2. Image Processing and Compositing

Reading

Recommended:

- Jain, Kasturi, Schunck. *Machine Vision*. McGraw-Hill, 1995. Chapter 4 and Sections 5.1–5.4.
- Castleman. *Digital Image Processing*. Prentice-Hall, 1979. Chapter 1.
- Thomas Porter and Tom Duff, Compositing digital images. SIGGRAPH '84, 253-259.
- Foley, et al., Section 17.6.
- Woo, et al., pages 214–222.

Outline of lecture

- Image processing
  - Contrast enhancement
  - Noise reduction
  - Edge detection
- Image compositing
  - The alpha channel
  - Deriving the compositing algebra
  - The various compositing operators
  - Compositing arithmetic

Image processing

Definition: *Image processing* is an operation that takes an image into an image.

Useful for:

- Enhancing contrast
- Removing noise
- Enhancing edges
- Rescaling images

Image analysis

Definition: *Image analysis* is an operation that takes an image into anything else.

Examples:

- Extracting measurements
- Recovering edges or geometry
- Image compression

(The term "image processing" is sometimes loosely used to cover both processing and analysis.)

Images and digital images

Q: What is an "image"?

Q: What is a "digital image"?

Issues to think about:

- Does an image have to be "visible"?
- Does an image have to be formed optically?
- Does an image have to be 2D?
- How is a digital image different from an image?
- Must a digital image be discrete?

Contrast enhancement

Image with poor contrast (Jain, fig 4.1)

Often an acquired image will have a poor distribution of gray values.

Histogram scaling:

- Simplest solution
- Rescale pixels from old range to new one

Q: What is a problem with this approach?

Histogram equalization:

- Equalize the number of pixels at each gray level

Q: How?

Comparison of contrast enhancement methods

Comparison between histogram scaling and histogram equalization (Jain, fig 4.2, bottom row of fig 4.3)

Noise reduction

Different types of noise (Jain, fig 4.5)

Some common types of noise include:

- *Salt and pepper noise*: contains random occurrences of black and white pixels.
- *Impulse noise*: contains random occurrences of white pixels.
- *Gaussian noise*: contains variations in intensity drawn from a Gaussian or normal distribution.

Space invariant filters

Many image processing operations can be modeled as a *linear system*:

$$\delta(x,y) \longrightarrow \boxed{\text{Linear system}} \longrightarrow g(x,y)$$

Here, $g(x,y)$ is called the *impulse response*.

If the response is the same regardless of position, then you have a linear *space invariant* (LSI) system:

$$\delta(x-x_0,y-y_0) \longrightarrow \boxed{\text{LSI system}} \longrightarrow g(x-x_0,y-y_0)$$

Q: Given an LSI with impulse response $g(x,y)$, what is the effect of the LSI on an arbitrary image $f(x,y)$?

Convolution

For an LSI, the output $h(x,y)$ is the *convolution* of the input $f(x,y)$ with the impulse response $g(x,y)$:

$$\begin{aligned} h(x,y) &= f(x,y) \star g(x,y) \\ &= \int\int f(x',y')\, g(x-x', y-y')\, dx'\, dy' \end{aligned}$$

Q: What is the analog for discrete functions?

Discrete convolution

For discrete images, the impulse response $g[i,j]$ is called a *convolution mask*.

Convolution corresponds to taking weighted sums over the pixels in an image — called *convolution filtering*.

Example: What is $f \star g$, where

$$f = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad g = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} ?$$

Mean filters

We just looked at the simplest example of a *mean filter*:

$$\frac{1}{k^2}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ & & & \ddots & \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

Q: What are mean filters good for?

Q: What is the effect of varying k?

Effect of mean filters

Mean filters, $k = 3, 5, 7$ (Jain, fig 4.5(c,d,e), fig 4.7)

Median filters

A *median filter* performs a similar but different operation under its $k \times k$ window: it selects the median pixel in the window.

Q: What advantage does a median filter have over a mean filter?

Q: Is a median filter a linear filter?

Q: Is a median filter a spatially invarant filter?

Effect of median filters

Median filters, $k = 3, 5, 7$ (Jain, fig 4.5(c,d,e), fig 4.10)

Gaussian filters

A 2D Gaussian (Jain, fig 4.11)

Choosing weights according to a Gaussian often does a nicer job than mean filters:

$$g[i,j] \;=\; e^{-(i^2+j^2)/(2\sigma^2)}$$

Gaussians provide a nice tradeoff between blurring and preserving features in the image.

Gaussian filters, cont.

Here are some more of their nice properties:

- Rotationally symmetric
- Degree of smoothing easily controlled (by σ)
- Separable (equivalent to convolving in x then y with 1D kernels)
- Cascading (Gaussian $\star$ Gaussian = Gaussian)
- Easily approximated with rows of Pascal's triangle; e.g.,

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Edge detection

Edge detection is

- Easy for humans
- Not so easy for computers

It's also

- Fundamental in computer vision
- Pretty useful in graphics too

What is an edge?

Some ideal edge profiles (Jain, fig 5.1)

Some real edge profiles (Jain, fig 5.2)

Q: How might you detect an edge in 1D?

Gradients

The *gradient* is the 2D equivalent of the first derivative:

$$\nabla f(x,y) \;=\; \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

Properties of the gradient:

- Points in direction of maximum increase in f
- Magnitude gives rate of increase along this direction

Image gradients

Q: What is the discrete analog for images?

Q: How can this be implemented with convolution masks?

Q: What is the result?

Steps in edge detection

Algorithms for edge detection contain three steps:

1. Filtering

   Q: Why?

2. Enhancement

   · Usually through gradient magnitude

3. Detection

   · Usually via thresholding

4. Localization (optional)

   · Estimating subpixel resolution and orientation

Enhancement step

One of the most common measures of gradient magnitude is the *Sobel operator*. Uses:

$$M = \sqrt{s_x^2 + s_y^2}$$

where the partials s_x and s_y are computed by:

$$s_x \;=\; \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_y \;=\; \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(Gives more emphasis to pixels near center of mask.)

Results of Sobel edge detection

(a) Original; (b) Filtered; (d) 2×2 gradient, $T = 64$; (f) Sobel, $T = 225$. (Jain, fig 5.4(a,b,d,f))

Second derivative operators

Comparison of first and second derivate approaches (Jain, fig 5.8)

Note that the Sobel operator gives thick edges anywhere the derivative is above threshold.

A better approach may be to look for peaks in the first derivative.

Corresponds to _____ in the second derivative.

Laplacian operator

The *Laplacian* is the 2D equivalent of the 2nd derivative:

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

The discrete equivalent is given by a difference of differences:

And here are the masks:

Laplacian, cont.

These two masks can be combined into a single operator:

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let's try it out on a couple of examples:

Note:

- Not all zeros are edges (**Q:** how do you tell?)
- Interpolation may be required to find exact edge.

Laplacian of Gaussian

The Mexican hat operator (Jain, fig 5.11)

The Laplacian and Gaussian operators can be combined:

$$\begin{aligned} h(x,y) &= \nabla^2[(g(x,y) \star f(x,y)] \\ &= \left[\nabla^2 g(x,y)\right] \star f(x,y) \end{aligned}$$

The resulting filter is sometimes called the "Mexican hat" operator.

Laplacian of Gaussian, cont.

Here's a 5×5 version of the Mexican hat:

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

And here are some results:

Results of LoG (Jain, fig 5.13)

Image compositing

Often, small programs work better than big ones:

- Example: Unix

The same is true for rendering a picture:

- Want different small programs for different objects
- Small errors should not force "recompilation" of entire image

Image compositing, cont.

Therefore, you want to separate an image into "elements":

- Rendered independently
- "Composited" together

Each element has its own "matte" with coverage information.

Goals:

- Do not want to induce aliasing:
  want to retain soft edges in elements
- Want to exploit associativity, allowing composites in any order
- Want to do dissolves and fades

The "alpha" channel

We encode the matte (or coverage) information in a separate channel, called "alpha":

$$\begin{array}{rl} \alpha = 0 & \text{no coverage} \\ \alpha = 1 & \text{full coverage} \\ 0 < \alpha < 1 & \text{partial coverage} \end{array}$$

The alpha will control the linear interpolation of foreground and background colors when one element is placed over another.

Relationship between alpha and RGB:

- Alpha computed at same time
- Alpha resolution needs to be comparable
- Alpha treated in almost exactly the same way

The meaning of alpha

Q: How do we express a pixel that is half covered by a full red object?

Porter-Duff solution: Use *pre-multiplied* value in the color component:

The meaning of alpha, cont.

In general, (r, g, b, α) indicates a pixel that is α covered by the color $(r/\alpha,\ g/\alpha,\ b/\alpha)$.

This convention allows you to draw the (r, g, b) components directly and see antialiased edges.

Q: What is the meaning of the values:

- $(0, 0, 0, 1)$?
- $(0, 0, 0, 0)$?
- $(1, 1, 1, 1/2)$?

The compositing algebra: Assumptions

We now want to start mixing images together, based on alpha.

We do not have exact coverage information in each pixel: just overall alpha.

So we need to make some assumptions.

We'll consider two cases:

1. The overlap of two semi-transparent objects like haze
2. The overlap of two opaque hard-edged elements

Case 1: Semi-transparent objects

Let α_A and α_B be the opacities of A and B.

Suppose we put A and B over a background G.

Questions:

- How much of G shows through both A and B?
- How much of G is blocked by A but passed by B?
- How much of G is blocked by B but passed by A?
- How much of G is blocked by A and B?

Case 2: Opaque objects

Let α_A and α_B be the coverages of two opaque objects A and B within a pixel.

So A divides pixel into subpixels areas of ratio $\alpha_A : 1 - \alpha_A$.

Compositing assumption:

No correlation between A and B.

So A divides B into areas of ratio:

Case 2, cont.

Q: What are the resulting areas of:

- $\overline{A} \cap \overline{B}$:
- $A \cap \overline{B}$:
- $\overline{A} \cap B$:
- $A \cap B$:

Q: When do these assumptions break down?

Compositing operators

Two pictures divide a pixel into 4 subareas:

Name	*Description*	*Choices*
0	$\overline{A} \cap \overline{B}$	0
A	$A \cap \overline{B}$	$0, A$
B	$\overline{A} \cap B$	$0, B$
AB	$A \cap B$	$0, A, B$

Each of these areas can be filled in with different choices, listed above.

Q: How many choices in all?

Compositing operators, cont.

Let's look at the 12 choices for filling in regions 0, A, B, and AB:

1. $(0, 0, 0, 0)$:
2. $(0, A, 0, A)$:
3. $(0, 0, B, B)$:
4. $(0, A, B, A)$:
5. $(0, A, B, B)$:
6. $(0, 0, 0, A)$:
7. $(0, 0, 0, B)$:
8. $(0, A, 0, 0)$:
9. $(0, 0, B, 0)$:
10. $(0, 0, B, A)$:
11. $(0, A, 0, B)$:
12. $(0, A, B, 0)$:

Blending alphas

Let F_A and F_B be the fractions of the original mattes of A and B that prevail in the composite.

Q: What is the alpha value of the composite?

Blending color components

Let C_A, C_B, C_O be the "true" color components of A, B, and the composite, and let c_A, c_B, c_O be the pre-multiplied versions (which we want to compute).

So:

$$c_O = \alpha_O C_O$$

$$= \alpha_O \text{———————}$$

Finding the fractions

Now we just need to find F_A and F_B.

These are easily found by inspection, from the compositing diagrams.

Examples:

Operation	F_A	F_B
clear		
A		
B		
A **over** B		
A **in** B		
A **"plus"** B		

Unary operators

A couple of unary operators are also useful:

$$darken(A, \phi) \equiv$$

$$dissolve(A, \delta) \equiv$$

Example

(*FFire* @plus@ (BFire @out@ Planet))
@over darken@(*Planet*, 0.8) @over@ *Stars*

Summary

What to take away from this lecture:

1. A richer understanding of the terms "image" and "image processing"
2. How contrast enhancement is done
3. How noise reduction is done
4. How convolution filtering works
5. The effect of mean, median, and Gaussian filters
6. What a separable filter is
7. How edge detection is done
8. What a gradient is and how it is computed
9. The reasons for doing compositing
10. The meaning of alpha
11. How the various compositing operations are defined
12. The definition and implications of the "compositing assumption"
13. How compositing operations are computed
14. How to use compositing in practice