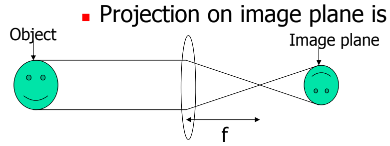
Computer Vision

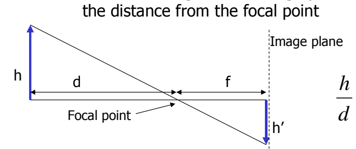
Optics

- Focal length
  - Length f of projection through lens on image plane
- Inversion
  - Projection on image plane is inverted



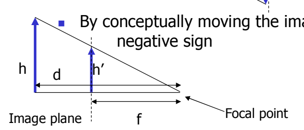
Projection on the image plane

- Size of an image on the image plane is inversely proportional to the distance from the focal point



$$\frac{h}{d} = \frac{-h'}{f}$$

- By conceptually moving the image plane, we can eliminate the negative sign

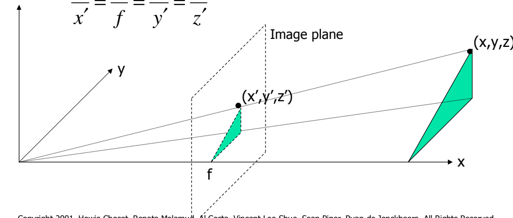


$$\frac{h}{d} = \frac{h'}{f}$$

Move to three dimensions

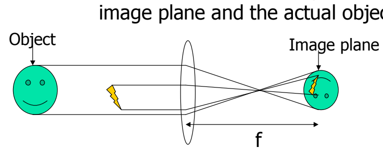
- Similarity holds when three dimensions are considered

$$\frac{x}{x'} = \frac{x}{f} = \frac{y}{y'} = \frac{z}{z'}$$



Perspective

- 1 Point Perspective
  - Using similar triangles, it is possible to determine the relative sizes of objects in an image
  - Given a calibrated camera (predetermine a mathematical relationship between size on the image plane and the actual object)



Preliminary Math: Trigonometry

- Pythagorean Theorem
  - $a^2 + b^2 = c^2$
- Law of sines

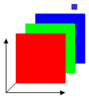
$$\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c}$$

- Law of cosines

$$c^2 = a^2 + b^2 - 2ab\cos\theta$$

Images

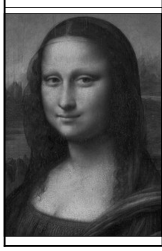
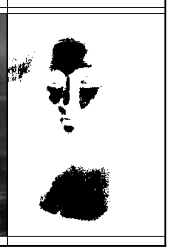
- Discrete representation of a continuous function
  - Pixel: Picture Element – cell of constant color in a digital image
  - An image is a two dimensional array of pixels
  - Pixel: numeric value representing a uniform portion of an image
- Grayscale
  - All pixels represent the intensity of light in an image, be it red, green, blue, or another color
    - Like holding a piece of transparent colored plastic over your eyes
  - Intensity of light in a pixel is stored as a number, generally 0..255 inclusive
- Color
  - Three grayscale images layered on top of eachother with each layer indicating the intensity of a specific color light, generally red, green, and blue (RGB)
  - Third dimension in a digital image



Images

- Resolution
  - Number of pixels across in horizontal
  - Number of pixels in the vertical
  - Number of layers used for color
    - Often measured in bits per pixel (bpp) where each color uses 8 bits of data
  - Ex: 640x480x24bpp
- Binary images: Two color image
  - Pixel is only one byte of information
  - Indicates if the intensity of color is above or below some nominal value
  - Thresholding

Grayscale vs. Binary image

Grayscale	Binary threshold
	

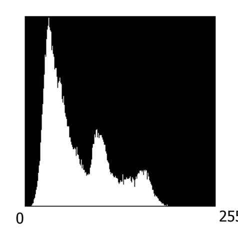
Thresholding

- Purpose
  - Trying to find areas of high color intensity
  - Highlights locations of different features of the image (notice Mona's eyes)
  - Image compression, use fewer bits to encode a pixel
- How done
  - Decide on a value μ
  - Scan every pixel in the image
    - If it is greater than μ, make it 255
    - If it is less than μ, make it 0
  - Picking a good μ
    - Often 128 is a good value to start with
    - Use a histogram to determine values based on color frequency features

Histogram

- Measure the number of pixels of different values in an image.
- Yields information such as the brightness of an image, important color features, possibilities of color elimination for compression
- Thresholding
  - Make pixels above a value one color and values below that value a different color
  - Binary threshold often used to transform a grayscale image into black and white
  - Also usable for compression and feature extraction

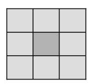
Mona's Histogram

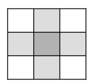


Connectivity

- Two conventions on considering two pixels next to each other

8 point connectivity	4 point connectivity
All pixels sharing a side or corner are considered adjacent	Only pixels sharing a side are considered adjacent





- To eliminate the ambiguity, we could define the shape of a pixel to be a hexagon

Object location - Segmentation

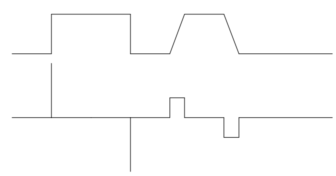
- One method of locating an object is through the use of a wave front
- Wavefront
  - Assume a binary image with values of 0 or 1
  - 1. Choose 1st pixel with value 1, make it a 2
  - 2. For each neighbor, if it is also a 1, make it a 2 as well
  - 3. Repeat step two for each neighbor until there are no neighbors with value 1
  - 4. All pixels with a value 2 are are a continuous object

Centroids

- Use the region filled image from above
- Compute the area of the region
  - Number of pixels with the same number value (n)
- Sum all of the x coords with the same pixel value. Do the same for y coords
- Divide each sum by n and the resulting x, y coord is the centroid

Edge detection

- Scanline: one row of pixels in an image
- Take the first derivative of a scanline



- The derivative becomes nonzero when an edge (pixels change values) is encountered

Implementing 1st derivative edge detection digitally

- Derivative is defined as $\lim_{x \to c} \frac{f(x)-f(c)}{x-c}$
- With a scan line, the run (x – c) is 1, and the rise (f(x) – f(c)) is B[m+1] – B[m]
- This becomes

  where I is the resulting image of edges

  $$I[m] = 1 \cdot B[m+1] + -1 B[m]$$

- This is really just a dot product of the vector [-1 1] repeated each pixel in the resulting image

$$I[m] = \begin{bmatrix} B[m] & B[m+1] \end{bmatrix} \bullet \begin{bmatrix} -1 & 1 \end{bmatrix}$$

More Math: Convolution

- This operation of moving a mask across an image has a name, called convolution
- In order to mathematically apply a filter to a signal, we must use convolution
  - If you know laplace transforms, this is a multiplication in the laplace domain

Convolution: Analog

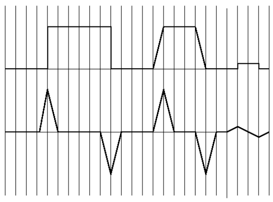
$$y(t)=\int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

Given a symmetric h (common in image processing), simplifies to

$$y(t)=\int_{-\infty}^{\infty} x(\tau)h(\tau)d\tau$$

h(t) = [-1 1]

Move across the signal x (possibly a scanline in an image)



Convolution: Digital

$$y[n]=\sum_{k=-\infty}^{\infty} x[n+k]h[k]$$

More useful in image processing on a digital computer

x[n] is a pixel in an image, y[n] is the resulting pixel

0	2	2	0	1	1	3

0	1	1

1\.

0	1	1

0	2	2	0	1	1	3

4				

2\.

0	1	1

0	2	2	0	1	1	3

4	2			

Convolution example, cont

3\.

0	1	1

0	2	2	0	1	1	3

4	2	1		

4\.

0	1	1

0	2	2	0	1	1	3

4	2	1	2	

5\.

0	1	1

0	2	2	0	1	1	3

4	2	1	2	4

Convolution: Two-dimensional

$$y(m,n)=\sum_{m_0}\sum_{n_0} x(m_0+m,n_0+n)h(m_0,n_0)$$

- Rotate your mask 180 degrees about the origin (if you were doing “correct” convolution, but since we are doing the “other” convolution, you can skip this step.
- Do the same dot product operation, this time using matrices instead of vectors
- Repeat the dot product for every pixel in the resulting image
- In the boundary case around the edges of the image there are two options
  - extend the original image out using the pixel values at the edge
  - Make the resulting image y smaller than the original and don’t compute pixels where the mask would extend beyond the edge of the original

Convolution: Old and New

- Analog

$$y(t)=\int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \qquad y(t)=\int_{-\infty}^{\infty} x(\tau+t)h(\tau)d\tau$$

- Digital

$$y[n]=\sum_{k=-\infty}^{\infty} x[k]h[n-k] \qquad y[n]=\sum_{k=-\infty}^{\infty} x[n+k]h[k]$$

- Two dimensional, digital

$$y(m,n)=\sum_{m_0}\sum_{n_0} x(m_0,n_0)h(m-m_0,n-n_0)$$

$$y(m,n)=\sum_{m_0}\sum_{n_0} x(m+m_0,n+n_0)h(m_0,n_0)$$

Filters, Masks, Transforms

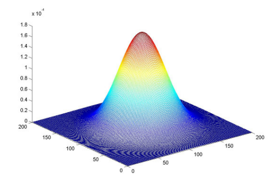
- Edge detection
  - Wide masks
- Smoothing
- Object detection

Filters, Masks, Transforms

- Gaussian (smooth)
- Wide Masks
  - Wide First Derivative
- Object detection

Gaussian Masks

- Used to smooth images and for noise reduction
- Use before edge detection to avoid spurious edges

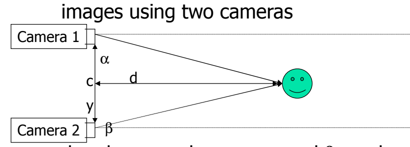


Wide Masks

- Wider masks lead to uncertainty about location of edge
- Can detect more gradual edges

Stereo Vision

- Way of calculating depth from two dimensional images using two cameras



- d and y are unknowns, α and β can be determined processing and c is known

$$\tan\alpha = \frac{d}{c-y} \qquad y = \frac{c\tan\alpha}{\tan\alpha - \tan\beta}$$

$$\tan\beta = \frac{d}{y} \qquad d = y\tan\beta$$