CSE 185

Sharpening Filters

Digital differentiation

- Initially focus on one-dimensional derivatives

- Interested in behavior of these derivatives (functions)
  - in areas of constant intensity
  - at the onset and end of discontinuities (step and ramp discontinuities)
  - along intensity ramps

  These kinds of discontinuities can be used to model noise points, lines, and edges in images

- Derivatives of a digital function are defined in terms of differences
  - Different ways to define these differences

- Require any definition for first derivative ("rate of change")
  1) must be zero in areas of constant intensity
  2) must be nonzero at the onset of an intensity step or ramp
  3) must be nonzero along ramps

- Require any definition for second derivative ("rate of change of rate of change")
  1) must be zero in constant areas
  2) must be nonzero at the onset and end of an intensity step or ramp
  3) must be zero along ramps of constant slope

Basic definition of first-order derivative of a one-dimensional function $f(x)$ is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

(note partial derivative)
$\frac{\partial f}{\partial x} = \frac{df}{dx}$ is f ~~is~~ is only function of x

Basic definition of second-order derivative of $f(x)$ is the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Easily shown that these definitions satisfy requirements above.

Figure 3.36 from text.
  Function contains
  - intensity ramp
  - three sections of constant intensity
  - intensity step

Derivatives ~~are~~ involve "look ahead" operation

Note: sign of second derivative changes at onset and end of step or ramp
  → zero ~~sec~~ crossing property of second derivative is very useful for finding edges (localizing)

Some conclusions from Fig. 3.36:

- Edges in images are often ramp-like
  - → first derivative often results in thick edges
  - → second derivative often produces a "double" edge

→ Second derivative enhances fine detail much better than first derivative, a property suited for sharpening images
2nd derivatives also easier to implement
So, we will focus on 2nd derivative initially for sharpening.

Using the Second Derivative for Image Sharpening - The Laplacian.

Implement 2-D, second order derivative as a linear spatial filter

Want our derivative to be isotropic
- response is independent of direction of discontinuities in image
- "rotation invariant" → rotating the image and applying the filter gives the same result as applying the filter to the image and rotating the result.

It can be shown that the simplest isotropic derivative operator is the Laplacian, which, for a function $f(x,y)$ of two variables is defined as

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

Extending our previous discretization of the 2nd derivative to 2 dimensions, we get

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

and $$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y).$$

So, $$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Does the form of this equation look familiar?

The Laplacian can be implemented as a linear spatial filter using the following mask

0	1	0
1	-4	1
0	1	0

This will give an isotropic result for rotations in increments of 90°.

Fig. 3.37 - other implementation of Laplacian.

Think about what Laplacian filtered images look like?
- Regions with slowly varying intensity levels will get low values
- Regions of high intensity changes will get large (positive and negative) values.
→ areas of detail have been emphasized

Can sharpen original image simply by adding Laplacian filtered image to original:

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

Fig. 3.38 in text.

Matlab example.

Unsharp Masking and Highboost Filtering

Unsharp masking is a process that has been used for many years by the printing and publishing industry to sharpen images.

Uses a smoothed (unsharp) image:

1) Blur the original image.
2) Subtract the blurred image from the original. The resulting difference is called the mask.
3) Add the mask to the original.

Let $\bar{f}(x,y)$ denote the blurred image

$$g_{mask} = f(x,y) - \bar{f}(x,y).$$

Add a weighted portion of mask back to original image:

$$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

$k \geq 0$

$k=1$ → unsharp masking
$k>1$ → high boost filtering
$k<1$ → deemphasize contribution of unsharp mask.

Figure 3.39

Example Fig. 3.40

photoshop → unsharp mask

First-Order Derivatives – The Gradient

First derivatives in image processing are implemented using the magnitude of the gradient.

The gradient is a vector that points in the direction of the greatest rate of change of f at location (x,y)

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnitude (length) of vector ∇f

$$M(x,y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

is value of rate of change in the direction of the gradient vector.

$M(x,y)$ is an image of the same size of $f \rightarrow$ gradient image or gradient

Often, approximate gradient as:

$$M(x,y) \simeq |g_x| + |g_y|$$

Discrete approximations of the gradient:

Consider pixel neighborhood of (x,y)

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$z_5 = f(x,y)$
$z_1 = f(x-1, y-1)$
...

Simplest approximation:

$$g_x = (z_8 - z_5)$$

$$g_y = (z_6 - z_5)$$

Can be implemented with masks

$$\begin{array}{|c|} \hline -1 \\ \hline 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|} \hline -1 & 1 \\ \hline \end{array}$$

So

$$M(x,y) = \sqrt{(z_8 - z_5)^2 + (z_6 - z_5)^2}$$

or $M(x,y) \approx |(z_8 - z_5)| + |(z_6 - z_5)|$

Alternate, is Roberts cross-difference operators

$$g_x = (z_9 - z_5) \quad \text{and} \quad g_y = (z_8 - z_6)$$

$$M(x,y) = \sqrt{(z_9 - z_5)^2 + (z_8 - z_6)^2}$$

or

$$M(x,y) = |(z_9 - z_5)| + |(z_8 - z_6)|$$

Implement by masks

-1	0
0	1

and

0	-1
1	0

Masks of even sizes are awkward to implement because they do not have a center of symmetry.

Approximations to g_x and g_y using 3×3 neighborhood

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Can be implemented with masks

-1	-2	-1
0	0	0
1	2	1

and

-1	0	1
-2	0	2
-1	0	1

Sobel operators

Values of 2 in center coefficient is to achieve some smoothing by giving more importance to center point

Fig. 3.42

Matlab script.