Chapter 3.2
Image Derivatives

Image Processing and Computer Vision

LE Thanh Sach
Faculty of Computer Science and Engineering
Ho Chi Minh University of Technology, VNU-HCM

Overview

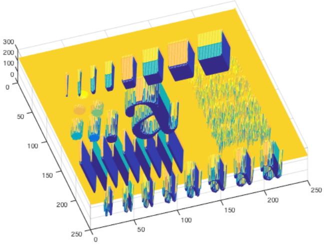
1. **Image model**
2. **First-order derivative**
3. **Gradient**
4. **Second-order derivative**
5. **Laplacian**
6. **Applications of Laplacican**

Image Model

Image model

- Image is a function of two variables x and y: $f(x, y)$
- It can seen as a surface on 2D-space.

An gray image



Mesh model of the image

Derivative of one variable function

Taylor expansion for $f(x + \Delta x)$:

$$f(x + \Delta x) = f(x) + \Delta x \times f^{'}(x) + \frac{\Delta x^2}{2!} \times f^{''}(x) + O(\Delta x^3) \tag{1}$$

First-order derivative from Eq. (1)

Forward Approximation:

$$\begin{aligned} f^{'}(x) &\cong \frac{f(x + \Delta x) - f(x)}{\Delta x} \\ &\cong f(x+1) - f(x) \quad \textbf{in discrete domain} \end{aligned}$$

- This approximation has error $O(\Delta x)$

Derivative of one variable function

Taylor expansion for $f(x - \Delta x)$:

$$f(x - \Delta x) = f(x) - \Delta x \times f^{'}(x) + \frac{\Delta x^2}{2!} \times f^{''}(x) - O(\Delta x^3) \tag{2}$$

First-order derivative from Eq. (2)

Backward Approximation:

$$\begin{aligned} f^{'}(x) &\cong \frac{f(x) - f(x - \Delta x)}{\Delta x} \\ &\cong f(x) - f(x-1) \quad \textbf{in discrete domain} \end{aligned}$$

- This approximation has error $O(\Delta x)$

Derivative of one variable function

Eq. (1) - Eq. (2)

$$f(x+\Delta x) = f(x) + \Delta x \times f^{'}(x) + \frac{\Delta x^2}{2!} \times f^{''}(x) + O(\Delta x^3)$$
$$f(x-\Delta x) = f(x) - \Delta x \times f^{'}(x) + \frac{\Delta x^2}{2!} \times f^{''}(x) - O(\Delta x^3)$$

First-order derivative from Eq. (1) - Eq. (2)

Central Approximation:

$$\begin{aligned} f^{'}(x) &\cong \frac{f(x+\Delta) - f(x-\Delta x)}{2\Delta x} \\ &\cong \frac{f(x+1) - f(x-1)}{2} && \text{in discrete domain} \\ &\cong f(x+1) - f(x-1) && \text{in discrete domain} \end{aligned}$$

- This approximation has error $O(\Delta x^2)$

Derivative of one variable function

Notation

$$f_x \equiv \frac{\partial f(x,y)}{\partial x}$$

$$f_y \equiv \frac{\partial f(x,y)}{\partial y}$$

First-Order derivative of image

Derivatives	Kernel of filters
$f_x \cong f(x+1,y) - f(x,y)$	$H_x = \begin{bmatrix} 1 & -1 \end{bmatrix}$
$f_x \cong f(x,y) - f(x-1,y)$	$H_x = \begin{bmatrix} 1 & -1 \end{bmatrix}$
$f_x \cong f(x+1,y) - f(x-1,y)$	$H_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$

First-Order derivative of image

In general, fist-order derivatives can be computed by linear filters with the following kernels

Derivative on x	Derivative on y
$H_x = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$	$H_y = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$
$H_x = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$	$H_y = \begin{bmatrix} -1 & 1 \end{bmatrix}^T$
$H_x = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$	$H_y = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$
$H_x = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T$	$H_y = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T$

First-Order derivative of image

Other kernels for computing first-order derivatives

Name	Derivative on x	Derivative on y
Prewitt	$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$H_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Robert	$H_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$H_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

First-Order derivative of image

Properties

1. First-order derivatives are operators that can be used to obtain the **variation of intensities** on x and y axis.
2. First-order derivatives can be either **positive** or **negative**.
3. A large variation of intensities $\equiv$ **strong edge** $\equiv$ large value in the absolute of derivatives.
4. In order to display derivatives, we need to scale derivative images.

First-Order derivative of image

Example

```
clear; close all; clc;
im = checkerboard(50, 5, 5);
im = uint8(255*(im > 0.5));

H_1st_x = [1 0 -1];
H_1st_y = [1 0 -1]';

im_1st_x = abs(imfilter(double(im), H_1st_x));
im_1st_y = abs(imfilter(double(im), H_1st_y));

figure, imshow(im), title('Input␣Image');
figure, imshow(gscale(im_1st_x)),
 title('1st␣derivative␣on␣x');
figure, imshow(gscale(im_1st_y)),
 title('1st␣derivative␣on␣y');
```

First-Order derivative of image

Exercise

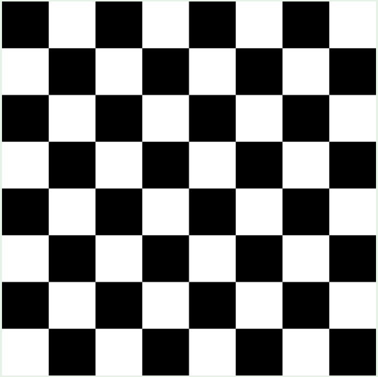


Figure: a chessboard shown by imshow(gscale(im))

- What is the image obtained from the filtering with derivatives kernels?

First-Order derivative of image

- Sobel and Prewitt kernel can be obtained by function **fspecial** in Matab
- In Matlab, the input image should be casted to **float** or **double** before filtering with function **imfilter**

Example

```
clear; close all; clc;
im = imread('cameraman.tif');

H_1st_x = fspecial('Sobel');
H_1st_y = fspecial('Sobel')'; %' = transpose

im_1st_x = abs(imfilter(double(im), H_1st_x));
im_1st_y = abs(imfilter(double(im), H_1st_y));

figure, imshow(im), title('Input Image');
figure, imshow(gscale(im_1st_x)),
title('1st derivative on x');
figure, imshow(gscale(im_1st_y)),
title('1st derivative on y');
```

Gradient

Definition

Gradient at a pixel in a image $f(x, y)$ is a vector ∇f. It is defined as

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient

Magnitude and angle of gradient vectors

Magnitude of gradient is computed by:

$$|\nabla f| = \sqrt{f_x^2 + f_y^2}$$

or, approximated by

$$|\nabla f| = |f_x| + |f_y|$$

- Magnitude of a gradient at pixel (u, v) tells us the rate of change of intensities at (u, v)
- In other words, it tells us the edge passing (u, v) is strong or not.

Gradient

Magnitude and angle of gradient vectors

Angle of gradient is computed by:
$\theta(\nabla f) = tan^{-1}(\frac{f_y}{f_x})$

- Angle of a gradient at pixel (u, v) tells us the orientation of edge passing (u, v)
- Gradient vector is perpendicular to the local edge passing (u, v)

First-Order derivative of image

Exercise

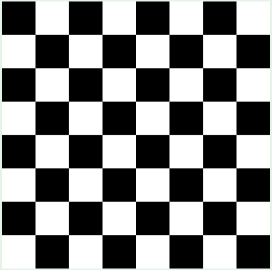


Figure: a chessboard

- How do we create an image emphasizing both of horizontal and vertical edges the above chessboard?
- Which is the direction of gradient vectors at strong edge points?

Second-order Derivative of one variable function

Second-order differential can be approximated by

$$f^{''}(x) \cong f^{'}(x) - f^{'}(x+1)$$

First-order derivatives can be approximated as

$$f^{'}(x) \cong f(x) - f(x-1)$$
$$f^{'}(x+1) \cong f(x+1) - f(x)$$

Second-order derivative

$$f^{''}(x) \cong -f^{'}(x-1) + 2f^{'}(x) - f(x+1)$$

Second-Order derivative of image

1. Second-order derivative on x
   - Math:
$$\frac{\partial^2 f(x,y)}{\partial x^2} = -f(x-1,y) + 2f(x,y) - f(x+1,y)$$
   - Kernel: $H_{conv} = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$
2. Second-order derivative on y
   - Math:
$$\frac{\partial^2 f(x,y)}{\partial y^2} = -f(x,y-1) + 2f(x,y) - f(x,y+1)$$
   - Kernel: $H_{conv} = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$

Second-Order derivative of image

In general, second-order derivatives can be approximated by using the following kernels.

	Derivative on x	Derivative on y
Positive at center:	$H_x = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$	$H_y = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$
Negative at center:	$H_x = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$	$H_y = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$

First-order and Second-order derivative

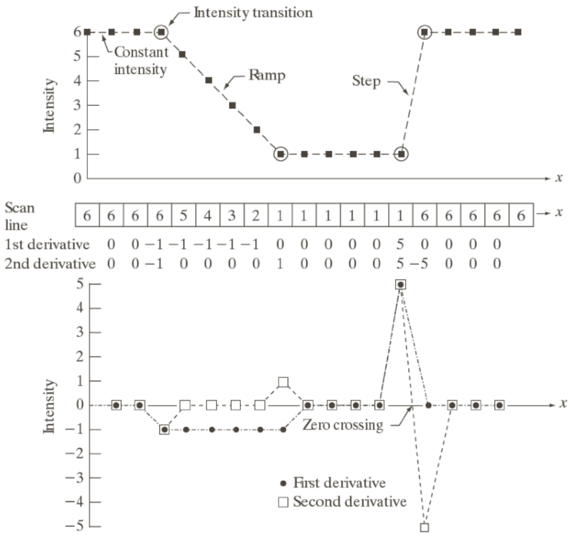


Figure: Example of First-order and Second-order derivatives

First-order and Second-order derivative

First-order derivative's properties

1. First-order derivatives produce **thicker edge** in an image.
2. First-order derivatives produce **strong response** to a gray-level step.

Second-order derivative's properties

1. Second-order derivatives produce **stronger response** to fine detail, such as thin lines or isolated points (noise).
2. Second-order derivatives produce **double response** (a **positive** and a **negative**) at step change in in gray level.
3. **Zero-crossing** at a point in second-order derivatives indicates that there is an edge passing that point.

Second-Order derivative of image

Second-order derivative on x and y

Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

- Math:

$$\begin{aligned}\nabla^2 f = \quad & -f(x-1,y) + 2f(x,y) - f(x+1,y) \\ & -f(x,y-1) + 2f(x,y) - f(x,y+1)\end{aligned}$$

- Kernel: $H_{lap} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Second-Order derivative of image

Second-order derivative on x, y, and diagonals:
Extended Laplacian operator:

- Math:

$$\begin{aligned}\nabla^2 f = \quad & -f(x-1,y) + 2f(x,y) - f(x+1,y) \\ & -f(x,y-1) + 2f(x,y) - f(x,y+1)\end{aligned}$$

- Kernel: $H_{lap} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Second-Order derivative of image

In general, Laplician can be computed by using the following kernels. The center coefficients maybe positive or negative.

$H_{lap} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$H_{lap} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
$H_{lap} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$H_{lap} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Applications of Laplacican operator

Properties of Laplacican operators

1. **Highlight** gray-level discontinues in images, i.e., edges on images
2. **De-emphasize** regions with slowly varying gray-levels

Therefore, adding (or subtracting) the original image with Laplacian image (the image obtained by filtering the original with Laplacian kernel) $\Rightarrow$ **Sharpened images**

Applications of Laplacican operator

Method for sharpening image $f(x,y)$:

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x,y) + \nabla^2 f(x,y) & \text{if the center coefficient of the Laplacian mask is positive} \end{cases}$$

Second-Order derivative of image

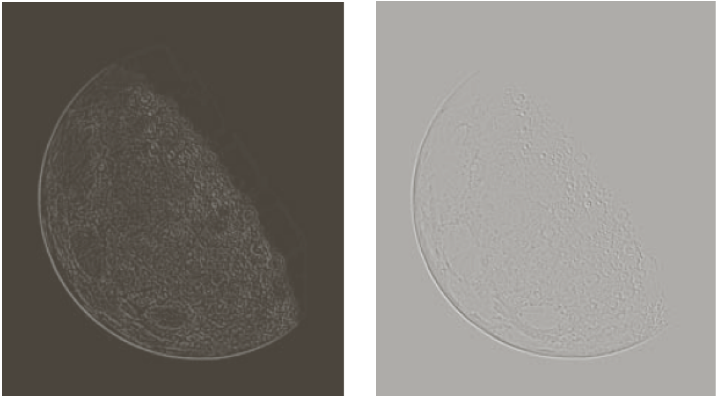
Method for sharpening image $f(x, y)$ by using linear filters. Kernels are as follows:

Positive central coefficient	Positive central coefficient
$H_{Lap} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$H_{Lap} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -9 & 1 \\ 1 & -1 & 1 \end{bmatrix}$
$H_{Lap} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$H_{Lap} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -5 & 1 \\ 0 & -1 & 0 \end{bmatrix}$

Applications of Laplacican operator: Examples

Figure: Input image: North pole of moon

Applications of Laplacican operator: Examples



(a) (b)

Figure: Laplacian images = images obtained by filtering with Laplacian operator: (a) without scaling, (b) with scaling for display purpose

Applications of Laplacican operator: Exercises

Exercise

1. How can you sharpen images, e.g., north pole of moon, with Laplacian by using Matlab and OpenCV?
2. Create blurred version and then sharpen the blurred version
   - Read checkboard image from file
   - Blur it with Box or Gaussian filter
   - Sharpen the resulting blurred version with Laplacian
   - Show the original, blurred version and sharpened version.

Unsharp masking and High-boost filtering

Unsharp masking: image obtained from subtracting a blurred version of an image from the image itself.

$$f_s(x, y) = f(x, y) - \bar{f}(x, y)$$

- $\bar{f}(x, y)$: blurred version of $f(x, y)$, e.g., using box filter
- $f_s(x, y)$: sharpened image

Unsharp masking and High-boost filtering

Exercise

1. How can you implement unsharp masking with Matlab and OpenCV?
2. Create blurred version and then sharpen the blurred version
    - Read checkboard image from file
    - Blur it with Box or Gaussian filter
    - Sharpen the resulting blurred version with Unsharp masking
    - Show the original, blurred version and sharpened version.

Unsharp masking and High-boost filtering

High-boost filtering: a generalized unsharpened image.

$$f_{hb}(x, y) = Af(x, y) - \bar{f}(x, y)$$

- A : a parameter, $A \geq 1$. $A = 1$, hight-boost filtering $\rightarrow$ unsharp masking
- $f_s(x, y)$: sharpened image

Unsharp masking and High-boost filtering

High-boost filtering: Other representations.

$$
\begin{aligned}
f_{hb}(x,y) &= Af(x,y) - \bar{f}(x,y) \\
&= (A-1)f(x,y) + f(x,y) - \bar{f}(x,y) \\
&= (A-1)f(x,y) + f_s(x,y) \qquad (3)
\end{aligned}
$$

Replace Eq. (3) with Laplacian sharpening filter:

$$
f_{hb}(x,y) = \begin{cases} Af(x,y) - \nabla^2 f(x,y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is negative} \\ Af(x,y) + \nabla^2 f(x,y) & \text{if the center coefficient of the} \\ & \text{Laplacian mask is positive} \end{cases}
$$

Unsharp masking and High-boost filtering

High-boost filtering: can be implemented as a linear filter, with one of the following kernels.

$$H_{hb8} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & A+8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \Bigg\| \quad H_{hb4} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & A+4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

A is a parameter:

1. $A = 0$, hight-boost filter becomes standard Laplacian filter
2. $A = 1$, hight-boost filter becomes standard Laplacian sharpening filter
3. A increases pass 1 the contribution of sharpening process becomes less and less important

Unsharp masking and High-boost filtering

Applications of High-boost filtering:

- High-boost filtering can sharpen images
- High-boost filtering can brighten dark images

Unsharp masking and High-boost filtering: Examples

a
b
c
d
e

FIGURE 3.40 (a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.