Computer Vision

Lecture Set 04
Image Features

Huei-Yung Lin

lin@ee.ccu.edu.tw

Robot Vision Lab
Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei 106, Taiwan

October 23, 2022

Announcements - 10/24/22

- Homework 2 will be given later, due in two weeks (11/7).
- Exam I is scheduled on 11/7, covers up to Lecture Set 04.

Image Features?

Image Features?

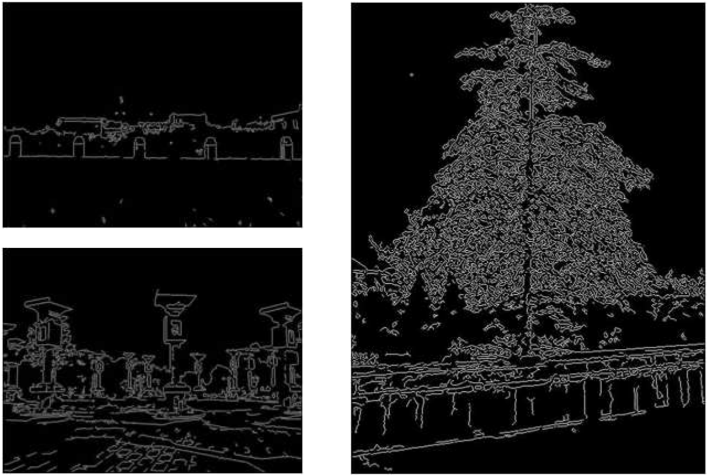


Image Features

- Image features in computer vision
  - Global feature – A global property of an image. e.g. the average grey value, the area in pixel, etc.
  - Local feature – A part of the image with some special properties. e.g. a circle, a line, a textured regions, etc.
- Image features (a practical definition): *local meaningful detectable parts of the image*
  - Points
  - Edges: step edges, line edges
  - Contours: closed contours are boundaries
  - Regions: similar color, similar feature, etc.

Image Features

- Image features in computer vision
  - Global feature – A global property of an image. e.g. the average grey value, the area in pixel, etc.
  - Local feature – A part of the image with some special properties. e.g. a circle, a line, a textured regions, etc.

- Image features (a practical definition): *local meaningful detectable parts of the image*
  - Points
  - Edges: step edges, line edges
  - Contours: closed contours are boundaries
  - Regions: similar color, similar feature, etc.

Remarks

- Most vision systems begin by detecting and locating some features in the input images
- In 3-D vision, feature extraction is an intermediate step, not the goal
  - We do not extract lines just to obtain line maps
  - We extract lines to navigate robots in corridors or for camera calibration
- Feature extraction is for certain purpose of the system
- "Perfect" feature extraction is not necessary (and also not possible?)

Remarks

- Most vision systems begin by detecting and locating some features in the input images
- In 3-D vision, feature extraction is an intermediate step, not the goal
  - We do not extract lines just to obtain line maps
  - We extract lines to navigate robots in corridors or for camera calibration
- Feature extraction is for certain purpose of the system
- "Perfect" feature extraction is not necessary (and also not possible?)

Remarks

- Most vision systems begin by detecting and locating some features in the input images
- In 3-D vision, feature extraction is an intermediate step, not the goal
  - We do not extract lines just to obtain line maps
  - We extract lines to navigate robots in corridors or for camera calibration
- Feature extraction is for certain purpose of the system
- "Perfect" feature extraction is not necessary (and also not possible?)

Edge Detection

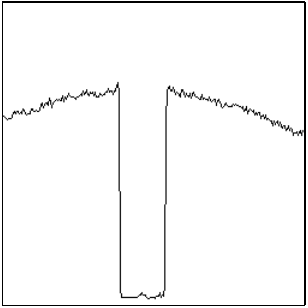
- Edges typically occur on the boundary between two different regions in an image
- Edge detection is frequently the first step in recovering information from images
- An edge is a significant local change in the intensity
  - Usually associated with a discontinuity in either the image intensity or its first derivative
  - The discontinuities can be step or line (ideally)
  - In reality, step becomes ramp and line becomes roof due to the smoothing of sharp edges in most images

Edge Detection

- Edges typically occur on the boundary between two different regions in an image
- Edge detection is frequently the first step in recovering information from images
- An edge is a significant local change in the intensity
  - Usually associated with a discontinuity in either the image intensity or its first derivative
  - The discontinuities can be step or line (ideally)
  - In reality, step becomes ramp and line becomes roof due to the smoothing of sharp edges in most images

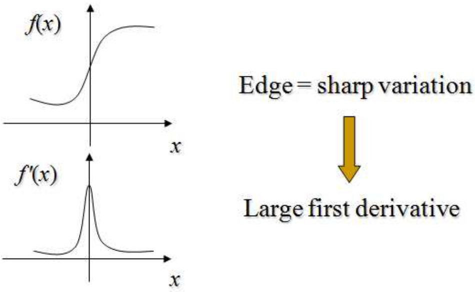
Edge Detection Scheme

- Edges happen at places where the image values exhibit sharp variation



Edge Detection Scheme

- The basic approach to edge detection is to compute “spatial derivatives” of the intensity image
- The act of taking spatial derivatives is usually approximated by convolution



Gradient

- The gradient is a measure of change in a function
- Significant changes in gray values can be detected by using a "discrete approximation" to the gradient
- The gradient is a 2-D equivalent of the first derivate and is defined as a vector

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \\ \partial f/\partial y \end{bmatrix}$$

  - The vector $\mathbf{G}[f(x,y)]$ points in the direction of the maximum rate of increase of the function $f(x,y)$
  - The magnitude of the gradient given by $|\mathbf{G}[f(x,y)]| = \sqrt{G_x^2 + G_y^2}$ equals the maximum rate of increase of $f(x,y)$ per unit distance

Gradient

- The gradient is a measure of change in a function
- Significant changes in gray values can be detected by using a "discrete approximation" to the gradient
- The gradient is a 2-D equivalent of the first derivate and is defined as a vector

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

  - The vector $\mathbf{G}[f(x,y)]$ points in the direction of the maximum rate of increase of the function $f(x,y)$
  - The magnitude of the gradient given by $|\mathbf{G}[f(x,y)]| = \sqrt{G_x^2 + G_y^2}$ equals the maximum rate of increase of $f(x,y)$ per unit distance

Approximation of Gradient

- The absolute values are commonly used to approximate the gradient magnitude

$$|\mathbf{G}[f(x,y)]| \approx |G_x| + |G_y| \quad \text{or} \quad |\mathbf{G}[f(x,y)]| \approx \max(|G_x|, |G_y|)$$

- The direction of gradient is define as

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_x}{G_y}\right)$$

- The magnitude of the gradient is “independent” of the direction of the edge
- Such operators are called isotropic operators

Digital Approximation

- For digital images, the gradient approximation can be

$$G_x \approx f[i, j+1] - f[i, j] \qquad G_y \approx f[i, j] - f[i+1, j]$$

Digital Approximation

- Simple convolution masks:
  - 2×1 and 1×2:

-1	1

-1
1

  - 2×2 and 2×2:

-1	1
-1	1

1	1
-1	-1

  - 3×1

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$\frac{df(x)}{dx} \approx \frac{f(x+1) - f(x-1)}{2}$$

Convolve with:

-1	0	1

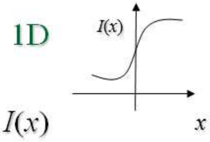
Edge Detection

- 1D Case
- $I(x)$
- $\dfrac{dI(x)}{dx}$
- $\left|\dfrac{dI(x)}{dx}\right| >$ threshold
- No orientation

- 2D
- $I(x,y)$
- $\nabla I(x,y) = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} I_x(x,y) \\ I_y(x,y) \end{pmatrix}$
- $|\nabla I(x,y)| = \sqrt{I_(x,y)^2 + I_y(x,y)^2} >$ threshold
- $\tan\theta = \dfrac{I_x(x,y)}{I_y(x,y)}$

Edge Detection

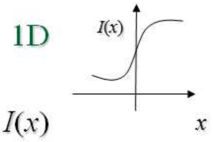
- 1D Case
- $I(x)$
- $\dfrac{dI(x)}{dx}$
- $\left|\dfrac{dI(x)}{dx}\right| > \text{threshold}$
- No orientation



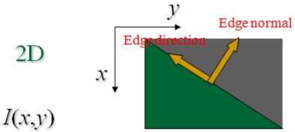
- 2D
- $I(x, y)$
- $\nabla I(x, y) = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} I_x(x, y) \\ I_y(x, y) \end{pmatrix}$
- $|\nabla I(x, y)| = \sqrt{I_x(x, y)^2 + I_y(x, y)^2} > \text{threshold}$
- $\tan\theta = \dfrac{I_x(x, y)}{I_y(x, y)}$

Edge Detection

- 1D Case
- $I(x)$
- $\dfrac{dI(x)}{dx}$
- $\left|\dfrac{dI(x)}{dx}\right| > \text{threshold}$
- No orientation



- 2D
- $I(x, y)$
- $\nabla I(x,y) = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} I_x(x,y) \\ I_y(x,y) \end{pmatrix}$
- $|\nabla I(x,y)| = \sqrt{I_(x,y)^2 + I_y(x,y)^2} >$ threshold
- $\tan\theta = \dfrac{I_x(x,y)}{I_y(x,y)}$



Edge Detection

- Vertical edges:
  - Convolve with

-1	0	1

- Horizontal edges:
  - Convolve with

-1
0
1

The Essential Edge Descriptor

- Edge position or center
  - The image position at which the edge is located
  - Usually saved in a binary image (1 : edge, 0 : no edge)
- Edge normal
  - The direction (unit vector) of the maximum intensity variation at the edge point
  - This identifies the direction perpendicular to the edge
- Edge direction
  - The direction perpendicular to the edge normal
  - This identifies the direction tangent to the edge
- Edge strength
  - A measure of the local image contrast; i.e., how marked the intensity variation is across the edge along the normal

The Essential Edge Descriptor

- Edge position or center
  - The image position at which the edge is located
  - Usually saved in a binary image (1 : edge, 0 : no edge)
- Edge normal
  - The direction (unit vector) of the maximum intensity variation at the edge point
  - This identifies the direction perpendicular to the edge
- Edge direction
  - The direction perpendicular to the edge normal
  - This identifies the direction tangent to the edge
- Edge strength
  - A measure of the local image contrast; i.e., how marked the intensity variation is across the edge along the normal

The Essential Edge Descriptor

- Edge position or center
  - The image position at which the edge is located
  - Usually saved in a binary image (1 : edge, 0 : no edge)
- Edge normal
  - The direction (unit vector) of the maximum intensity variation at the edge point
  - This identifies the direction perpendicular to the edge
- Edge direction
  - The direction perpendicular to the edge normal
  - This identifies the direction tangent to the edge
- Edge strength
  - A measure of the local image contrast; i.e., how marked the intensity variation is across the edge along the normal

The Essential Edge Descriptor

- Edge position or center
  - The image position at which the edge is located
  - Usually saved in a binary image (1 : edge, 0 : no edge)
- Edge normal
  - The direction (unit vector) of the maximum intensity variation at the edge point
  - This identifies the direction perpendicular to the edge
- Edge direction
  - The direction perpendicular to the edge normal
  - This identifies the direction tangent to the edge
- Edge strength
  - A measure of the local image contrast; i.e., how marked the intensity variation is across the edge along the normal

Edge Detection Steps

- Noise smoothing (filtering)
  - Suppress noise without destroying the true edges
- Edge enhancement
  - Design a filter responding to edges
  - Usually performed by computing gradient magnitude
- Edge localization
  - Thinning (non-maximum suppression)
  - Thresholding (used to decide whether the output is an edge point or not)

- Some Assumptions
  - The edge enhancement filter is linear
  - The filter must be optimal for noisy step edge
  - The image noise is additive, white and Gaussian

Edge Detection Steps

- Noise smoothing (filtering)
  - Suppress noise without destroying the true edges
- Edge enhancement
  - Design a filter responding to edges
  - Usually performed by computing gradient magnitude
- Edge localization
  - Thinning (non-maximum suppression)
  - Thresholding (used to decide whether the output is an edge point or not)

- Some Assumptions
  - The edge enhancement filter is linear
  - The filter must be optimal for noisy step edge
  - The image noise is additive, white and Gaussian

Edge Detection Steps

- Noise smoothing (filtering)
  - Suppress noise without destroying the true edges
- Edge enhancement
  - Design a filter responding to edges
  - Usually performed by computing gradient magnitude
- Edge localization
  - Thinning (non-maximum suppression)
  - Thresholding (used to decide whether the output is an edge point or not)

- Some Assumptions
  - The edge enhancement filter is linear
  - The filter must be optimal for noisy step edge
  - The image noise is additive, white and Gaussian

Edge Detection Steps

- Noise smoothing (filtering)
  - Suppress noise without destroying the true edges
- Edge enhancement
  - Design a filter responding to edges
  - Usually performed by computing gradient magnitude
- Edge localization
  - Thinning (non-maximum suppression)
  - Thresholding (used to decide whether the output is an edge point or not)

- Some Assumptions
  - The edge enhancement filter is linear
  - The filter must be optimal for noisy step edge
  - The image noise is additive, white and Gaussian

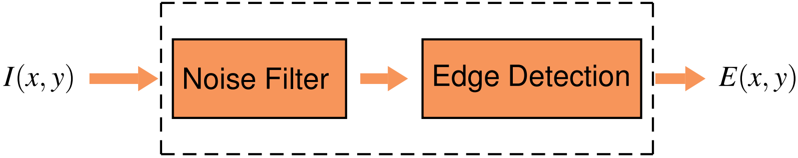
Edge Detection Steps

- Noise smoothing (filtering)
  - Suppress noise without destroying the true edges
- Edge enhancement
  - Design a filter responding to edges
  - Usually performed by computing gradient magnitude
- Edge localization
  - Thinning (non-maximum suppression)
  - Thresholding (used to decide whether the output is an edge point or not)

- Some Assumptions
  - The edge enhancement filter is linear
  - The filter must be optimal for noisy step edge
  - The image noise is additive, white and Gaussian

Noise Suppression

- The differential kernels act as “high pass filters” which tend to amplify noise
- This is why edge detection is usually preceded by a noise reduction or filtering operation



Combined Linear Filter

Noise Smoothing & Edge Detection

- Prewitt Edge Detector (vertical)
  - Convolve with:

-1	0	1
-1	0	1
-1	0	1

Noise Smoothing

Vertical Edge Detection

- Prewitt Edge Detector (horizontal)
  - Convolve with:

-1	-1	-1
0	0	0
1	1	1

Horizontal Edge Detection

Noise Smoothing

Roberts Detector

- G_x :

+1	0
0	-1

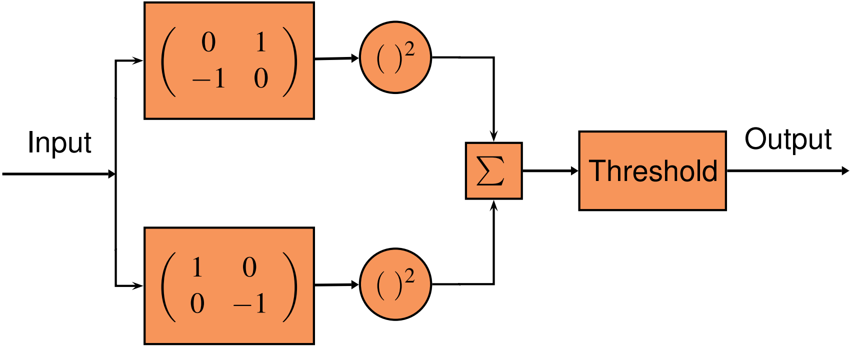
G_y :

0	+1
-1	0

- $|G| = \sqrt{G_x^2 + G_y^2}$

Roberts Detector

- The Roberts detector gives an approximation to the continuous gradient at $[i + 1/2, j + 1/2]$. (not at $[i, j]$, why?)



Sobel Detector

- Sobel detector gives more weight to the 4-neighbors
- Emphasize the pixels closer to the center of the mask (compare with Prewitt!)
- G_x :

-1	0	1
-2	0	2
-1	0	1

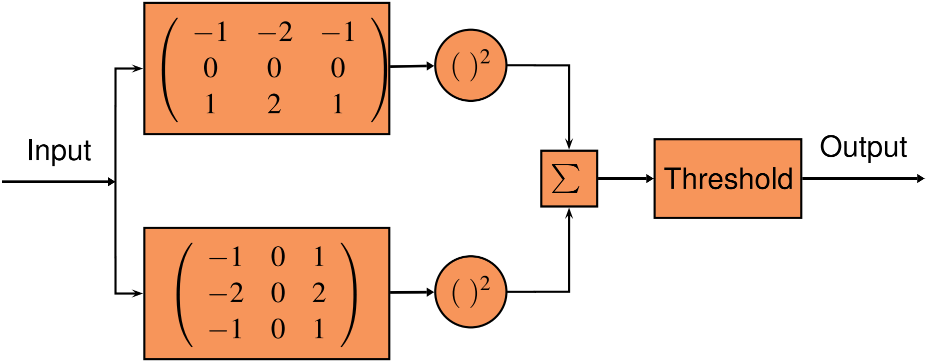
G_y :

-1	-2	-1
0	0	0
1	2	1

- $\theta = \tan^{-1} \frac{G_y}{G_x}$

Sobel Detector

- Sobel operator is one of the most commonly used edge detector (See handout for how it is derived)



Examples

Original

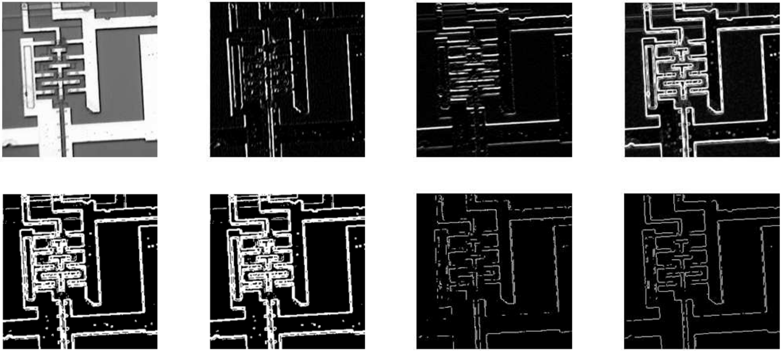
Sobel

Roberts

Canny

Prewitt

Examples



The Canny Principle

- Canny derived the form of the “optimal” linear filter for detecting edges in 1-D
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- The edge was modeled as a simple step corrupted by additive Gaussian noise
- Experiments consistently show that it performs very well
- Probably the most used by computer vision practitioners

The Canny Principle

- Canny derived the form of the “optimal” linear filter for detecting edges in 1-D
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- The edge was modeled as a simple step corrupted by additive Gaussian noise
- Experiments consistently show that it performs very well
- Probably the most used by computer vision practitioners

Canny's Criteria

- Detection – the important edges should not be missed and there should be no spurious responses
- Localization – the distance between the actual and located position of the edge should be minimal
- Single response – minimize multiple responses to a single edge
  - This is partly covered by the first criterion (since if there are two responses to a single edge, one of them should be considered as false)
  - This criterion solves the problem of an edge corrupted by noise

- Canny Edge Detector
  - Uses a mathematical model of the edge and the noise
  - Formalizes a performance criteria
  - Synthesizes the best filter

Canny's Criteria

- Detection – the important edges should not be missed and there should be no spurious responses
- Localization – the distance between the actual and located position of the edge should be minimal
- Single response – minimize multiple responses to a single edge
  - This is partly covered by the first criterion (since if there are two responses to a single edge, one of them should be considered as false)
  - This criterion solves the problem of an edge corrupted by noise
- Canny Edge Detector
  - Uses a mathematical model of the edge and the noise
  - Formalizes a performance criteria
  - Synthesizes the best filter

Canny's Criteria

- Detection – the important edges should not be missed and there should be no spurious responses
- Localization – the distance between the actual and located position of the edge should be minimal
- Single response – minimize multiple responses to a single edge
  - This is partly covered by the first criterion (since if there are two responses to a single edge, one of them should be considered as false)
  - This criterion solves the problem of an edge corrupted by noise

- Canny Edge Detector
  - Uses a mathematical model of the edge and the noise
  - Formalizes a performance criteria
  - Synthesizes the best filter

Canny's Criteria

- Detection – the important edges should not be missed and there should be no spurious responses
- Localization – the distance between the actual and located position of the edge should be minimal
- Single response – minimize multiple responses to a single edge
  - This is partly covered by the first criterion (since if there are two responses to a single edge, one of them should be considered as false)
  - This criterion solves the problem of an edge corrupted by noise

- Canny Edge Detector
  - Uses a mathematical model of the edge and the noise
  - Formalizes a performance criteria
  - Synthesizes the best filter

Canny's Criteria

- Detection – the important edges should not be missed and there should be no spurious responses
- Localization – the distance between the actual and located position of the edge should be minimal
- Single response – minimize multiple responses to a single edge
  - This is partly covered by the first criterion (since if there are two responses to a single edge, one of them should be considered as false)
  - This criterion solves the problem of an edge corrupted by noise

- Canny Edge Detector
  - Uses a mathematical model of the edge and the noise
  - Formalizes a performance criteria
  - Synthesizes the best filter

Performance Criteria

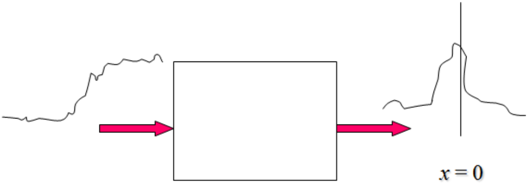
- Good detection
  - The filter must have a stronger response at the edge location ($x = 0$) than to noise
- Good localization
  - The filter response must be maximum very close to $x = 0$
- Low false positives
  - There should be only one maximum in a reasonable neighborhood of $x = 0$

Performance Criteria

- Good detection
  - The filter must have a stronger response at the edge location ($x = 0$) than to noise
- Good localization
  - The filter response must be maximum very close to $x = 0$
- Low false positives
  - There should be only one maximum in a reasonable neighborhood of $x = 0$

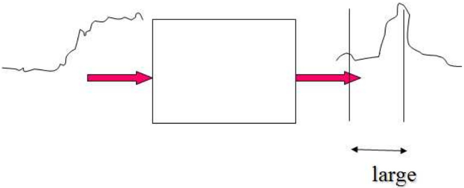
Performance Criteria

- Good detection
  - The filter must have a stronger response at the edge location ($x = 0$) than to noise
- Good localization
  - The filter response must be maximum very close to $x = 0$
- Low false positives
  - There should be only one maximum in a reasonable neighborhood of $x = 0$



Performance Criteria

- Good detection
  - The filter must have a stronger response at the edge location ($x = 0$) than to noise
- Good localization
  - The filter response must be maximum very close to $x = 0$
- Low false positives
  - There should be only one maximum in a reasonable neighborhood of $x = 0$



Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria
  - There is no close-form solution for the optimal filter
  - However, it looks "very similar" to the first derivative of a Gaussian (DoG)
- Strictly speaking, Canny's filter was derived in the context of 1-D profiles
  - To extend them to 2-D images we can run the filter at several different orientations in the image and detect edge elements at all orientations
- Canny Approximation
  - Smooth image with Gaussian
  - Compute the derivatives in x and y directions
  - Perform non-maximal suppression and sub-pixel interpolation using edge strength and direction values
  - Perform edge linking / hysteresis thresholding

Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria
  - There is no close-form solution for the optimal filter
  - However, it looks "very similar" to the first derivative of a Gaussian (DoG)
- Strictly speaking, Canny's filter was derived in the context of 1-D profiles
  - To extend them to 2-D images we can run the filter at several different orientations in the image and detect edge elements at all orientations
- Canny Approximation
  - Smooth image with Gaussian
  - Compute the derivatives in x and y directions
  - Perform non-maximal suppression and sub-pixel interpolation using edge strength and direction values
  - Perform edge linking / hysteresis thresholding

Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria
  - There is no close-form solution for the optimal filter
  - However, it looks "very similar" to the first derivative of a Gaussian (DoG)
- Strictly speaking, Canny's filter was derived in the context of 1-D profiles
  - To extend them to 2-D images we can run the filter at several different orientations in the image and detect edge elements at all orientations
- Canny Approximation
  - Smooth image with Gaussian
  - Compute the derivatives in x and y directions
  - Perform non-maximal suppression and sub-pixel interpolation using edge strength and direction values
  - Perform edge linking / hysteresis thresholding

- 1-D Gaussian with zero mean is given by

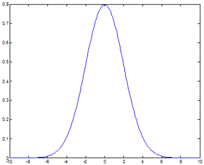
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

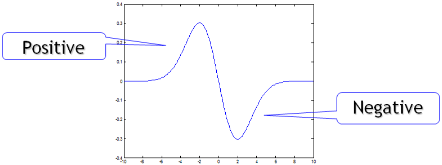
- Ignore the scale factor,

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$

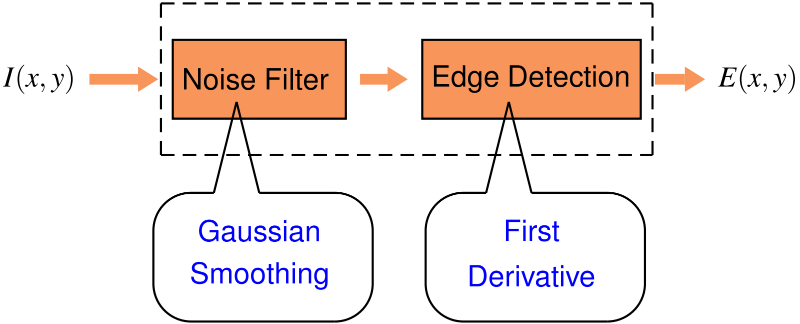
- The first derivative of Gaussian is given by

$$g'(x) = -\frac{1}{2\sigma^2} 2x e^{-\frac{x^2}{2\sigma^2}} = -\frac{x^2}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$





Another Interpretation



$$E(x) = \frac{d(I(x) * G(x))}{dx} = I(x) * \frac{dG(x)}{dx}$$

Canny Edge Detector

- 1-D
- First derivative
- $E(x) = \dfrac{d(I(x) * G(x))}{dx}$
- Absolute value
- $|E(x)| \geq Th$

- 2-D
- Gradient vector
- $E(x,y) = \nabla(I(x,y) * G(x,y))$
- Magnitude
- $|E(x)| \geq Th$

CANNY_ENHANCER

- The input is image I; G is a zero mean Gaussian filter with standard derivation σ
  - $J = I * G$ (smoothing)
  - For each pixel (i,j): (edge enhancement)
    - Compute the image gradient: $\nabla J(i,j) = (J_x(i,j), J_y(i,j))$
    - Estimate edge strength: $e_s(i,j) = \sqrt{J_x^2(i,j) + J_y^2(i,j)}$
    - Estimate edge orientation: $e_o = \tan^{-1} \frac{J_x(i,j)}{J_y(i,j)}$
- The output are images E_s and E_o (edge strength and edge orientation)

- The output image E_s has the magnitudes of the smoothed gradient
- σ determines the amount of smoothing
- E_s has large values at edges

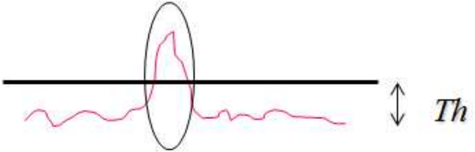
CANNY_ENHANCER

- The input is image I; G is a zero mean Gaussian filter with standard derivation σ
  - $J = I * G$ (smoothing)
  - For each pixel (i,j): (edge enhancement)
    - Compute the image gradient: $\nabla J(i,j) = (J_x(i,j), J_y(i,j))$
    - Estimate edge strength: $e_s(i,j) = \sqrt{J_x^2(i,j) + J_y^2(i,j)}$
    - Estimate edge orientation: $e_o = \tan^{-1} \frac{J_x(i,j)}{J_y(i,j)}$
- The output are images E_s and E_o (edge strength and edge orientation)

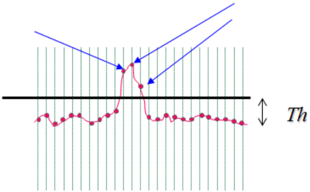
- The output image E_s has the magnitudes of the smoothed gradient
- σ determines the amount of smoothing
- E_s has large values at edges

Edge Detection

- E_s has large values at edges:
  - Find local maxima



  - But it also may have wide ridges around the local maxima (large value around the edges)



Non-Maximal Suppression

- One approach to overcoming the problem of edge thickening is to explicitly look for maxima of the response to the edge enhancement filter

- Sub-pixel Localization
  - One can try to further localize the position of the edge within a pixel by analyzing the response to the edge enhancement filter
  - One common approach is to fit a "quadratic polynomial" to the filter response in the region of a maxima and compute the true maximum
  - Let $y(x) = ax^2 + bx + c$ and perform interpolation to find max or min

Non-Maximal Suppression

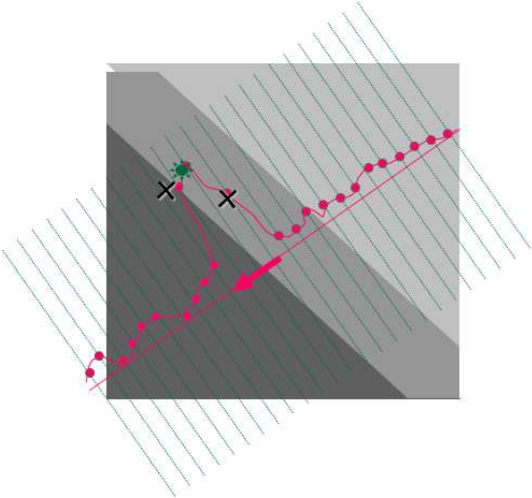
- One approach to overcoming the problem of edge thickening is to explicitly look for maxima of the response to the edge enhancement filter

- Sub-pixel Localization
  - One can try to further localize the position of the edge within a pixel by analyzing the response to the edge enhancement filter
  - One common approach is to fit a "quadratic polynomial" to the filter response in the region of a maxima and compute the true maximum
  - Let $y(x) = ax^2 + bx + c$ and perform interpolation to find max or min

NONMAX_SUPPRESSION

- The inputs are E_s and E_o (outputs of CANNY_ENHANCER)
- Consider 4 directions $D = \{0, 45, 90, 135 \quad \text{degrees}\}$ with respect to horizontal axis image reference frame
- For each pixel (i,j) do:
  - Find the direction $d \in D$ such that $d \approx E_o(i,j)$ (normal to the edge)
  - If $E_s(i,j)$ smaller than at least one of its neighbors along d
    - $I_N(i,j) = 0$ (suppression)
    - Otherwise, $I_N(i,j) = E_s(i,j)$
- The output is the thinned edge image I_N

Graphical Interpretation



Edge Linking

- The output of the edge enhancement stage is a binary array indicating the locations of edgels (edge elements) in the image
- The edge linking stage attempts to group these discrete elements into chains much like stringing pearls
- Problems in edge linking:
  - Edges can be broken because of low contrast
  - Junctions can cause major problems since the edge enhancement procedure tends to fail in these situations and the linker can become confused

Thresholding

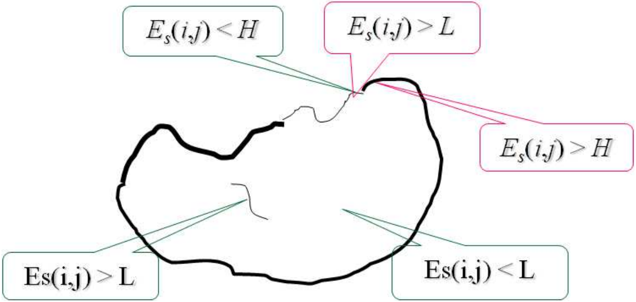
- Edges are found by thresholding the output of NONMAX_SUPRESSION
- If the threshold is too high: Very few (none) edges
  - High misdetections
  - Many gaps
- If the threshold is too low: Too many (all pixels) edges
  - High false positives
  - Many extra edges

Hysteresis Thresholding

- Canny proposed an approach to dealing with broken edge chains in the linking phase
- The idea is to maintain two thresholds on edge strength one for starting a chain and a lower one for use during linking
- In this way the linker will work well even when a chain has some low contrast sections

Solution

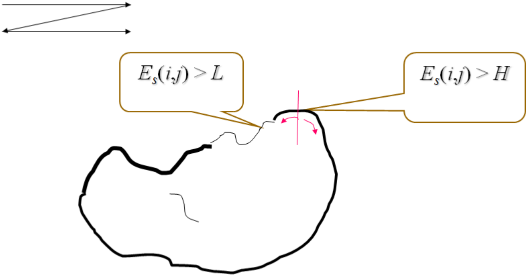
- Hysteresis Thresholding
- "Strong edges" reinforce adjacent "weak edges"



HYSTERESIS_THRESH

- Inputs:
  - I_N (output of NONMAX_SUPRESSION)
  - E_o (output of CANNY_ENHANCER)
  - Thresholds L and H
- Scanning all edge points in I_N in a fixed order:
  - Locate the next unvisited pixel such that $I_N(i,j) > H$
  - Starting from $I_N(i,j)$, follow the chains of connected local maxima, in both directions perpendicular to the edge normal, as long as $I_N > L$; Mark all visited points, and save the location of the contour points
- Output:
  - A set of lists describing the contours

Hysteresis Thresholding



Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the "feature synthesis" approach

Algorithm: Canny Edge Detector

- Convolve an image f with a Gaussian of scale σ
- Estimate local edge normal directions n for each pixel in the image
- Find the location of the edges (non-maximal suppression)
- Compute the magnitude of the edge
- Threshold edges in the image with hysteresis to eliminate spurious responses
- Repeat steps 1. through 5. for ascending values of the standard deviation σ
- Aggregate the final information about edges at multiple scale using the “feature synthesis” approach

Example

Threshold = 0.1, σ = 1

Threshold = 0.1, σ = 2.8

Threshold = 0.3, σ = 1

Threshold = 0.3, σ = 2.8

Disclaimer

- No edge detection scheme is going to work perfectly in all cases
- This is due to the fact that our notion of what constitutes a salient edge in the image is actually somewhat subtle

Detecting Corners

- We are often interested in detecting point features in an image
- These features are usually defined as regions in the image where there is significant edge strength in two or more directions
- They can be used for
  - Object tracking
  - 3D triangulation (stereo)
  - Object recognition
- Need two strong edges
- If E_x and E_y denote the gradients of the image in the x and y directions, then the behavior of the gradients in a region around a point can be obtained by considering the following matrix

$$C = \sum \begin{pmatrix} E_x \\ E_y \end{pmatrix} \begin{pmatrix} E_x & E_y \end{pmatrix} = \sum \begin{pmatrix} E_x^2 & E_x E_y \\ E_x E_y & E_y^2 \end{pmatrix}$$

Detecting Corners

- We are often interested in detecting point features in an image
- These features are usually defined as regions in the image where there is significant edge strength in two or more directions
- They can be used for
  - Object tracking
  - 3D triangulation (stereo)
  - Object recognition
- Need two strong edges
- If E_x and E_y denote the gradients of the image in the x and y directions, then the behavior of the gradients in a region around a point can be obtained by considering the following matrix

$$C = \sum \begin{pmatrix} E_x \\ E_y \end{pmatrix} \begin{pmatrix} E_x & E_y \end{pmatrix} = \sum \begin{pmatrix} E_x^2 & E_x E_y \\ E_x E_y & E_y^2 \end{pmatrix}$$

Examining The Matrix

- One way to decide on the presence of a corner is to look at the eigenvalues of the 2×2 matrix C
  - If the area is a region of “constant intensity” we would expect both eigenvalues to be small (or zero)
  - If it contains a edge we expect one large eigenvalue and one small one
  - If it contains edges at two or more orientations we expect two large eigenvalues
- If $\min(\lambda_1, \lambda_2) > T$, then there is a corner!

Finding Corner

- One approach to finding corners is to find locations where the smaller eigenvalue is greater than some threshold
- We could also consider the ratio of the two eigenvalues
- Issues:
  - Localization – It can be difficult to precisely localize the corner in the intensity image
  - Modeling – It can be helpful to have a model of the corners you are trying to find in order to detect and localize them more systematically

Finding Corner

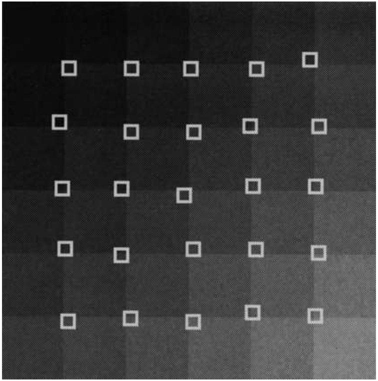
- One approach to finding corners is to find locations where the smaller eigenvalue is greater than some threshold
- We could also consider the ratio of the two eigenvalues
- Issues:
  - Localization – It can be difficult to precisely localize the corner in the intensity image
  - Modeling – It can be helpful to have a model of the corners you are trying to find in order to detect and localize them more systematically

Corner Detection

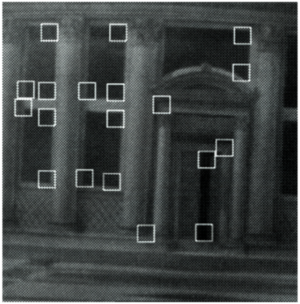
- Compute image gradient
- For each $m \times m$ neighborhood, compute matrix C
- If smaller eigenvalue λ_2 is greater than threshold τ, record a corner
- Non-maximum suppression: only keep strongest corner in each $m \times m$ window

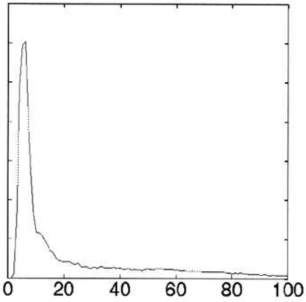
Corner Detection Results

- Checkerboard with noise



Corner Detection Results





Histogram of λ_2 (smaller eigenvalues)

Detecting Lines

- The difference between line detection and edge detection:
  - Edges = local
  - Lines = non-local
- Line detection usually performed on the output of an edge detector
- Several different approaches:
  - For each possible line, check whether the line is present: “brute force”
  - Given detected edges, record lines to which they might belong: “Hough transform + voting”
  - Given guess for approximate location of a line, refine that guess: “fitting”
- Second method (Hough transform) is efficient for finding unknown lines, but not always accurate

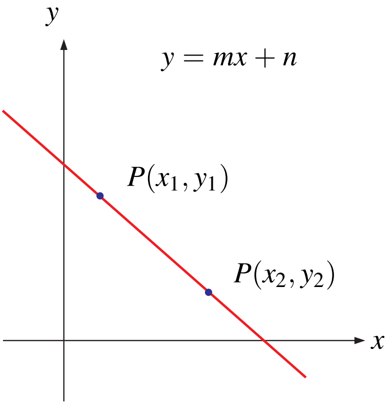
Detecting Lines

- The difference between line detection and edge detection:
  - Edges = local
  - Lines = non-local
- Line detection usually performed on the output of an edge detector
- Several different approaches:
  - For each possible line, check whether the line is present: "brute force"
  - Given detected edges, record lines to which they might belong: "Hough transform + voting"
  - Given guess for approximate location of a line, refine that guess: "fitting"
- Second method (Hough transform) is efficient for finding unknown lines, but not always accurate

Detecting Lines

- The difference between line detection and edge detection:
  - Edges = local
  - Lines = non-local
- Line detection usually performed on the output of an edge detector
- Several different approaches:
  - For each possible line, check whether the line is present: “brute force”
  - Given detected edges, record lines to which they might belong: “Hough transform + voting”
  - Given guess for approximate location of a line, refine that guess: “fitting”
- Second method (Hough transform) is efficient for finding unknown lines, but not always accurate

Mathematical model of a line:



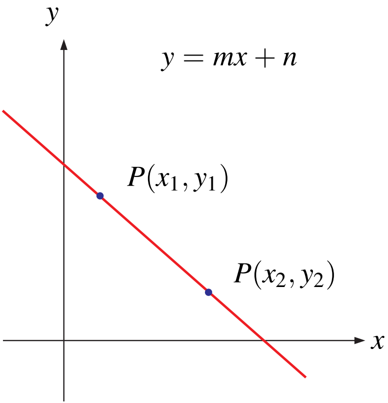
$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$\vdots$$

$$y_N = mx_N + n$$

Mathematical model of a line:



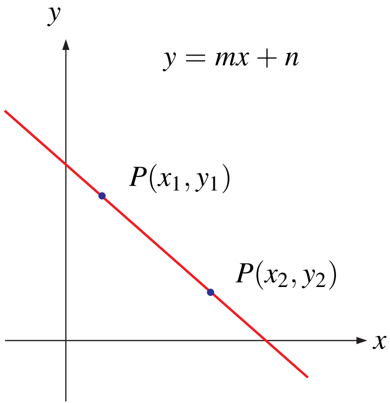
$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$\vdots$$

$$y_N = mx_N + n$$

Mathematical model of a line:



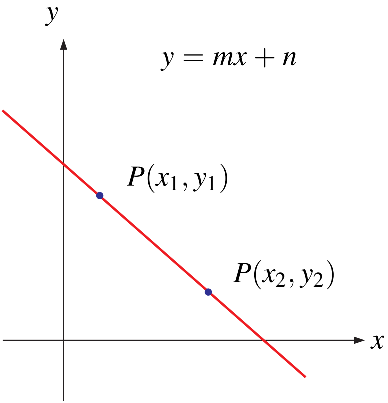
$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$\vdots$$

$$y_N = mx_N + n$$

Mathematical model of a line:



$$y_1 = mx_1 + n$$

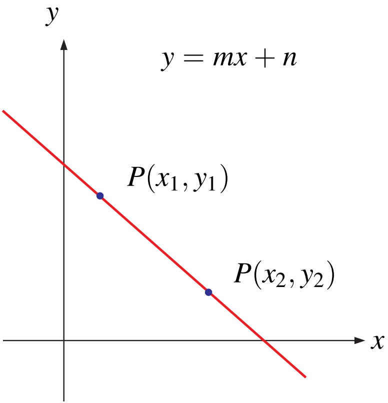
$$y_2 = mx_2 + n$$

$$\vdots$$

$$y_N = mx_N + n$$

Line Detection

Mathematical model of a line:



$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$\vdots$$

$$y_N = mx_N + n$$

Image and Parameter Spaces

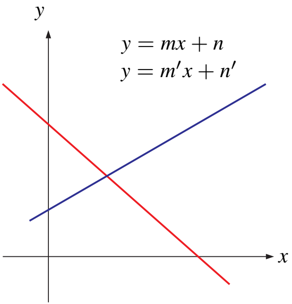
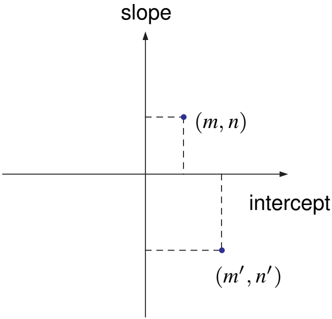


Image Space

$$y_1 = mx_1 + n$$
$$y_2 = mx_2 + n$$
$$\vdots$$
$$y_N = mx_N + n$$

$$y_1 = m'x_1 + n'$$
$$y_2 = m'x_2 + n'$$
$$\vdots$$
$$y_N = m'x_N + n'$$



Parameter Space

Line in Image Space $\sim$ Point in Parameter Space

Image and Parameter Spaces

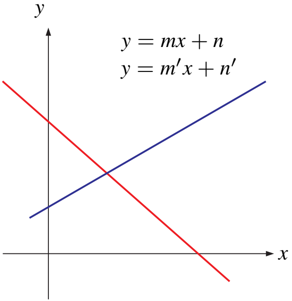
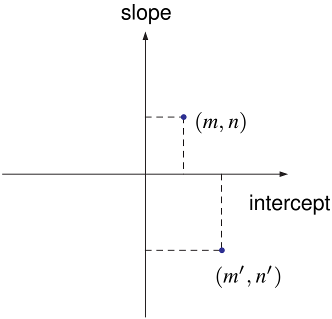


Image Space

$$y_1 = mx_1 + n$$
$$y_2 = mx_2 + n$$
$$\vdots$$
$$y_N = mx_N + n$$

$$y_1 = m'x_1 + n'$$
$$y_2 = m'x_2 + n'$$
$$\vdots$$
$$y_N = m'x_N + n'$$



Parameter Space

Line in Image Space $\sim$ Point in Parameter Space

Image and Parameter Spaces

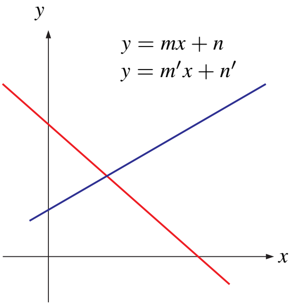
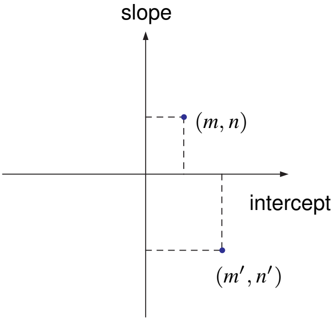


Image Space

$$
\begin{aligned}
y_1 &= mx_1 + n \\
y_2 &= mx_2 + n \\
&\vdots \\
y_N &= mx_N + n
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= m'x_1 + n' \\
y_2 &= m'x_2 + n' \\
&\vdots \\
y_N &= m'x_N + n'
\end{aligned}
$$



Parameter Space

Line in Image Space $\sim$ Point in Parameter Space

Image and Parameter Spaces

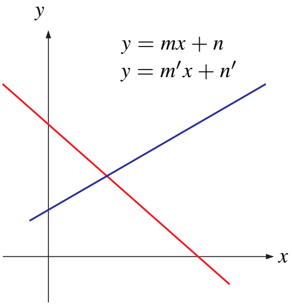
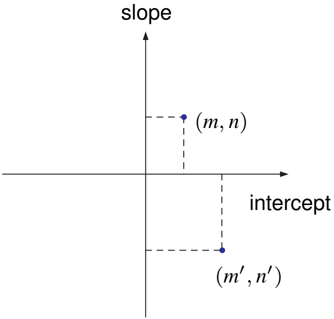


Image Space

$$
\begin{aligned}
y_1 &= mx_1 + n \\
y_2 &= mx_2 + n \\
&\vdots \\
y_N &= mx_N + n
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= m'x_1 + n' \\
y_2 &= m'x_2 + n' \\
&\vdots \\
y_N &= m'x_N + n'
\end{aligned}
$$



Parameter Space

Line in Image Space $\sim$ Point in Parameter Space

Image and Parameter Spaces

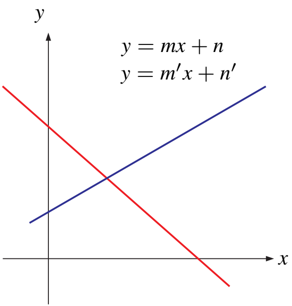
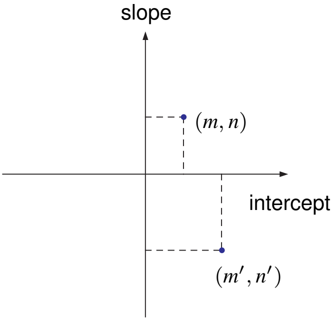


Image Space

$$
\begin{aligned}
y_1 &= mx_1 + n \\
y_2 &= mx_2 + n \\
&\vdots \\
y_N &= mx_N + n
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= m'x_1 + n' \\
y_2 &= m'x_2 + n' \\
&\vdots \\
y_N &= m'x_N + n'
\end{aligned}
$$



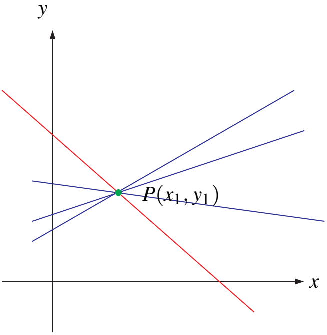
Parameter Space

Line in Image Space $\sim$ Point in Parameter Space

Image Space

Fix (m, n), vary (x, y) – Line $\quad y = mx + n$

Fix (x_1, y_1), vary (m, n) – Lines through a Point $\quad y_1 = mx_1 + n$



Parameter Space

The line $y_1 = mx_1 + n$ can be re-written as: $n = -x_1 m + y_1$

Fix $(-x_1, y_1)$, vary (m, n) – Line $n = -x_1 m + y_1$

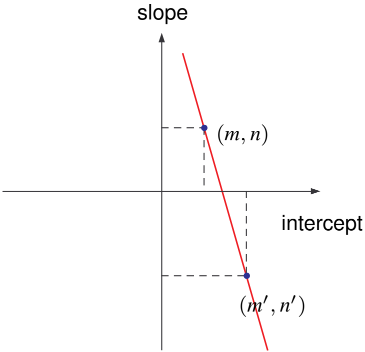


Image & Parameter Spaces

Image Space

- Lines
- Points
- Collinear points

Parameter Space

- Points
- Lines
- Intersecting lines

This is called duality!

Hough Transform

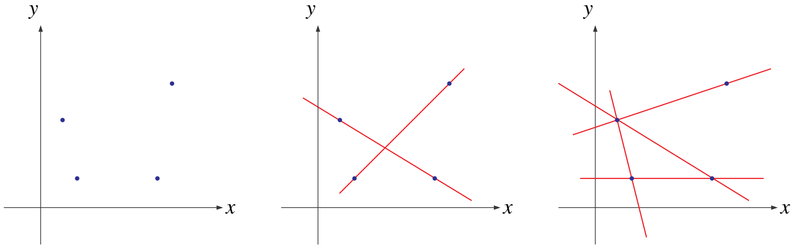
- General idea: transform from image coordinates to parameter space of features
  - Map a difficult pattern problem into a simple peak detection problem
  - Need parameterized model of features
  - For each pixel, determine all parameter values that might have given rise to that pixel; vote
  - At end, look for peaks in parameter space
- This approach is a voting scheme based on accumulating evidence in a parameter space

Hough Transform

- General idea: transform from image coordinates to parameter space of features
  - Map a difficult pattern problem into a simple peak detection problem
  - Need parameterized model of features
  - For each pixel, determine all parameter values that might have given rise to that pixel; vote
  - At end, look for peaks in parameter space
- This approach is a voting scheme based on accumulating evidence in a parameter space

Hough Transform for Lines

- Each input measurement indicates its contribution to a globally consistent solution
- Here this problem is under constrained
  - Generic line: $y = ax + b$
  - Parameters: a and b



Hough Transform for Lines

- Given an edge point, there is an infinite number of lines passing through it (vary m and n)
- These lines can be represented as a line in parameter space

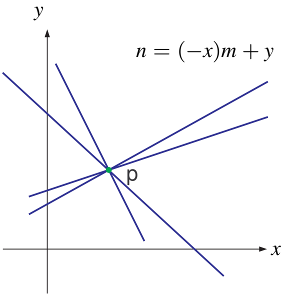
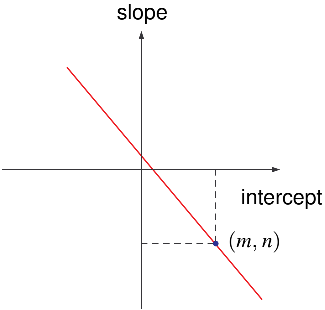


Image Space



Parameter Space

Hough Transform for Lines

- Given a set of collinear edge points, each of them have associated a line in parameter spaces
- These lines intersect at the point (m, n) corresponding to the parameters of the line in the image space

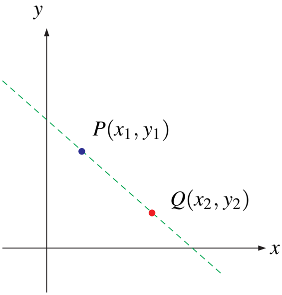
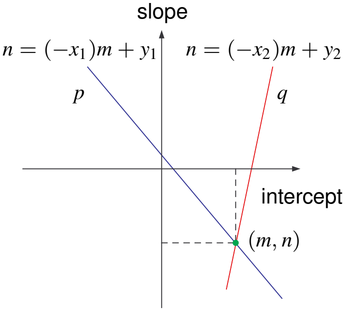


Image Space



Parameter Space

Hough Transform Technique

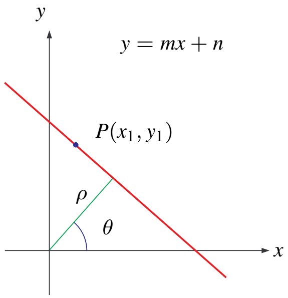
- At each point of the (discrete) parameter space, count how many lines pass through it
  - Use an array of counters
  - Can be thought as a “parameter image”
- The higher the count, the more edges are collinear in the image space
  - Find a peak in the counter array
  - This is a “bright” point in the parameter image
  - It can be found by thresholding
- Practical Issues
  - The slope of the line is $-\infty < m < \infty$
    - ★ The parameter space is infinite
  - The representation $y = mx + n$ does not express lines of the form $x = k$

Hough Transform Technique

- At each point of the (discrete) parameter space, count how many lines pass through it
  - Use an array of counters
  - Can be thought as a “parameter image”
- The higher the count, the more edges are collinear in the image space
  - Find a peak in the counter array
  - This is a “bright” point in the parameter image
  - It can be found by thresholding
- Practical Issues
  - The slope of the line is $-\infty < m < \infty$
    - The parameter space is infinite
  - The representation $y = mx + n$ does not express lines of the form $x = k$

Solution

- Use the “normal” equation of a line:



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

θ is the line orientation
ρ is the distance between the origin and the line

New Parameter Space

- Use the parameter space (ρ, θ)
- The new space is finite
  - $0 < \rho < D$, where D is the image diagonal
  - $0 < \theta < 2\pi$
- The new space can represent all lines
  - $y = k$ is represented with $\rho = k, \theta = 90°$
  - $x = k$ is represented with $\rho = k, \theta = 0°$
- A point in image space is now represented as a sinusoid
  - $\rho = x \cos\theta + y \sin\theta$

New Parameter Space

- Use the parameter space (ρ, θ)
- The new space is finite
  - $0 < \rho < D$, where D is the image diagonal
  - $0 < \theta < 2\pi$
- The new space can represent all lines
  - $y = k$ is represented with $\rho = k, \theta = 90°$
  - $x = k$ is represented with $\rho = k, \theta = 0°$
- A point in image space is now represented as a sinusoid
  - $\rho = x \cos\theta + y \sin\theta$

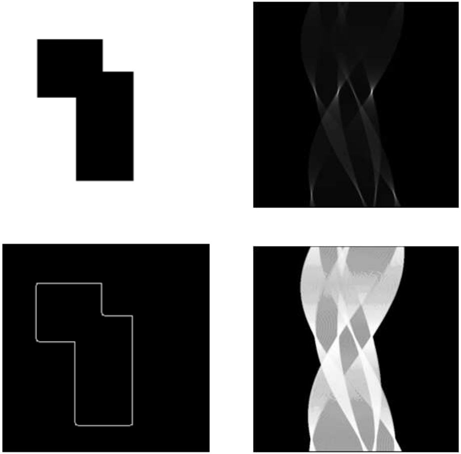
Hough Transform Algorithm

- Input is an edge image ($E(i,j) = 1$ for edgels)
  - Discretize θ and ρ in increments of θ_d and ρ_d
  - Let $A(R,T)$ be an array of integer accumulators, initialized to 0
  - For each pixel $E(i,j) = 1$ and $h = 1, 2, \ldots, T$ do
    - $\rho = i\cos(h\theta_d) + j\sin(h\theta_d)$
    - Find closest integer k of the element of ρ_d, corresponding to ρ
    - Increment counter $A(h,k)$ by one
  - Find all local maxima in $A(R,T) > \mathit{threshold}$
- Output is a set of pairs (ρ_d, θ_d) describing the lines detected in E in polar form

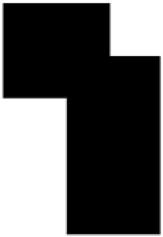
Hough Transform Speed Up

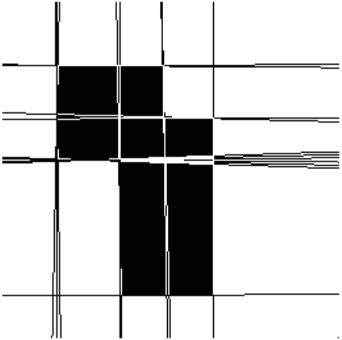
- If we know the orientation of the edge – usually available from the edge detection step
  - We fix θ in the parameter space and increment only one counter!
  - We can allow for orientation uncertainty by incrementing a few counters around the “nominal” counter

Example



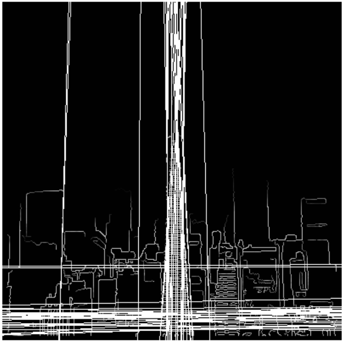
Example

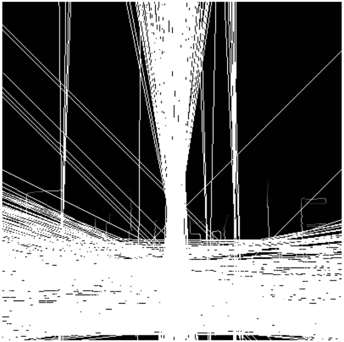




Example

Example





Active or Deformable Contours

- How to fit a curve of arbitrary shape to a set of image edge points? (restricted to closed contours only)
- General closed curves can be represented by snake (also called active contour or deformable contour)

- Deformable models represent
  - Class of objects of differing shape (bananas)
  - Objects which change shape (such as lips)
- Deformable models may be
  - 3D surface, (a balloon squeezed out of shape)
  - 3D space curves, which we bend to form figures
  - 2D contours, e.g. the Snake

Active or Deformable Contours

- How to fit a curve of arbitrary shape to a set of image edge points? (restricted to closed contours only)
- General closed curves can be represented by snake (also called active contour or deformable contour)

- Deformable models represent
  - Class of objects of differing shape (bananas)
  - Objects which change shape (such as lips)
- Deformable models may be
  - 3D surface, (a balloon squeezed out of shape)
  - 3D space curves, which we bend to form figures
  - 2D contours, e.g. the Snake

Deformable Contours

- Goal
  - Start with image and initial closed curve
  - Evolve curve to lie along “important” feature: Edges, Corners, Detected features, User input

- The concept of a snake applied to computer vision
  - It is an elastic band of arbitrary shape
  - It is sensitive to the image gradient
  - Initially it is located near the image contour of interest
  - It can wiggle in the image
  - It is represented as a necklace of points
  - It is then attracted towards the target contour by forces depending on the intensity gradient

Deformable Contours

- Goal
  - Start with image and initial closed curve
  - Evolve curve to lie along “important” feature: Edges, Corners, Detected features, User input

- The concept of a snake applied to computer vision
  - It is an elastic band of arbitrary shape
  - It is sensitive to the image gradient
  - Initially it is located near the image contour of interest
  - It can wiggle in the image
  - It is represented as a necklace of points
  - It is then attracted towards the target contour by forces depending on the intensity gradient

Active Contour Models – Snakes

- Introduced by Kass, Witkin, and Terzopoulos
- Framework: energy minimization
  - Bending and stretching curve = more energy
  - Good features = less energy
  - Curve evolves to minimize energy
- The key idea of deformable contour
  - To associate an energy functional to each possible contour shape such that the image contour to be detected corresponds to a minimum of functional
  - The snake is applied to the intensity image
  - Other curve fitting algorithms are applied to edge points

Snake

- User-Visible Options
  - Initialization: user-specified, automatic
  - Curve properties: continuity, smoothness
  - Image features: intensity, edges, corners, ...
  - Other forces: hard constraints, springs, attractors, ...
  - Scale: local, multi-resolution, global

- Behind-the-Scenes Options
  - Framework: energy minimization, forces acting on curve
  - Curve representation: ideal curve, sampled, spline, implicit function
  - Evolution method: calculus of variations, numerical differential equations, local search

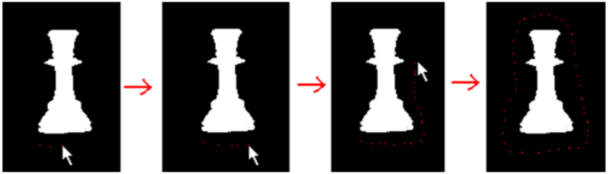
Snake

- User-Visible Options
  - Initialization: user-specified, automatic
  - Curve properties: continuity, smoothness
  - Image features: intensity, edges, corners, ...
  - Other forces: hard constraints, springs, attractors, ...
  - Scale: local, multi-resolution, global

- Behind-the-Scenes Options
  - Framework: energy minimization, forces acting on curve
  - Curve representation: ideal curve, sampled, spline, implicit function
  - Evolution method: calculus of variations, numerical differential equations, local search

Main Idea

- "Drop" a snake
- Let the snake "wiggle" attracted by image gradient, until it glues itself against a contour



Introductory Demo

Active Contour Model – Snakes

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

Active Contour Model – Snakes

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

Active Contour Model – Snakes

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

Active Contour Model – Snakes

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

Active Contour Model – Snakes

- Active contour models may be used in image segmentation and understanding, and are also suitable for analysis of dynamic image data or 3D image data
- It is defined as an energy-minimizing spline – the snake's energy depends on its shape and location within the image
- Local minima of this energy then correspond to desired image properties
- The snake is active, always minimizing its energy functional, therefore exhibiting dynamic behavior
- The idea behind deformable contours is to find a contour $c(s)$ which best approximates the perimeter of an object
- The approach is to construct an energy functional which measures the appropriateness of a contour and to optimize this functional with respect to the contour parameters

Energy Functional

- Associate to each possible shape and location of the snake a value E
  - Values should be such that the image contour to be detected has the minimum value
  - E is called the energy of the snake
- Keep wiggling the snake towards smaller value
- We need a function that given a snake state, associates to it an energy value
- The function should be designed so that the snake moves towards the contour that we are seeking!

Energy Functional

- Associate to each possible shape and location of the snake a value E
  - Values should be such that the image contour to be detected has the minimum value
  - E is called the energy of the snake
- Keep wiggling the snake towards smaller value
- We need a function that given a snake state, associates to it an energy value
- The function should be designed so that the snake moves towards the contour that we are seeking!

What Moves The Snake

- Forces moving the snake (External)
  - It needs to be attracted to contours:
    - Edge pixels "pull" the snake points
    - The stronger the edge, the stronger the pull
    - The force is proportional to $|\nabla|$
- Forces preserving the snake (Internal)
  - The snake should not break apart!
    - Points on the snake must stay close to each other
    - Each point on the snake pulls its neighbors
    - The farther the neighbor, the stronger the force
    - The force is proportional to the distance $|\mathbf{P}_i - \mathbf{P}_{i-1}|$
  - The snake should avoid "oscillations"
    - Penalize high curvature
    - Force proportional to snake curvature

What Moves The Snake

- Forces moving the snake (External)
  - It needs to be attracted to contours:
    - ★ Edge pixels “pull” the snake points
    - ★ The stronger the edge, the stronger the pull
    - ★ The force is proportional to $|\nabla|$
- Forces preserving the snake (Internal)
  - The snake should not break apart!
    - ★ Points on the snake must stay close to each other
    - ★ Each point on the snake pulls its neighbors
    - ★ The farther the neighbor, the stronger the force
    - ★ The force is proportional to the distance $|\mathbf{P}_i - \mathbf{P}_{i-1}|$
  - The snake should avoid “oscillations”
    - ★ Penalize high curvature
    - ★ Force proportional to snake curvature

Energy Functional

- The minimized energy functional is a weighted combination of internal and external forces
  - Internal forces – emanate from the shape of the snake
  - External forces – come from the image and/or from high-level image understanding processes
- The snake is defined parametrically as $v(s) = (x(s), y(s))$, where $x(s), y(s)$ are x, y coordinates along the contour and $s \in [0, 1]$
- The energy functional to be minimized may be written as

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))ds = \int_0^1 \{[E_{int}(\mathbf{v}(s))] + [E_{image}(\mathbf{v}(s))] + [E_{con}(\mathbf{v}(s))]$$

  - E_{int} – the internal energy of the spline due to bending
  - E_{image} – image forces
  - E_{con} – external constraint

Energy Functional

- The minimized energy functional is a weighted combination of internal and external forces
  - Internal forces – emanate from the shape of the snake
  - External forces – come from the image and/or from high-level image understanding processes
- The snake is defined parametrically as $v(s) = (x(s), y(s))$, where $x(s), y(s)$ are x, y coordinates along the contour and $s \in [0, 1]$
- The energy functional to be minimized may be written as

$$E^*_{snake} = \int_0^1 E_{snake}(\mathbf{v}(s))ds = \int_0^1 \{[E_{int}(\mathbf{v}(s))]+[E_{image}(\mathbf{v}(s))]+[E_{con}(\mathbf{v}(s))]$$

  - E_{int} – the internal energy of the spline due to bending
  - E_{image} – image forces
  - E_{con} – external constraint

Internal Energy

- The internal spline energy can be written as

$$E_{int} = \alpha(s) \left| \frac{d\mathbf{v}}{ds} \right|^2 + \beta(s) \left| \frac{d^2\mathbf{v}}{ds^2} \right|^2$$

  where $\alpha(s), \beta(s)$ specify the elasticity and stiffness of the snake
- First term is "membrane" term – minimum energy when curve minimizes length ("soap bubble")
- Second term is "thin plate" term – minimum energy when curve is smooth
- Control α and β to vary between extremes
- Set β to 0 at a point to allow corner (2nd-order discontinuous)

Image Energy

- The second term of the energy integral is derived from the image data over which the snake lies
- Variety of terms gives different effects
- For example, a weighted combination of three different functionals is presented which attracts the snake to lines, edges and terminations:

$$E_{image} = w_{int}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$$

- The line-based functional $E_{line} = f(x,y)$
- The edge-based functional $E_{edge} = -|\nabla f(x,y)|^2$ attracts the snake to contours with large image gradients (location of strong edges)
- Line terminations and corners may influence the snake using a weighted energy functional $E_{term} = \dfrac{\partial \phi}{\partial \mathbf{n}_R}$ where $\phi(x,y)$ denotes the gradient direction along the spline, etc.

Constraint Forces

- The third term of the integral comes from external constraints imposed either by a user or some other high-level process which may force the snake toward or away from particular features
- If the snake is near to some desired feature, the energy minimization will pull the snake the rest of the way
- If the snake settles in a local energy minimum that a high-level process determines as incorrect, an area of energy peak may be made at this location to force the snake away to a different local minimum
- Spring: $E_{con} = k|\mathbf{v} - \mathbf{x}|^2$
- Repulsion: $E_{con} = \dfrac{k}{|\mathbf{v} - \mathbf{x}|^2}$

Minimization

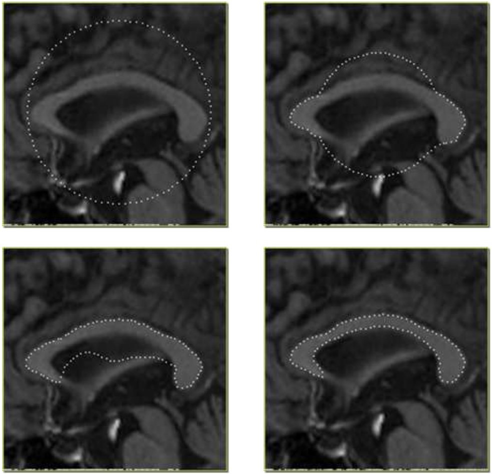
- A contour is defined to lie in the position in which the snake reaches a local energy minimum
- The functional to be minimized is

$$E_{snake}^{*} = \int_{0}^{1} E_{snake}[\mathbf{v}(s)]ds$$

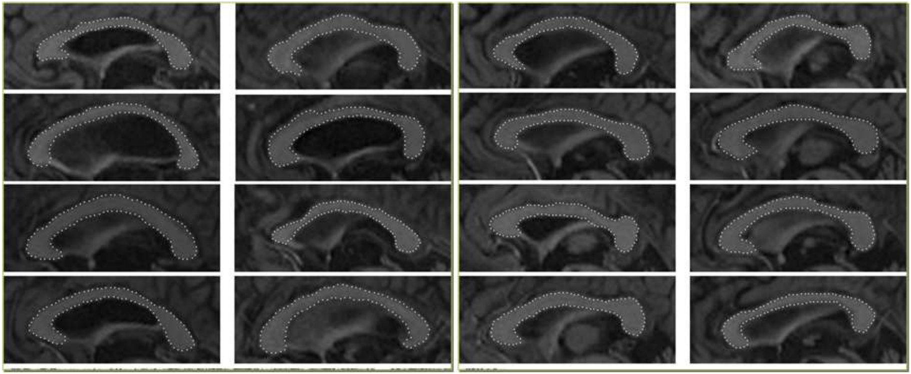
- From the calculus of variations, the Euler-Lagrange condition states that the spline v(s) which minimizes E_{snake}^{*} must satisfy

$$-\frac{d^2}{ds^2}\left(\frac{\partial E}{\partial\left(\frac{d^2x}{ds^2}\right)} + \frac{\partial E}{\partial\left(\frac{d^2y}{ds^2}\right)}\right) + \frac{d}{ds}E_{\mathbf{v}_s} - E_{\mathbf{v}} = 0$$

Corpus Callosum



Corpus Callosum



Evolving Curve

- Computing forces on $\mathbf{v}$ that locally minimize energy gives differential equation for $\mathbf{v}$
- Discretize $\mathbf{v}$: samples (x_i, y_i)
- Approximate derivatives using finite differences
- Numerical solver iteratively converges to minimum
- Write equations directly in terms of forces, not energy
- Implicit equation solver
- Search neighborhood of each (x_i, y_i) for pixel that minimizes energy
- Exact solution: calculus of variations

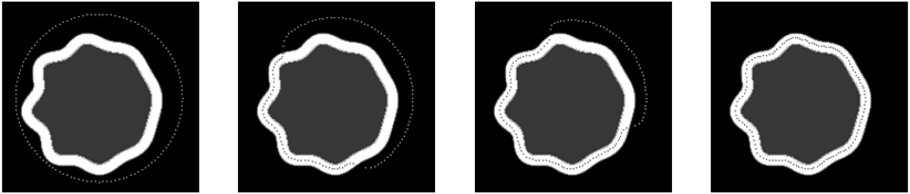
Some Comments

- Advantage of Snakes
  - Easy to manipulate (intuitive)
  - Sensitive to image scale by Gaussian smoothing in the image energy function
  - Insensitive to noise and other ambiguities in the images
  - They can be used to track dynamic objects in temporal as well as the spatial dimensions
- Disadvantage of Snakes
  - Often get stuck in local minima states
    - Overcome by simulated annealing techniques at the expense of longer computation times
  - Often overlook minute features in the process of minimizing the energy over the entire path of their contours
  - Their accuracy is governed by the convergence criteria used in the energy minimization technique – Higher accuracy requires tighter convergence criteria and longer computation

Some Comments

- Advantage of Snakes
  - Easy to manipulate (intuitive)
  - Sensitive to image scale by Gaussian smoothing in the image energy function
  - Insensitive to noise and other ambiguities in the images
  - They can be used to track dynamic objects in temporal as well as the spatial dimensions
- Disadvantage of Snakes
  - Often get stuck in local minima states
    - Overcome by simulated annealing techniques at the expense of longer computation times
  - Often overlook minute features in the process of minimizing the energy over the entire path of their contours
  - Their accuracy is governed by the convergence criteria used in the energy minimization technique – Higher accuracy requires tighter convergence criteria and longer computation

Brain Cortex Segmentation



Add energy term for constant-color regions of a single color

Scale

- In the simplest snakes algorithm, image features only attract locally
- Greater region of attraction: smooth image
  - Curve might not follow high-frequency detail
- Multi-resolution processing
- Looking for global minimum vs. local minima
  - Start with smoothed image to attract curve
  - Finish with unsmoothed image to get details

Scale

- In the simplest snakes algorithm, image features only attract locally
- Greater region of attraction: smooth image
  - Curve might not follow high-frequency detail
- Multi-resolution processing
- Looking for global minimum vs. local minima
  - Start with smoothed image to attract curve
  - Finish with unsmoothed image to get details

Scale

- In the simplest snakes algorithm, image features only attract locally
- Greater region of attraction: smooth image
  - Curve might not follow high-frequency detail
- Multi-resolution processing
- Looking for global minimum vs. local minima
  - Start with smoothed image to attract curve
  - Finish with unsmoothed image to get details

A Greedy Algorithm

- A greedy algorithm for deformable model
  - Makes locally optimal choices and hopes to lead to a globally optimal solution
  - Simplicity:
    - ★ Knowledge of the calculus of variations is not required
  - Low computational complexity:
    - ★ The number of iterations to converge proportional to the number of movement in each iteration times the number of contour points
- The energy functional is given by

$$\epsilon = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds$$

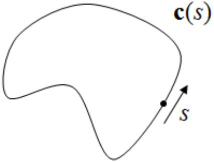
A Greedy Algorithm

- A greedy algorithm for deformable model
  - Makes locally optimal choices and hopes to lead to a globally optimal solution
  - Simplicity:
    - Knowledge of the calculus of variations is not required
  - Low computational complexity:
    - The number of iterations to converge proportional to the number of movement in each iteration times the number of contour points
- The energy functional is given by

$$\epsilon = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds$$

The Energy Functional

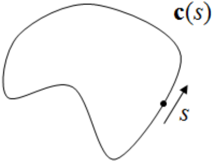
- The three energy terms:
  - Internal energy: E_{cont} and E_{curv} are for continuity and smoothness of the snake
  - External energy: E_{image} is for edge attraction
- α, β, γ control the relative influence of the energy term (can vary along the curve)



$$\epsilon = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds$$

The Energy Functional

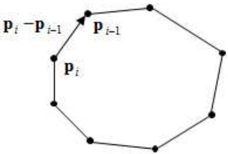
- The three energy terms:
  - Internal energy: E_{cont} and E_{curv} are for continuity and smoothness of the snake
  - External energy: E_{image} is for edge attraction
- α, β, γ control the relative influence of the energy term (can vary along the curve)



$$\epsilon = \int (\alpha(s)E_{cont} + \beta(s)E_{curv} + \gamma(s)E_{image})ds$$

The Energy Terms

- Continuity term: $E_{cont} = \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^2$
  - To make equal space between points
- Smoothness term: $E_{curv} = \|\mathbf{p}_{i-1} - 2\mathbf{p}_i + \mathbf{p}_{i+1}\|^2$
  - To avoid oscillation, penalize high contour curvature
- Edge attraction term: $E_{image} = -\|\nabla I\|$
  - Computed at each snake point, becomes very small (negative) near image edges (large gradient)



Snake Problem

- Given N initial points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$, representing the initial position of the snake, fit the target image contour by minimizing the energy functional

$$\sum_{i=1}^{N} (\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image})$$

- Greedy minimization
  - The neighborhood over which the energy functional is locally minimized is typically small (3×3 or 5×5 window)
  - The local minimization is done by direct comparison of the energy functional values at each location
- Corner elimination
  - If a curvature maximum is found at point $\mathbf{p}_i$, then set β_i to zero to make the deformable contour piecewise smooth

Snake Problem

- Given N initial points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$, representing the initial position of the snake, fit the target image contour by minimizing the energy functional

$$\sum_{i=1}^{N} (\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image})$$

- Greedy minimization
  - The neighborhood over which the energy functional is locally minimized is typically small (3×3 or 5×5 window)
  - The local minimization is done by direct comparison of the energy functional values at each location
- Corner elimination
  - If a curvature maximum is found at point $\mathbf{p}_i$, then set β_i to zero to make the deformable contour piecewise smooth

Snake Problem

- Given N initial points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N$, representing the initial position of the snake, fit the target image contour by minimizing the energy functional

$$\sum_{i=1}^{N} (\alpha_i E_{cont} + \beta_i E_{curv} + \gamma_i E_{image})$$

- Greedy minimization
  - The neighborhood over which the energy functional is locally minimized is typically small (3×3 or 5×5 window)
  - The local minimization is done by direct comparison of the energy functional values at each location
- Corner elimination
  - If a curvature maximum is found at point $\mathbf{p}_i$, then set β_i to zero to make the deformable contour piecewise smooth

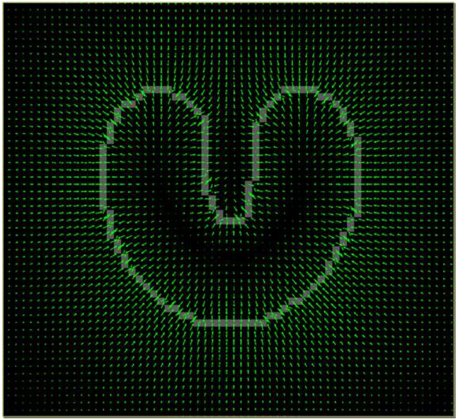
Remarks

- The values of the parameters α_i, β_i and γ_i can be all set to 1, or $\alpha_i = \beta_i = 1, \gamma_i = 1.2$ (more weight on edge attraction)
- To prevent noisy corner:
  - A point is a corner if and only if the curvature is locally maximum at that point, and
  - The norm of the intensity gradient at that point is sufficiently large

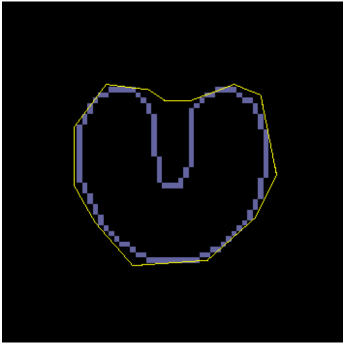
Diffusion-Based Methods

- Another way to attract curve to localized features: vector flow or diffusion methods
- Example:
  - Find edges using Canny
  - For each point, compute distance to nearest edge
  - Push curve along gradient of distance field

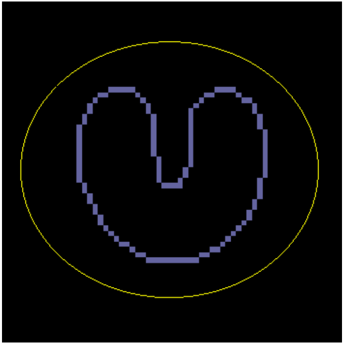
Check http://iacl.ece.jhu.edu/projects/gvf/ for more details.



Gradient Vector Fields



Simple Snake



With Gradient Vector Field

Gradient Vector Fields