Lecture 3

Image Sampling, Pyramids, and Edge Detection

© UW CSE vision faculty
Motivation

This image is too big to fit on the screen.

How can we shrink it to half its size?
Image sub-sampling

Throw away every other row and column to create a smaller image - called *image sub-sampling*
Image sub-sampling

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

Waarom zo jagged kijk ik?*

*Dutch for “Why do I look so jagged?”*
Artifacts arising from bad sampling

Jagged profiles

• represents a sampled location

Missing details
Artifacts arising from bad sampling

Checkers should get smaller as distance increases.
Moire patterns

http://www.wfu.edu/~matthews/misc/DigPhotog/alias/

http://www.daube.ch/docu/glossary/moiree.html
Aliasing can arise when you sample a continuous signal or image

- occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an alias
- formally, the image contains structure at different scales
  - called “frequencies” in the Fourier domain
- the sampling rate must be high enough to capture the highest frequency in the image

To avoid aliasing:

- sampling rate \( \geq 2 \times \text{max frequency in the image} \)
  - said another way: \( \geq \) two samples per cycle
- This minimum sampling rate is called the Nyquist rate
2D example

Good sampling

Bad sampling
Avoiding aliasing in sub-sampling

To avoid aliasing, sampling rate $\geq 2 \times$ max frequency in the image (this is called the **Nyquist rate**)

Therefore, can avoid aliasing in sub-sampling by reducing the max frequency in the image.

How?
Sub-sampling with Gaussian pre-filtering

Blur the image (low pass filter) the image, *then* subsample

- Blur using Gaussian filter
- Filter size should double for each $\frac{1}{2}$ size reduction (Nyquist)
Sub-sampling with Gaussian pre-filtering

Gaussian 1/2  G 1/4 (2xzoom)  G 1/8 (4xzoom)
Compare with...

1/2

1/4 (2x zoom)

1/8 (4x zoom)
Images by Dave Etchells of Imaging Resource using the Canon D60 (with an antialias filter) and the Sigma SD-9 (which has no antialias filter). The bands below the fur in the image at right are the kinds of artifacts that appear in images when no antialias filter is used. Sigma chose to eliminate the filter to get more sharpness, but the resulting apparent detail may or may not reflect features in the image.
Some times we want many resolutions

Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]
  - In computer graphics, a *mip map* [Williams, 1983]
  - A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision
  - Texture synthesis, compression, feature detection, object recognition
Gaussian pyramid construction

Repeat
  • Filter w/ Gaussian
  • Subsample

Until minimum resolution reached
  • can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!
Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?

Recall how a digital image is formed

\[ F[x, y] = \text{quantize}\{f(xd, yd)\} \]

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale
Image resampling and interpolation
Image resampling and interpolation

So what to do if we don’t know $f$

- Answer: guess an approximation $\tilde{f}$
- Can be done in a principled way: filtering

Image reconstruction

- Define a continuous function based on $F$
  \[ f_F(x) = F\left(\frac{x}{d}\right) \text{ when } \frac{x}{d} \text{ is an integer, 0 otherwise} \]
- Reconstruct by cross-correlation with filter $h$:
  \[ \tilde{f} = h \otimes f_F \]
Interpolation filters

What does the 2D version of this hat function look like?

\[ h(x) \quad h(x, y) \]

- performs linear interpolation
- (tent function) performs \textbf{bilinear interpolation}

Better filters give better resampled images
- Bicubic is common choice
  - fit 3\textsuperscript{rd} degree polynomial surface to pixels in neighborhood
  - can also be implemented by a convolution
Bilinear interpolation (exercise)

A simple method for resampling images

\[ f(x, y) = \begin{array}{c}
\frac{a}{a+b} f[i, j] \\
\frac{b}{a+b} f[i+1, j]
\end{array} \]

\[ \begin{array}{c}
\frac{a}{a+b} f[i+1, j] \\
\frac{b}{a+b} f[i+1, j+1]
\end{array} \]

\[ \begin{array}{c}
\frac{a}{a+b} f[i, j+1]
\end{array} \]
Okay, enough about sampling

Suppose we want to extract useful features from images

One type of feature: Edges
Why extract edges?

- Edges and lines are used in
  - object recognition
  - image matching (e.g., stereo, mosaics)
  - document analysis
  - horizon detection
  - line following robots
  - and many more apps

- More compact than pixels
Where do edges come from?

Edges in images are caused by a variety of factors:

- Surface normal discontinuity
- Depth discontinuity
- Surface color discontinuity
- Illumination discontinuity
Edge detection

How can you tell that a pixel is on an edge?
Images as functions…

Edges look like steep cliffs
The gradient of an image:

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid increase in intensity.

The gradient direction is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

• how does this relate to the direction of the edge?

The edge strength is given by the gradient magnitude

\[ ||\nabla f|| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
The discrete gradient

How can we differentiate a *digital* image $F[x,y]$?
The discrete gradient

How can we differentiate a digital image $F[x,y]$?

- Answer: take discrete derivative (“finite difference”)

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

How would you implement this as a cross-correlation?
The Sobel operator

Better approximations of the derivatives exist

- The Sobel operators below are very commonly used

\[
\begin{array}{ccc}
\frac{1}{8} & \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix} & \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{bmatrix} \\
\end{array}
\]

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term is needed to get the right gradient value, however
Edge detection using the Sobel operator
Effects of noise

Consider a single row or column of the image

• Plotting intensity as a function of position gives a signal

Where is the edge?
Solution: smooth first

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$
Derivative theorem of convolution

$$\frac{\partial}{\partial x} (h \ast f) = \left( \frac{\partial}{\partial x} h \right) \ast f$$

This saves us one operation:

Need to find (local) maxima of a function
Laplacian of Gaussian

Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

Where is the edge?  Zero-crossings of bottom graph
2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

Laplacian of Gaussian

\[ \nabla^2 h_\sigma(u, v) \]

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

\( \nabla^2 \) is the **Laplacian** operator:
LoG filter

Laplacian of Gaussian

\[ \nabla^2 h_\sigma(u, v) \]

Discrete approximation with \( \sigma = 1.4 \)

\[
\begin{array}{cccccccccc}
0 & 1 & 1 & 2 & 2 & 1 & 1 & 0 \\
1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\
1 & 4 & 5 & 3 & 3 & 5 & 4 & 1 \\
2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\
2 & 5 & 0 & -24 & -40 & -24 & 0 & 5 & 2 \\
2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\
1 & 4 & 5 & 3 & 3 & 5 & 4 & 1 \\
1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\
0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \\
\end{array}
\]
Edge detection using LoGs

original image (Lena)  LoG followed by zero crossing detection
Canny Edge Detector

1. **Smoothing**: Smooth the image with a Gaussian filter with spread $\sigma$
2. **Gradient**: Compute gradient magnitude and direction at each pixel of the smoothed image
3. **Thresholding**: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. **Non-maximum suppression (thinning)**: Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)
Step 4: Thinning (Non-maximum suppression)

Check if pixel is local maximum along gradient direction
- requires checking interpolated pixels p and r
Canny Edge Detector

1. **Smoothing**: Smooth the image with a Gaussian filter with spread $\sigma$
2. **Gradient**: Compute gradient magnitude and direction at each pixel of the smoothed image
3. **Thresholding**: Threshold the gradient magnitude image such that strong edges are kept and noise is suppressed
4. **Non-maximum suppression (thinning)**: Zero out all pixels that are not the maximum along the direction of the gradient (look at 1 pixel on each side)
5. **Tracing edges**: Trace high-magnitude contours and keep only pixels along these contours, so weak little segments go away
The Canny edge detector

original image (Lena)
The Canny edge detector: Step 2

Norm of the gradient of smoothed image

\[ \|
\n\text{\n\n\n\n\n} \|
\n\text{\n\n\n\n\n} = \sqrt{( \frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} \]
The Canny edge detector: Step 3

thresholding
The Canny edge detector: Steps 4 & 5

Thinning (single pixel edges) & tracing edges
Effect of $\sigma$ (Gaussian kernel spread/size)

The choice of $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features
Next Time: Intelligent image scissors

- Things to do:
  - Read intelligent scissors paper(s) online
  - Project 1 will be assigned next class
  - Prepare for C/C++ programming
  - Visit Vision and Graphics Lab (Sieg 327)
    - Your ID card should open Sieg 327

Lena (1972)  Lena in 1997