4. Image processing

Reading

Recommended:

- Jain, Kasturi, Schunck, *Machine Vision.*
  McGraw-Hill, 1995. Chapter 4 and Sections 5.1-5.4.

Image processing

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

Useful for:

- Enhancing contrast
- Removing noise
- Enhancing edges
- Warping (rotating, scaling, shearing, ...)

Image analysis

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

Examples:

- Extracting measurements
- Recovering edges or geometry
- Image segmentation

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

What is an image?

In the context of this course, an *image* is a continuous function over two dimensions:

$$f(x, y)$$

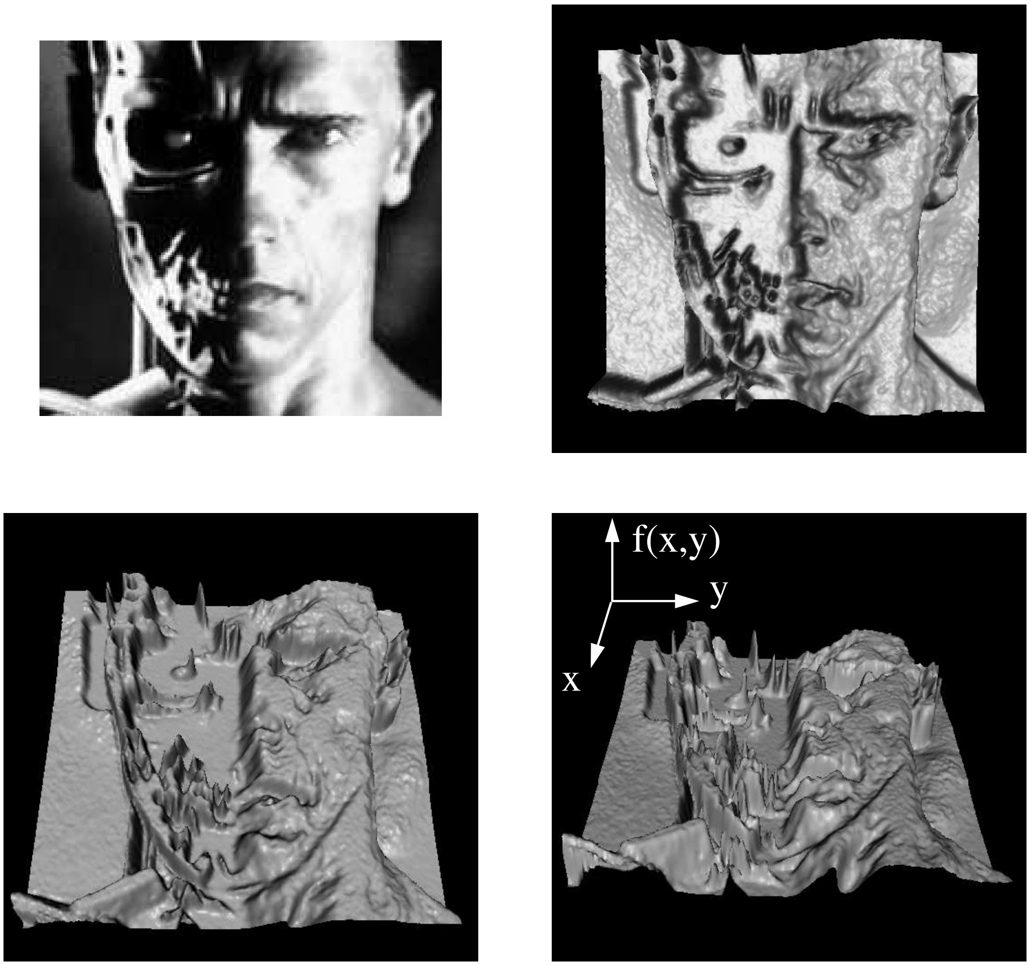
The image may represent light captured by a real or synthetic camera. It could also represent a drawing or painting or visualization of data.

A color image is a "vector-valued" function:

$$f(x, y) = \begin{pmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{pmatrix}$$

For now, we will concentrate on grayscale images, $f(x, y)$.

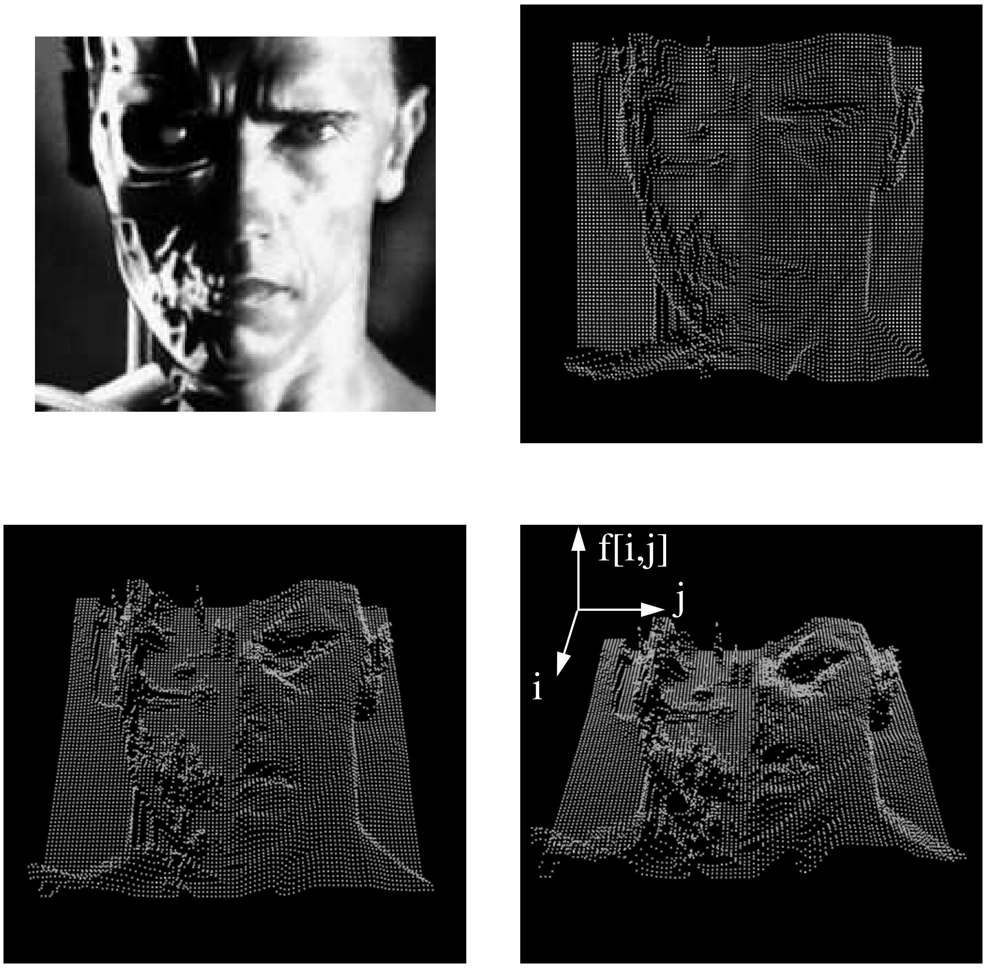
Images as functions



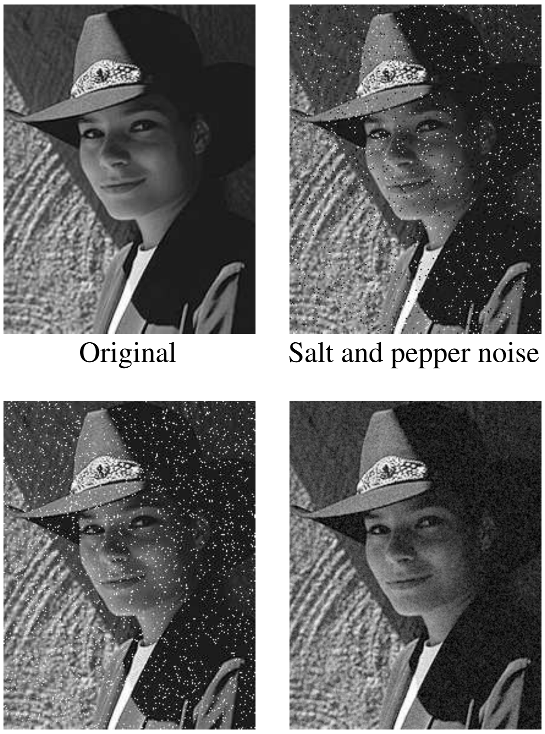
What is a digital image?

A *digital image* results from sampling and quantizing an image.

$$f[i,j] = \textbf{Quantize}\{f(i\Delta, j\Delta)\}$$



Noise



Original — Salt and pepper noise

Impulse noise — Gaussian noise

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 normal distribution.

Convolution

One of the most common methods for filtering an image is called *convolution*. In 1D, convolution is defined as:

$$
\begin{aligned}
h(x) &= f(x) * g(x) \\
&= \int_{-\infty}^{\infty} f(x')g(x-x')dx' \\
&= \int_{-\infty}^{\infty} f(x')\tilde{g}(x'-x)dx'
\end{aligned}
$$

where $\tilde{g}(x) = g(-x)$.

Example:

Discrete convolution

For a digital signal, we define *discrete convolution* as:

$$
\begin{aligned}
h[i] &= f[i] * g[i] \\
&= \sum_j f[j]g[j-i] \\
&= \sum_j f[j]\tilde{g}[i-j]
\end{aligned}
$$

where $\tilde{g}[i] = g[-i]$.

Example:

Convolution in 2D

In two dimensions, convolution becomes:

$$
\begin{aligned}
h(x, y) &= f(x, y) * g(x, y) \\
&= \int_{-\infty}^{\infty} f(x', y') g(x - x', y - y') dx' dy' \\
&= \int_{-\infty}^{\infty} f(x', y') \tilde{g}(x' - x, y - y') dx' dy'
\end{aligned}
$$

where $\tilde{g}(x, y) = g(-x, -y)$.

Similarly, discrete convolution in 2D is:

$$
\begin{aligned}
h[i, j] &= f[i, j] * g[i, j] \\
&= \sum_k \sum_l f[k, l] g[i - k, j - l] \\
&= \sum_k \sum_l f[k, l] \tilde{g}[k - i, l - j]
\end{aligned}
$$

where $\tilde{g}[i, j] = g[-i, -j]$.

Mean filters

One of the simplest filters we can design is the *mean filter*. It takes the form:

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

Q: What are mean filters good for?

Q: What is the effect of varying k?

Effect of mean filters

Mean filtering of various filter sizes applied to two types of noisy images

Gaussian filters

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 trade-off between blurring and preserving features in the image.

Effect of Gaussian filters

Gaussian filtering of various filter sizes applied to two types of noisy images

Median filters

A *median filter* operates over a $k \times k$ window by selecting the median pixel in the window.

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

Q: Is a median filter a kind of convolution filter?

Effect of median filters

Median filtering of various filter sizes applied to two types of noisy images

Edge detection

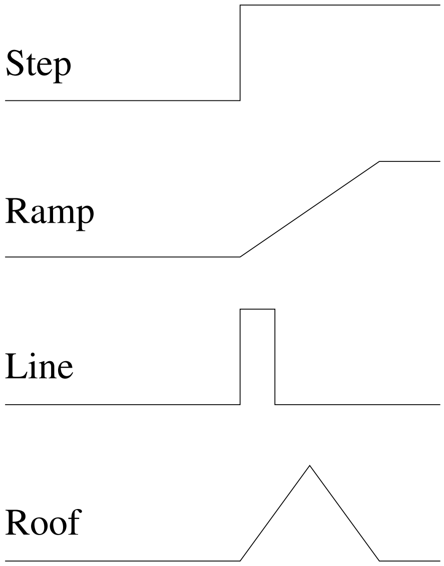
Edge detection is

- 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

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

Gradients

The *gradient* is the 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 the direction of maximum increase of f
- Magnitude gives the rate of increase along this direction

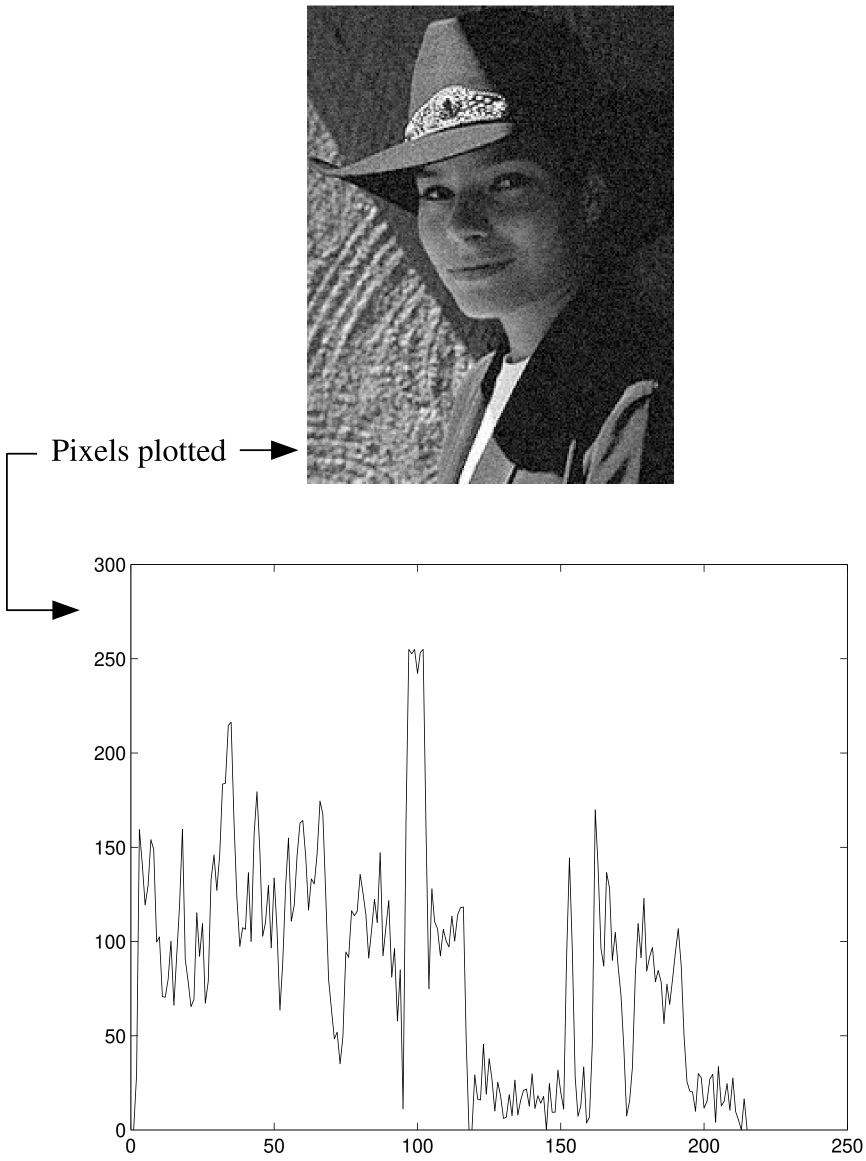
Image gradients

Q: What is the discrete analog for images?

Q: How can this be implemented with convolution masks?

Q: What is the result?

Less than ideal edges



Some real edge profiles

Steps in edge detection

Algorithms for edge detection contain 3 or 4 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

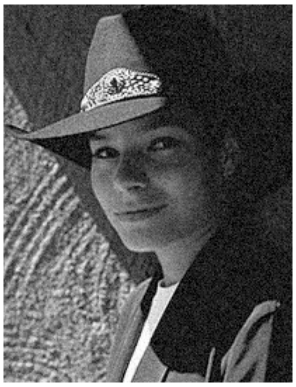
One of the most common measures of gradient magnitude is the *Sobel operator*. The partials s_x and s_y are computed by:

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

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

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

Results of Sobel edge detection



Original

Smoothed

Sx + 128

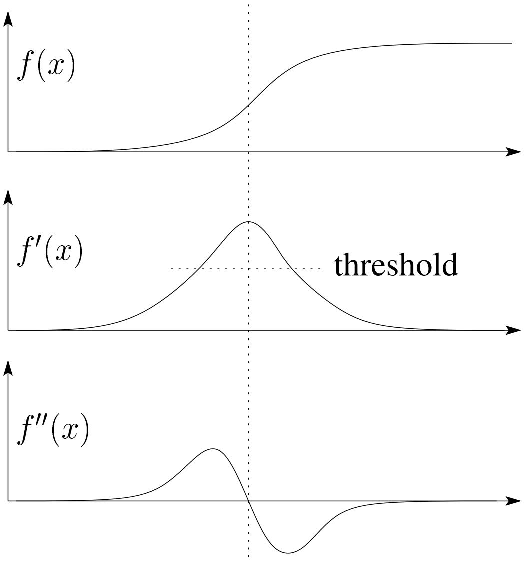
Sy + 128

Magnitude

Threshold = 64

Threshold = 128

Second derivative operators



Comparison of first and second derivative approaches

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

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

Q: A peak in the first derivative corresponds to what in the second derivative?

Localization with the Laplacian

An equivalent measure of the second derivative in 2D is the *Laplacian*:

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

Using the same arguments we used to compute the gradient filters, we can derive a Laplacian filter to be:

$$\Delta^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Zero crossings of this filter correspond to positions of maximum gradient. These zero crossings can be used to localize edges.

Summary

1. A richer understanding of the terms "image" and "image processing"
2. How noise reduction is done
3. How convolution filtering works
4. The effect of mean, Gaussian, and median filters
5. How edge detection is done
6. What a gradient is and how it is computed