From Pixels to Features:
Review of Part 1

COMP 4900D
Winter 2006

Topics in part 1 – *from pixels to features*

- Introduction
  - what is computer vision? It's applications.
- Linear Algebra
  - vector, matrix, points, linear transformation, eigenvalue, eigenvector, least square methods, singular value decomposition.
- Image Formation
  - camera lens, pinhole camera, perspective projection.
- Camera Model
  - coordinate transformation, homogeneous coordinate, intrinsic and extrinsic parameters, projection matrix.
- Image Processing
  - noise, convolution, filters (average, Gaussian, median).
- Image Features
  - image derivatives, edge, corner, line (Hough transform), ellipse.

General Methods

- Mathematical formulation
  - Camera model, noise model
- Treat images as functions

$$I = f(x, y)$$

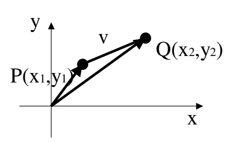
- Model intensity changes as derivatives $\nabla f = [I_x, I_y]^T$
  - Approximate derivative with finite difference.
- First-order approximation

$$I(i+u, j+v) \approx I(i, j) + I_x u + I_y v = I(i, j) + \begin{bmatrix} u & v \end{bmatrix} \nabla f$$

- Parameter fitting – solving an optimization problem

Vectors and Points

We use vectors to represent points in 2 or 3 dimensions



$$v = Q - P = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$$

The distance between the two points:

$$D = \|Q - P\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Homogeneous Coordinates

Go one dimensional higher:

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix}$$

w is an arbitrary non-zero scalar, usually we choose 1.

From homogeneous coordinates to Cartesian coordinates:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \rightarrow \begin{bmatrix} x_1/x_4 \\ x_2/x_4 \\ x_3/x_4 \end{bmatrix}$$

2D Transformation with Homogeneous Coordinates

2D coordinate transformation:

$$p'' = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

2D coordinate transformation using homogeneous coordinates:

$$\begin{bmatrix} p_x'' \\ p_y'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi & T_x \\ -\sin\phi & \cos\phi & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$

Eigenvalue and Eigenvector

We say that x is an eigenvector of a square matrix A if

$$Ax = \lambda x$$

λ is called <u>eigenvalue</u> and x is called <u>eigenvector</u>.

The transformation defined by A changes only the magnitude of the vector x

Example:

$$\begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} = 5 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix} = 2 \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

5 and 2 are eigenvalues, and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 2 \\ -1 \end{bmatrix}$ are eigenvectors.

Symmetric Matrix

We say matrix A is <u>symmetric</u> if

$$A^T = A$$

Example: $B^T B$ is symmetric for any B, because

$$(B^T B)^T = B^T (B^T)^T = B^T B$$

A symmetric matrix has to be a square matrix

Properties of symmetric matrix:
- has real eignvalues;
- eigenvectors can be chosen to be orthonormal.
- $B^T B$ has positive eigenvalues.

Orthogonal Matrix

A matrix A is orthogonal if

$$A^T A = I \quad \text{or} \quad A^T = A^{-1}$$

The columns of A are orthogonal to each other.

Example:

$$A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad A^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Least Square

When m>n for an m-by-n matrix A, $Ax = b$ has no solution.

In this case, we look for an approximate solution.
We look for vector x such that

$$\|Ax - b\|^2$$

is as small as possible.

This is the least square solution.

Least Square

Least square solution of linear system of equations

$$Ax = b$$

Normal equation: $A^T Ax = A^T b$

$A^T A$ is square and symmetric

The Least square solution $\bar{x} = (A^T A)^{-1} A^T b$

makes $\|A\bar{x} - b\|^2$ minimal.

SVD: Singular Value Decomposition

An $m \times n$ matrix A can be decomposed into:

$$A = UDV^T$$

U is $m \times m$, V is $n \times n$, both of them have orthogonal columns:

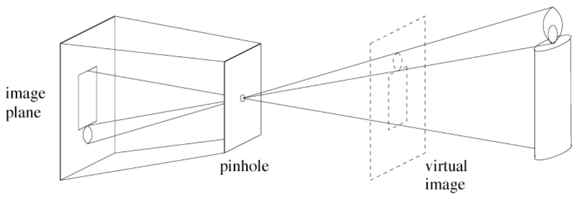
$$U^T U = I \qquad V^T V = I$$

D is an $m \times n$ diagonal matrix.

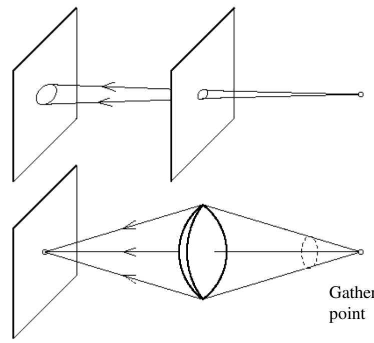
Example:

$$\begin{bmatrix} 2 & 0 \\ 0 & -3 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 3 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Pinhole Camera

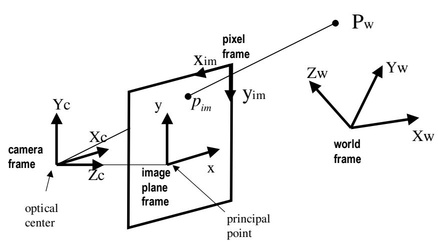


Why Lenses?



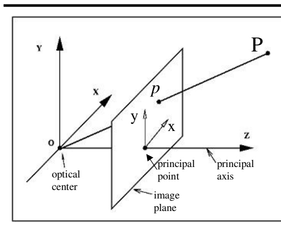
Gather more light from each scene point

Four Coordinate Frames



Camera model: $p_{im} = \begin{bmatrix} transformation \\ matrix \end{bmatrix} P_w$

Perspective Projection



$$x = f\frac{X}{Z} \quad y = f\frac{Y}{Z}$$

These are *nonlinear*.

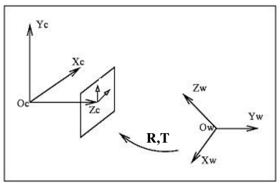
Using homogenous coordinate, we have a *linear* relation:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$x = u/w \quad y = v/w$$

World to Camera Coordinate

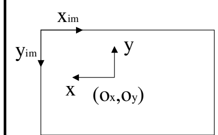
Transformation between the camera and world coordinates:



$$\mathbf{X}_c = \mathbf{R}\mathbf{X}_w + \mathbf{T}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Image Coordinates to Pixel Coordinates



$$x = (o_x - x_{im})s_x \quad y = (o_y - y_{im})s_y$$

s_x, s_y : pixel sizes

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Put All Together – World to Pixel

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$= \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K\begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\boxed{x_{im} = x_1 / x_3 \quad y_{im} = x_2 / x_3}$$

Camera Intrinsic Parameters

$$K = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$$

K is a 3x3 upper triangular matrix, called the **Camera Calibration Matrix**.

There are five intrinsic parameters:
(a) The pixel sizes in x and y directions s_x, s_y
(b) The focal length f
(c) The principal point (o_x,o_y), which is the point where the optic axis intersects the image plane.

Extrinsic Parameters

$$p_{im} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = K[R \quad T]\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

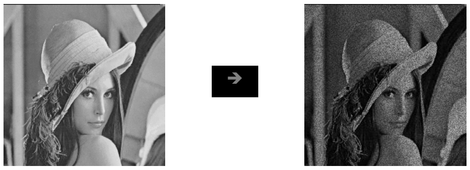
$[R|T]$ defines the **extrinsic parameters**.
The 3x4 matrix $M = K[R|T]$ is called the **projection matrix**.

Image Noise

Additive and random noise:

$$\hat{I}(x,y) = I(x,y) + n(x,y)$$

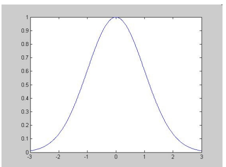
$I(x,y)$: the true pixel values
$n(x,y)$: the (random) noise at pixel (x,y)



Gaussian Distribution

Single variable

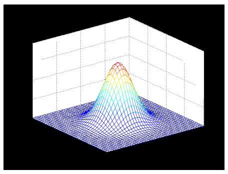
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$



Gaussian Distribution

Bivariate with zero-means and variance σ^2

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$



Gaussian Noise

Is used to model additive random noise

- The probability of n(x,y) is $e^{\frac{-n^2}{2\sigma^2}}$
- Each has zero mean
- The noise at each pixel is independent


Impulsive Noise

- Alters random pixels
- Makes their values very different from the true ones

Salt-and-Pepper Noise:

- Is used to model impulsive noise

$$I_{sp}(h,k) = \begin{cases} I(h,k) & x < l \\ i_{\min} + y(i_{\max} - i_{\min}) & x \geq l \end{cases}$$

x, y are uniformly distributed random variables

$l, i_{\min}, i_{\max}$ are constants

Image Filtering

Modifying the pixels in an image based on some function of a local neighbourhood of the pixels

N(p)

p

10	30	10
20	11	20
11	9	1

$\xrightarrow{f(p)}$

	5.7	

Linear Filtering – convolution

The output is the linear combination of the neighbourhood pixels

$$I_A(i,j) = I * A = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} A(h,k) I(i-h, j-k)$$

The coefficients come from a constant matrix A, called kernel.
This process, denoted by '*', is called (discrete) convolution.

Image:

1	3	0
2	10	2
4	1	1

$*$

Kernel:

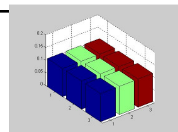
1	0	-1
1	0.1	-1
1	0	-1

=

Filter Output:

	5	

Smoothing by Averaging



$* \frac{1}{9}$

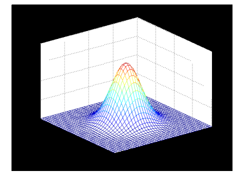
1	1	1
1	1	1
1	1	1

$=$

Convolution can be understood as weighted averaging.

Gaussian Filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right)$$

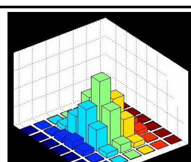


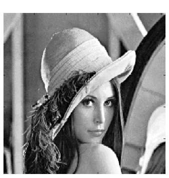
Discrete Gaussian kernel:

$$G(h,k) = \frac{1}{2\pi\sigma^2} e^{-\frac{h^2+k^2}{2\sigma^2}}$$

where $G(h,k)$ is an element of an $m \times m$ array

Gaussian Filter

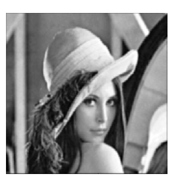




$* \frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

$=$



$$\sigma = 1$$

Gaussian Kernel is Separable

$$I_G = I * G =$$

$$= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} G(h,k) I(i-h, j-k) =$$

$$= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} e^{-\frac{h^2+k^2}{2\sigma^2}} I(i-h, j-k) =$$

$$= \sum_{h=-m/2}^{m/2} e^{-\frac{h^2}{2\sigma^2}} \sum_{k=-m/2}^{m/2} e^{-\frac{k^2}{2\sigma^2}} I(i-h, j-k)$$

since $e^{-\frac{h^2+k^2}{2\sigma^2}} = e^{-\frac{h^2}{2\sigma^2}} e^{-\frac{k^2}{2\sigma^2}}$

Gaussian Kernel is Separable

Convolving rows and then columns with a 1-D Gaussian kernel.

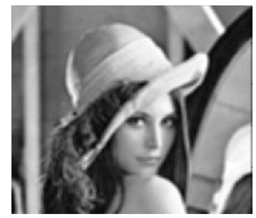
$$I * \frac{1}{38}\begin{bmatrix} 1 & 9 & 18 & 9 & 1 \end{bmatrix} = I_r$$

$$I_r * \frac{1}{38}\begin{bmatrix} 1 \\ 9 \\ 18 \\ 9 \\ 1 \end{bmatrix} = \text{result}$$

The complexity increases linearly with m instead of with m^2.

Gaussian vs. Average

Gaussian Smoothing



Smoothing by Averaging

Nonlinear Filtering – median filter

Replace each pixel value $I(i, j)$ with the median of the values found in a local neighbourhood of (i, j).

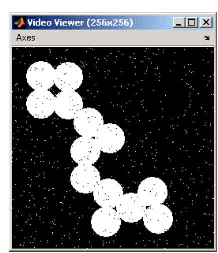
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

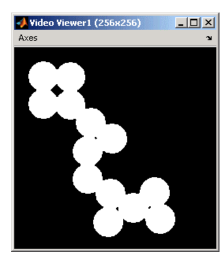
115, 119, 120, 123, 124, 125, 126, 127, 150

Median value: 124

Median Filter



Salt-and-pepper noise

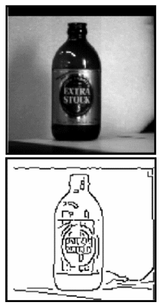


After median filtering

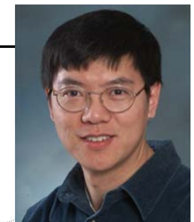
Edges in Images

Definition of edges

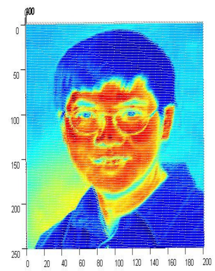
- Edges are significant local changes of intensity in an image.
- Edges typically occur on the boundary between two different regions in an image.



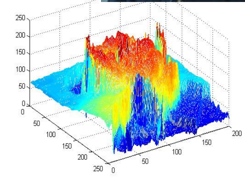
Images as Functions



2-D



Red channel intensity



$I = f(x, y)$

Finite Difference – 2D

Continuous function:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h \to 0} \frac{f(x+h,y) - f(x,y)}{h}$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{h \to 0} \frac{f(x,y+h) - f(x,y)}{h}$$

Discrete approximation:

$$I_x = \frac{\partial f(x,y)}{\partial x} \approx f_{i+1,j} - f_{i,j}$$

$$I_y = \frac{\partial f(x,y)}{\partial y} \approx f_{i,j+1} - f_{i,j}$$

Convolution kernels:

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Image Derivatives

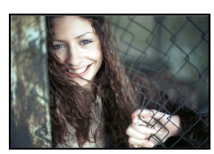
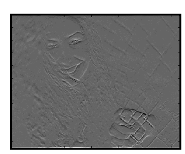


Image I

$I_x = I * \begin{bmatrix} -1 & 1 \end{bmatrix}$



$I_y = I * \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Edge Detection using Derivatives

1-D image $f(x)$

1st derivative $f'(x)$

$|f'(x)|$ threshold

Pixels that pass the threshold are edge pixels

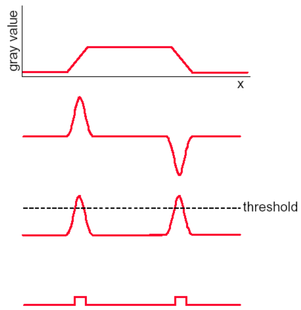
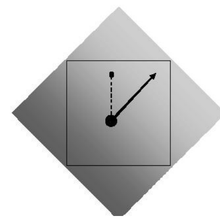


Image Gradient



gradient

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

magnitude

$$\|\nabla f\| = \sqrt{\left(\tfrac{\partial f}{\partial x}\right)^2 + \left(\tfrac{\partial f}{\partial y}\right)^2}$$

direction

$$\arctan\left(\tfrac{\partial f}{\partial y} / \tfrac{\partial f}{\partial x}\right)$$

Finite Difference for Gradient

Discrete approximation:

$$I_x(i,j) = \frac{\partial f}{\partial x} \approx f_{i+1,j} - f_{i,j}$$

$$I_y(i,j) = \frac{\partial f}{\partial y} \approx f_{i,j+1} - f_{i,j}$$

Convolution kernels:

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

magnitude $G(i,j) = \sqrt{I_x^2(i,j) + I_y^2(i,j)}$

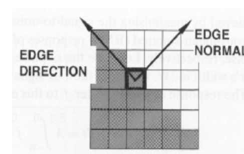
aprox. magnitude $G(i,j) \approx |I_x| + |I_y|$

direction $\arctan(I_y / I_x)$

Edge Detection Using the Gradient

Properties of the gradient:

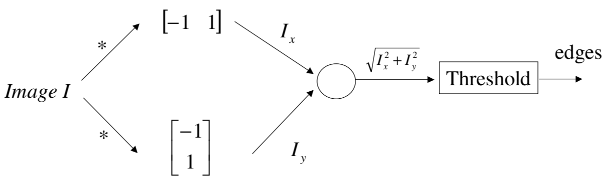
- The magnitude of gradient provides information about the strength of the edge
- The direction of gradient is always perpendicular to the direction of the edge



Main idea:

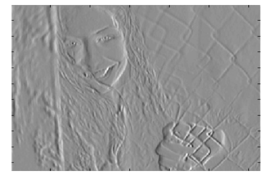
- Compute derivatives in x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

Edge Detection Algorithm



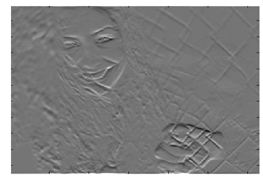
Edge Detection Example

I_x



I

I_y



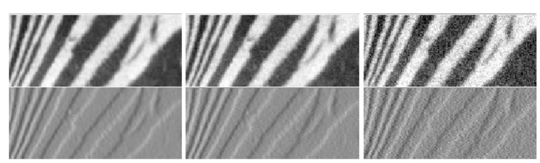
Edge Detection Example

$$G(i,j)=\sqrt{I_x^2(i,j)+I_y^2(i,j)}$$

I

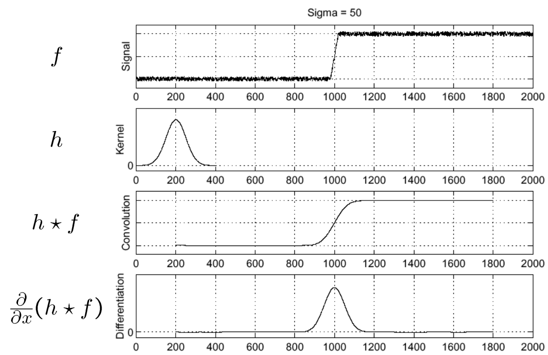
$$G(i,j) > Threshold = \tau$$

Finite differences responding to noise



Increasing noise ->
(this is zero mean additive gaussian noise)

Solution: smooth first



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

Sobel Edge Detector

Approximate derivatives with central difference

$$I_x(i,j) = \frac{\partial f}{\partial x} \approx f_{i-1,j} - f_{i+1,j}$$

Convolution kernel

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

Smoothing by adding 3 column neighbouring differences and give more weight to the middle one

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

Convolution kernel for I_y

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

Sobel Operator Example

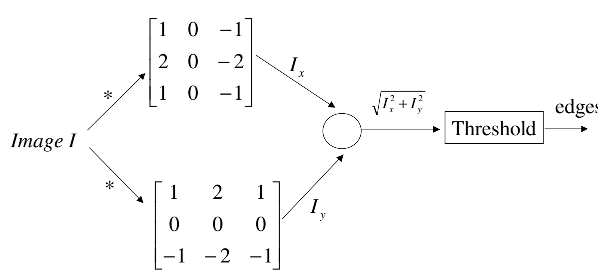
$$\begin{array}{|c|c|c|} \hline a_1 & a_2 & a_3 \\ \hline a_4 & \boxed{a_5} & a_6 \\ \hline a_7 & a_8 & a_9 \\ \hline \end{array} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\begin{array}{|c|c|c|} \hline a_1 & a_2 & a_3 \\ \hline a_4 & \boxed{a_5} & a_6 \\ \hline a_7 & a_8 & a_9 \\ \hline \end{array} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The approximate gradient at a_5

$$I_x = (a_1 - a_3) + 2(a_4 - a_6) + (a_7 - a_9)$$
$$I_y = (a_1 - a_7) + 2(a_2 - a_8) + (a_3 - a_9)$$

Sobel Edge Detector



Edge Detection Summary

Input: an image I and a threshold τ.

1. Noise smoothing: $I_s = I * h$
   (e.g. h is a Gaussian kernel)

2. Compute two gradient images I_x and I_y by convolving I_s with gradient kernels (e.g. Sobel operator).

3. Estimate the gradient magnitude at each pixel

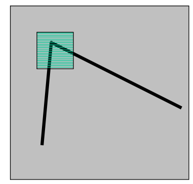
$$G(i, j) = \sqrt{I_x^2(i, j) + I_y^2(i, j)}$$

4. Mark as edges all pixels (i, j) such that $G(i, j) > \tau$

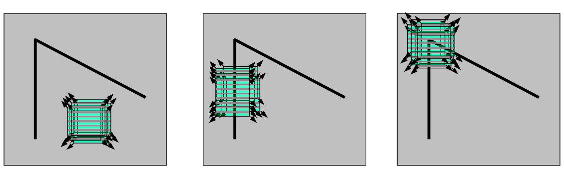
Corner Feature

Corners are image locations that have large intensity changes in more than one directions.

Shifting a window in *any direction* should give *a large change* in intensity



Harris Detector: Basic Idea



"flat" region: no change in all directions

"edge": no change along the edge direction

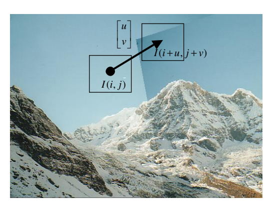
"corner": significant change in all directions

C.Harris, M.Stephens. "A Combined Corner and Edge Detector". 1988

Change of Intensity

The intensity change along some direction can be quantified by sum-of-squared-difference (SSD).

$$D(u, v) = \sum_{i,j} \left(I(i+u, j+v) - I(i, j) \right)^2$$



Change Approximation

If u and v are small, by Taylor theorem:

$$I(i+u, j+v) \approx I(i,j) + I_x u + I_y v$$

where $I_x = \frac{\partial I}{\partial x}$ *and* $I_y = \frac{\partial I}{\partial y}$

therefore

$$\begin{aligned}(I(i+u, j+v) - I(i,j))^2 &= (I(i,j) + I_x u + I_y v - I(i,j))^2 \\ &= (I_x u + I_y v)^2 \\ &= I_x^2 u^2 + 2 I_x I_y uv + I_y^2 v^2 \\ &= \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}\end{aligned}$$

Gradient Variation Matrix

$$D(u,v) = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

This is a function of ellipse.

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix C characterizes how intensity changes in a certain direction.

Eigenvalue Analysis

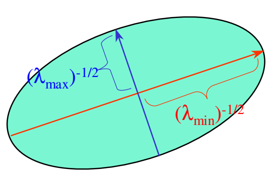
$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = Q^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q$$

If either λ is close to 0, then this is **not** a corner, so look for locations where both are large.

$C = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix}$

$$C = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = A^T A$$

- C is symmetric
- C has two positive eigenvalues



Corner Detection Algorithm

Algorithm

Input: image f, threshold t for λ_2, size of Q

(1) Compute the gradient over the entire image f

(2) For each image point p:

(2.1) form the matrix C over the neighborhood Q of p
(2.2) compute λ_2, the smaller eigenvalue of C
(2.3) if $\lambda_2 > t$, save the coordinates of p in a list L

(3) Sort the list in decreasing order of λ_2

(4) Scanning the sorted list top to bottom: delete all the points that appear in the list that are in the same neighborhood Q with p

Line Detection

The problem:
•How many lines?
•Find the lines.

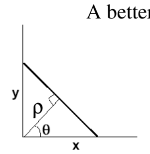
Equations for Lines

The slope-intercept equation of line

$$y = ax + b$$

What happens when the line is vertical? The slope a goes to infinity.

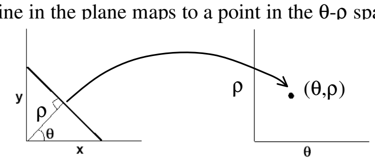
A better representation – the polar representation



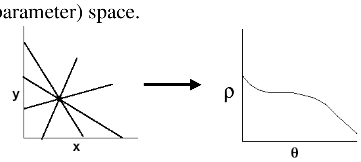
$$\rho = x\cos\theta + y\sin\theta$$

Hough Transform: line-parameter mapping

A line in the plane maps to a point in the θ-ρ space.

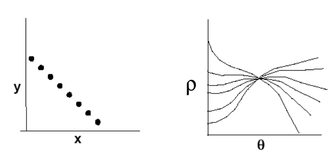


All lines passing through a point map to a sinusoidal curve in the θ-ρ (parameter) space.



$$\rho = x\cos\theta + y\sin\theta$$

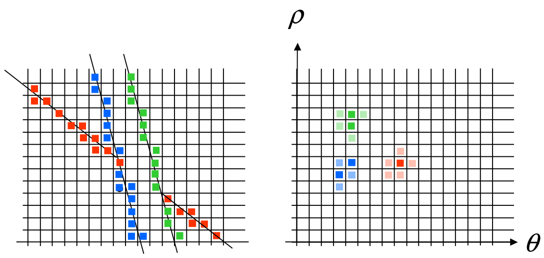
Mapping of points on a line



Points on the same line define curves in the parameter space that pass through a single point.

Main idea: transform edge points in x-y plane to curves in the parameter space. Then find the points in the parameter space that has many curves passing through.

Quantize Parameter Space



Detecting Lines by finding maxima / clustering in parameter space.

Examples

input image | Hough space | lines detected

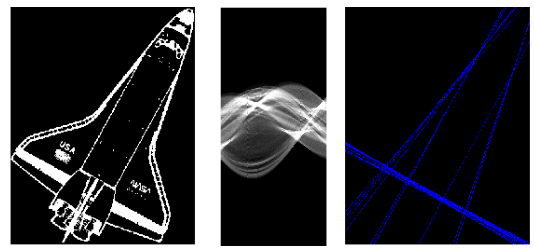


Image credit: NASA Dryden Research Aircraft Photo Archive

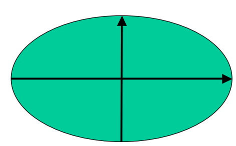
Algorithm

1. Quantize the parameter space
   int P[0, ρ_{max}][0, θ_{max}]; // accumulators

2. For each edge point (x, y) {
   For (θ = 0; θ <= θ_{max}; θ = θ+$\Delta\theta$) {
   $\rho = x\cos\theta + y\sin\theta$ // round off to integer
   (P[ρ][θ])++;
   }
   }

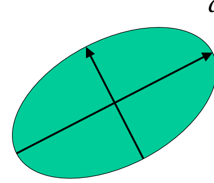
3. Find the peaks in P[ρ][θ].

Equations of Ellipse



$$\frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} = 1$$

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$



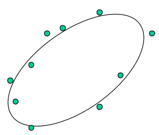
Let $\mathbf{x} = [x^2, xy, y^2, x, y, 1]^T$

$\mathbf{a} = [a, b, c, d, e, f]^T$

Then $\mathbf{x}^T\mathbf{a} = 0$

Ellipse Fitting: Problem Statement

Given a set of N image points $\mathbf{p}_i = [x_i, y_i]^T$
find the parameter vector $\mathbf{a}_0$ such that the ellipse



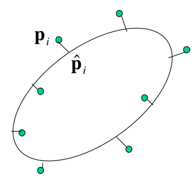
$$f(\mathbf{p}, \mathbf{a}) = \mathbf{x}^T \mathbf{a} = 0$$

fits $\mathbf{p}_i$ best in the least square sense:

$$\min_{\mathbf{a}} \sum_{i=1}^{N} [D(\mathbf{p}_i, \mathbf{a})]^2$$

Where $D(\mathbf{p}_i, \mathbf{a})$ is the distance from $\mathbf{p}_i$ to the ellipse.

Euclidean Distance Fit



$$D(\mathbf{p}_i, \mathbf{a}) = \|\hat{\mathbf{p}}_i - \mathbf{p}_i\|$$

$\hat{\mathbf{p}}_i$ is the point on the ellipse that is nearest to $\mathbf{p}_i$

$$f(\hat{\mathbf{p}}_i, \mathbf{a}) = 0$$

$\hat{\mathbf{p}}_i - \mathbf{p}_i$ is normal to the ellipse at $\hat{\mathbf{p}}_i$

Compute Distance Function

Computing the distance function is a constrained optimization problem:

$$\min_{\hat{\mathbf{p}}_i} \|\hat{\mathbf{p}}_i - \mathbf{p}_i\|^2 \quad \text{subject to} \quad f(\hat{\mathbf{p}}_i, \mathbf{a}) = 0$$

Using Lagrange multiplier, define:

$$L(x, y, \lambda) = \|\hat{\mathbf{p}}_i - \mathbf{p}_i\|^2 - 2\lambda f(\hat{\mathbf{p}}_i, \mathbf{a})$$

where $\hat{\mathbf{p}}_i = [x, y]^T$

Then the problem becomes: $\min_{\hat{\mathbf{p}}_i} L(x, y, \lambda)$

Set $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} = 0$ we have $\hat{\mathbf{p}}_i - \mathbf{p}_i = \lambda \nabla f(\hat{\mathbf{p}}_i, \mathbf{a})$

Ellipse Fitting with Euclidean Distance

Given a set of N image points $\mathbf{p}_i = [x_i, y_i]^T$
find the parameter vector $\mathbf{a}_0$ such that

$$\min_{\mathbf{a}} \sum_{i=1}^{N} \frac{|f(\mathbf{p}_i, \mathbf{a})|^2}{\|\nabla f(\mathbf{p}_i, \mathbf{a})\|^2}$$

This problem can be solved by using a numerical nonlinear optimization system.