivatives

Recall, for a 2D (continuous) function, $f(x,y)$

\[
\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}
\]

Differentiation is linear and shift invariant, and therefore can be implemented as a convolution.

A (discrete) approximation is

\[
\frac{\partial f}{\partial x} \approx \frac{F(X + 1, y) - F(x, y)}{\Delta x}
\]

[ ]
Estimating Derivatives

A similar definition (and approximation) holds for \( \frac{\partial f}{\partial y} \).

Image noise tends to result in pixels not looking exactly like their neighbours, so simple “finite differences” are sensitive to noise.

The usual way to deal with this problem is to smooth the image prior to derivative estimation.
Example 1D
Example 1D

Signal  

| 0.5 | 0.5 | 0.5 | 0.4 | 0.3 | 0.2 | 0.2 | 0.2 | 0.35 | 0.5 | 0.5 |
Example 1D

Signal

| 0.5 | 0.5 | 0.5 | 0.4 | 0.3 | 0.2 | 0.2 | 0.2 | 0.35 | 0.5 | 0.5 |

Derivative
Example 1D

Signal: 0.5 0.5 0.5 0.4 0.3 0.2 0.2 0.2 0.35 0.5 0.5

Derivative: 0.0
Example 1D

Signal: 0.5 0.5 0.5 0.4 0.3 0.2 0.2 0.2 0.35 0.5 0.5

Derivative: 0.0
Example 1D

Signal 0.5  0.5  0.5  0.4  0.3  0.2  0.2  0.2  0.35  0.5  0.5

Derivative  0.0  0.0
Example 1D

Signal

| 0.5 | 0.5 | 0.5 | 0.4 | 0.3 | 0.2 | 0.2 | 0.2 | 0.35 | 0.5 | 0.5 |

Derivative

| 0.0 | 0.0 |
Example 1D

<table>
<thead>
<tr>
<th>Signal</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.4</th>
<th>0.3</th>
<th>0.2</th>
<th>0.2</th>
<th>0.2</th>
<th>0.35</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derivative</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 1D

<table>
<thead>
<tr>
<th>Signal</th>
<th>0.5</th>
<th>0.5</th>
<th>0.5</th>
<th>0.4</th>
<th>0.3</th>
<th>0.2</th>
<th>0.2</th>
<th>0.2</th>
<th>0.35</th>
<th>0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Derivative</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.15</td>
<td>0.15</td>
<td>0.0</td>
<td>X</td>
</tr>
</tbody>
</table>
Estimating **Derivatives**

**Derivative** in Y (i.e., vertical) direction

Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top middle)
Estimating Derivatives

**Derivative** in Y (i.e., vertical) direction

*Note:* visualized by adding 0.5/128

Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top middle)
Estimating **Derivatives**

**Derivative** in X (i.e., horizontal) direction

Forsyth & Ponce (1st ed.) Figure 7.4 (top left & top right)
Summary

**Template matching** as (normalized) correlation. Template matching is not robust to changes in:
- 2D spatial scale and 2D orientation
- 3D pose and viewing direction
- illumination

**Scaled representations** facilitate
- template matching at multiple scales
- efficient search for image–to–image correspondences
- image analysis at multiple levels of detail

A **Gaussian pyramid** reduces artifacts introduced when sub-sampling to coarser scales