EECS 4422/5323 Computer Vision

Unit 3: Image Features

Outline

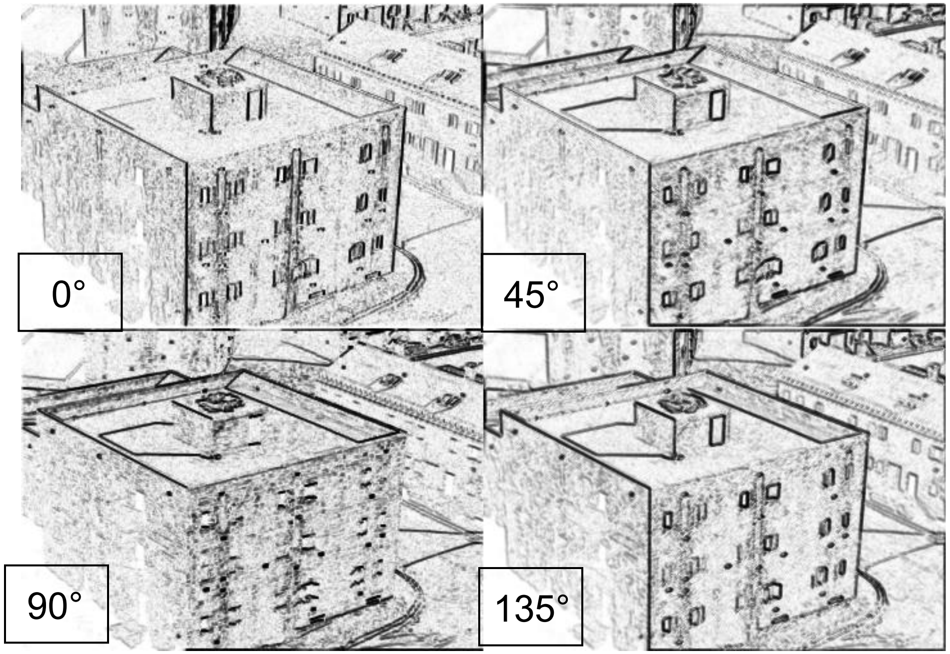
- Introduction
- Edge detection
- Corner detection
- Hough transform
- Deformable templates

Introduction: Motivation

Incremental abstraction

- We have investigated approaches to representing images so as to make local stuctural information explicit.
  - Local scale
  - Local orientation
- Information content in images can be fairly localized.
  - Abrupt changes in image irradiance: edges, corners,…
  - Configurations of intensity changes corresponding to simple patterns: extended lines, circles,…

Source image



Edge features detected at 4 orientations

Definition

- Image features are local, meaningful and detectable parts of an image.
- Local implies limited spatial support.
- Meaningful implies that they can be of use to subsequent operations.
- Detectable implies that we can develop an algorithm for extracting the position and description of these entities given image data.

Outline

- Introduction
- **Edge detection**
- Corner detection
- Hough transform
- Deformable templates

Edge detection: Basics

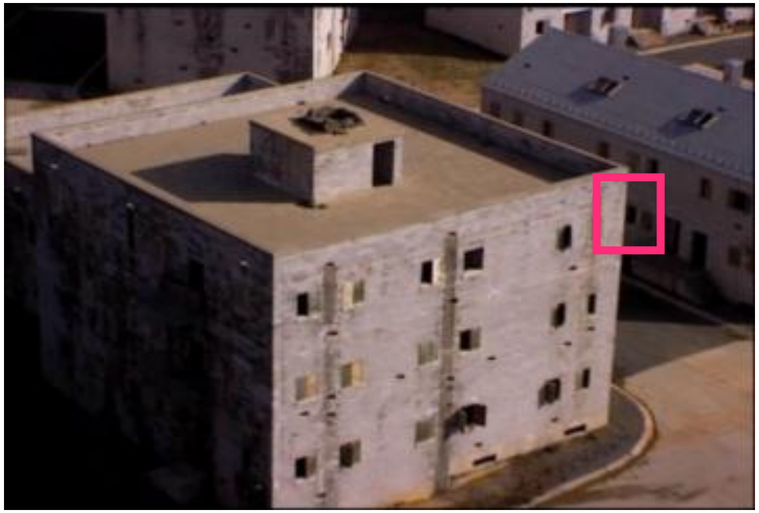
What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance

Edge detection: Basics

What is an edge

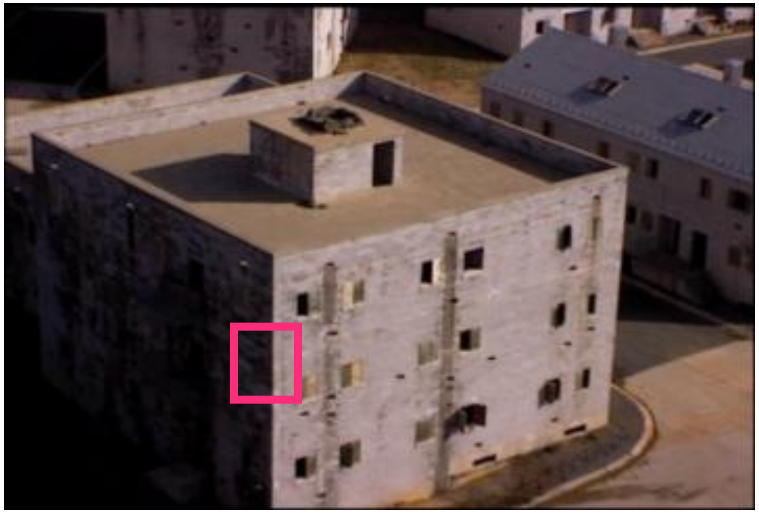
- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance



Edge detection: Basics

What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance



Edge detection: Basics

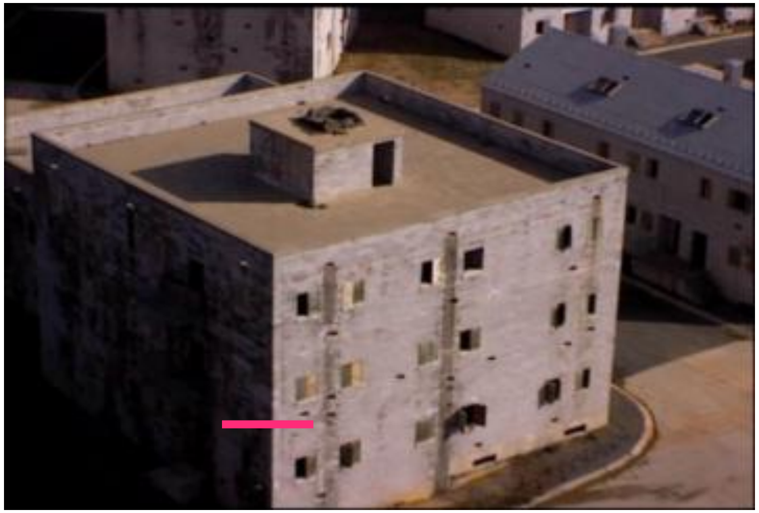
What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance

Edge detection: Basics

What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance



Simple model

- Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.

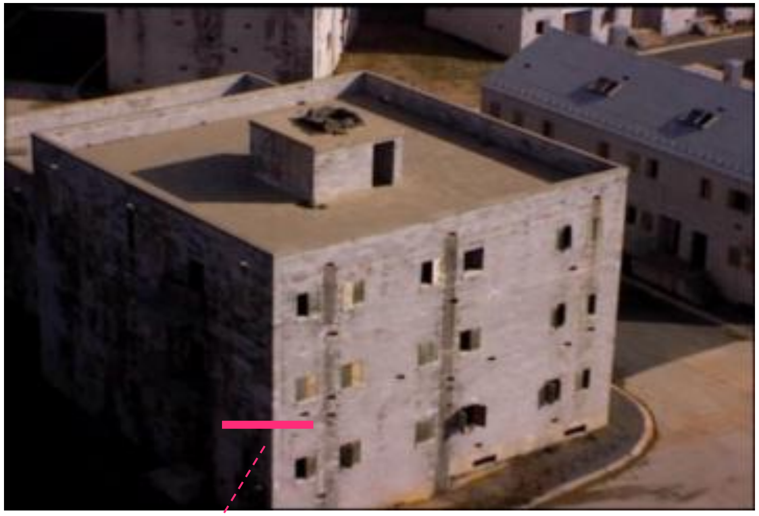
Edge detection: Basics

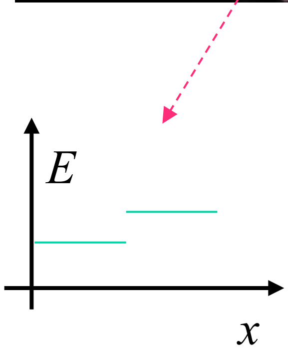
What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance

Simple model

- Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.
- Ideally, we might expect to find a step change in a plot of brightness vs. position.



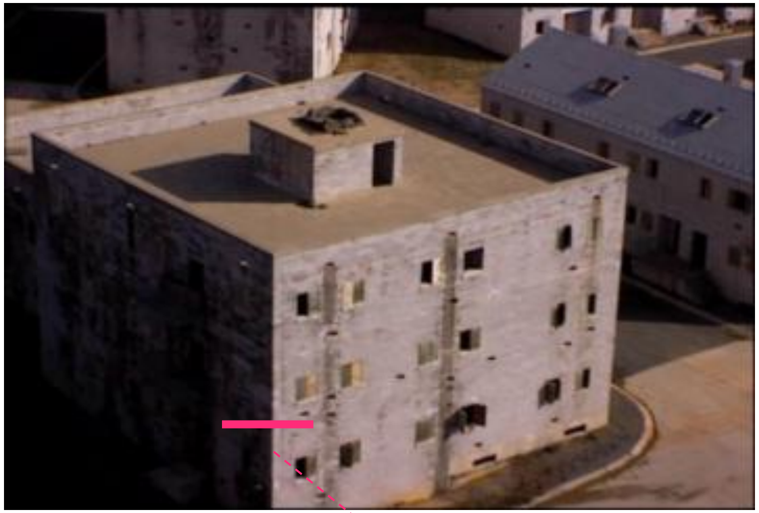


Ideal step edge

Edge detection: Basics

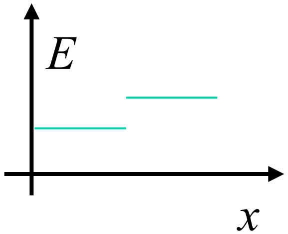
What is an edge

- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance

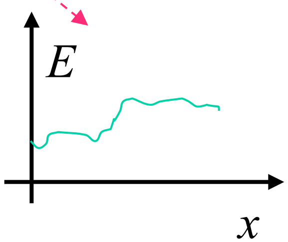


Simple model

- Consider a one-dimensional slice through an image in the vicinity of an abrupt change in image brightness.
- Ideally, we might expect to find a step change in a plot of brightness vs. position.
- In practice, we find a corrupted version of this ideal
  - Step transition smoothed
  - Noise transitions superimposed.



Ideal step edge



Real step edge

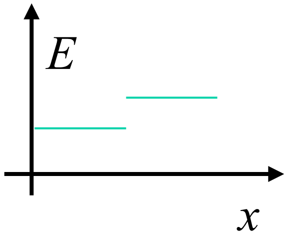
Edge detection: Basics

What is an edge

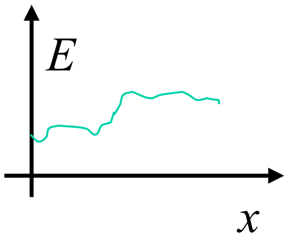
- Intuitively, an edge is a border between two regions, each of which have approximately uniform brightness.
- In an image, edges often arise as the result of occluding contours in an image
  - The two image regions correspond to two different surfaces.
- Other sources of image edges include
  - Abrupt changes in surface orientation
  - Discontinuities in surface reflectance

General approach

1. Suppress noise.
2. Enhance edges.
3. Locate edges.



Ideal step edge



Real step edge

Edge detection: Differential operators for enhancement

Formalizing the edge model

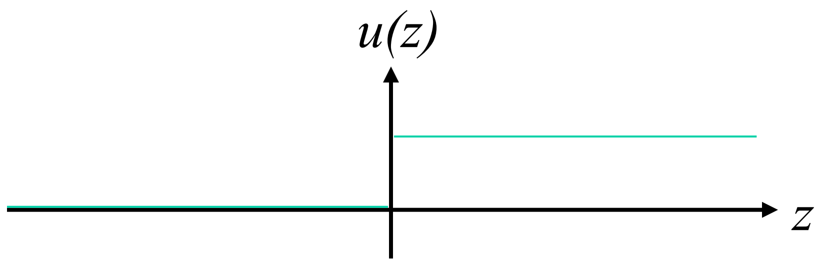
- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.

Edge detection: Differential operators for enhancement

Formalizing the edge model

- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$

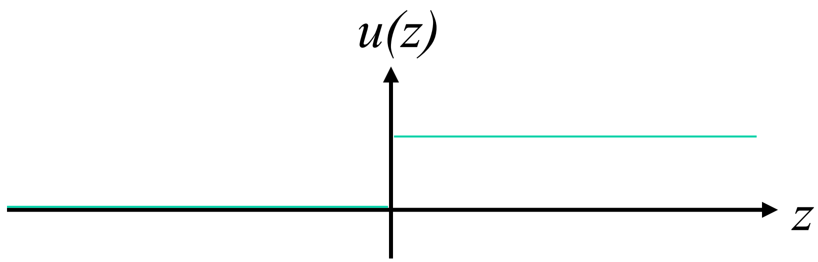


Edge detection: Differential operators for enhancement

Formalizing the edge model

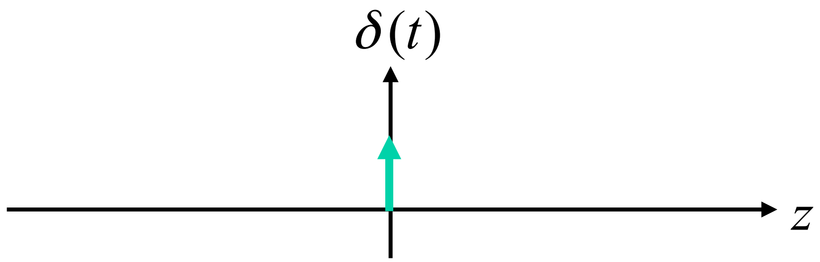
- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$



- We note that *u(z)* is just the integral of the one-dimensional unit impulse, i.e.,

$$u(z) = \int_{-\infty}^{z} \delta(t)dt$$



Edge detection: Differential operators for enhancement

Formalizing the edge model

- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

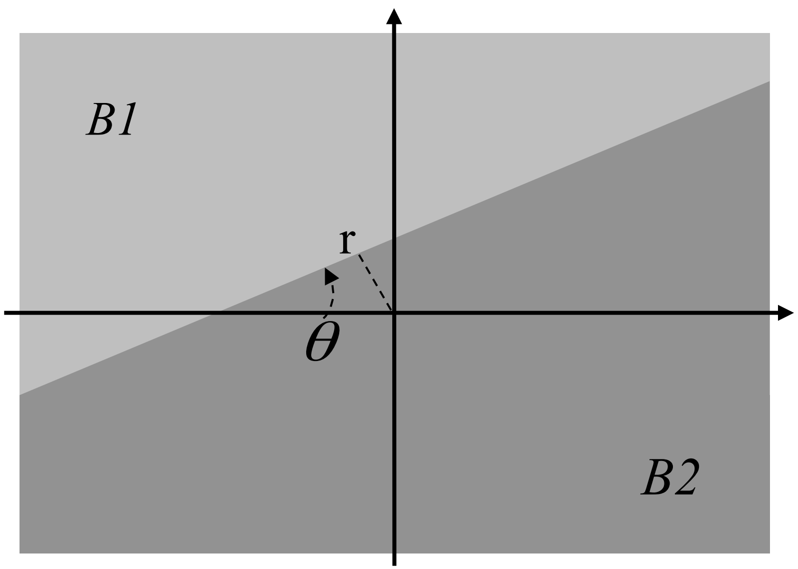
$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$

- We note that $u(z)$ is just the integral of the one-dimensional unit impulse, i.e.,

$$u(z) = \int_{-\infty}^{z} \delta(t)dt$$

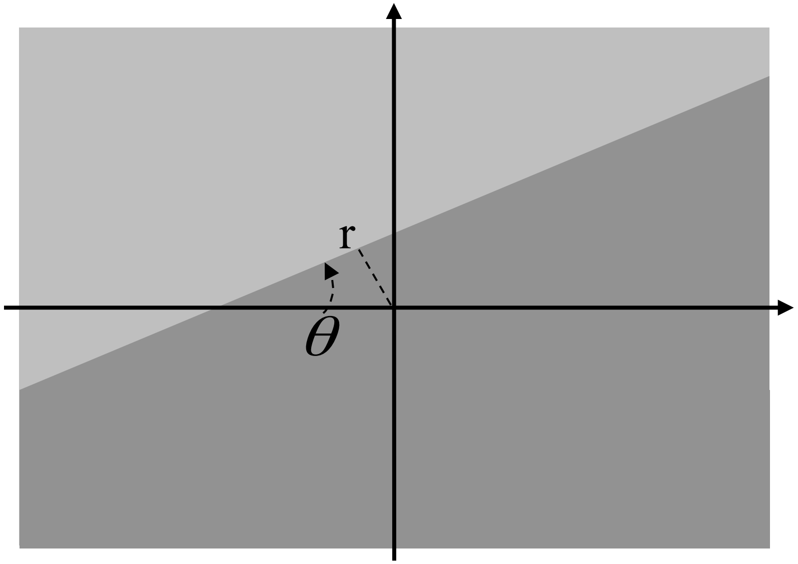
- Suppose that the edge lies along the line

$$x \sin\theta - y\cos\theta + r = 0$$



Interlude: r, θ line parameterization

$$x\sin\theta - y\cos\theta + r = 0$$

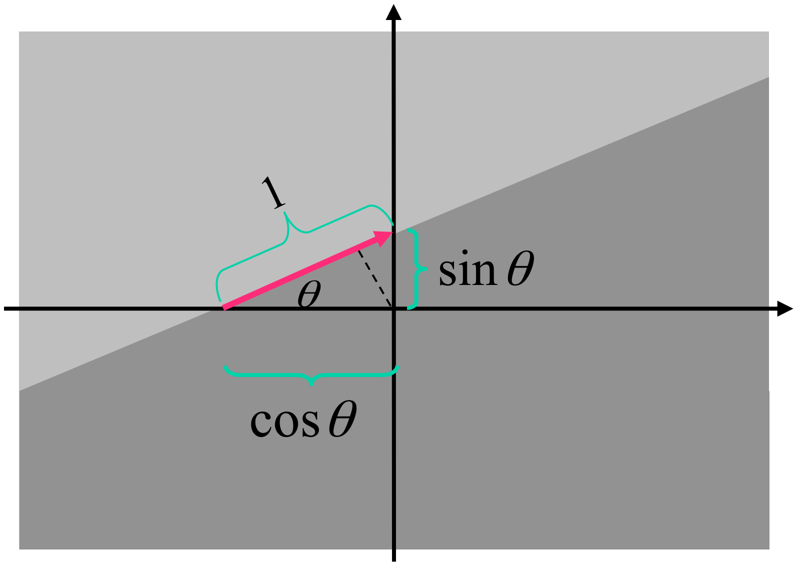


Observations

- The line intersects the x-axis at $-r/\sin\theta$
- The line intersects the y-axis at $r/\cos\theta$
- The closest point on the line to the origin is $(-r\sin\theta, r\cos\theta)$

Interlude: r, θ line parameterization

$$x\sin\theta - y\cos\theta + r = 0$$



Observations

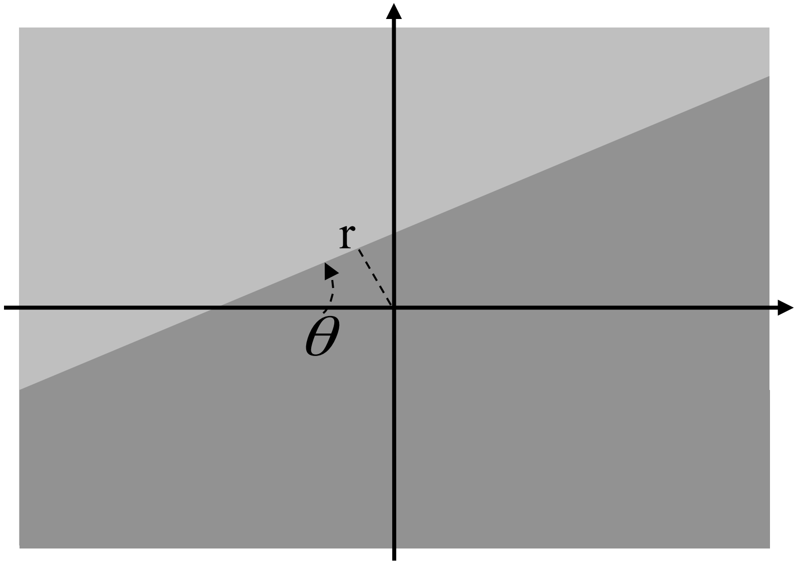
- The line intersects the x-axis at $-r/\sin\theta$
- The line intersects the y-axis at $r/\cos\theta$
- The closest point on the line to the origin is $(-r\sin\theta, r\cos\theta)$
- Parametrically we write

$$x_0 = -r\sin\theta + s\cos\theta$$

$$y_0 = r\cos\theta + s\sin\theta$$

Interlude: r, θ line parameterization

$$x \sin\theta - y\cos\theta + r = 0$$



Observations

- The line intersects the x-axis at $-r/\sin\theta$
- The line intersects the y-axis at $r/\cos\theta$
- The closest point on the line to the origin is $(-r\sin\theta, r\cos\theta)$
- Parametrically we write

$$x_0 = -r\sin\theta + s\cos\theta$$

$$y_0 = r\cos\theta + s\sin\theta$$

Exercise

- Given a point (x,y), find the nearest point (x0,y0) on the line and its distance.
- We define a distance

$$d^2 = (x - x_0)^2 + (y - y_0)^2$$

Interlude: r, θ line parameterization

Exercise

- Given a point (x,y), find the nearest point (x0,y0) on the line and its distance.
- We define a distance

$$d^2 = (x - x_0)^2 + (y - y_0)^2$$

- Substituting our parametric expressions for (x0,y0) we obtain

$$d^2 = (x^2 + y^2) + r^2 + 2r(x\sin\theta - y\cos\theta) - 2s(x\cos\theta + y\sin\theta) + s^2$$

- Differentiating WRT s and setting to zero leads to

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

- This result can be substituted back into the parameteric equations for (x0,y0).
- To find the distance to the line we compute the differences

$$x - x_0 = \sin\theta(x\sin\theta - y\cos\theta + r)$$

$$y - y_0 = -\cos\theta(x\sin\theta - y\cos\theta + r)$$

- And enter into the distance formula to yield

$$d^2 = (x\sin\theta - y\cos\theta + r)^2$$

- We conclude that the distance of a point from the line is given by the expression of the line itself!

Interlude: r, θ line parameterization

Exercise

- Given a point (x,y), find the nearest point (x0,y0) on the line and its distance.
- We define a distance

$$d^2 = (x - x_0)^2 + (y - y_0)^2$$

$$x_0 = -r\sin\theta + s\cos\theta$$
$$y_0 = r\cos\theta + s\sin\theta$$

- Substituting our parametric expressions for (x0,y0) we obtain

$$d^2 = (x^2 + y^2) + r^2 + 2r(x\sin\theta - y\cos\theta) - 2s(x\cos\theta + y\sin\theta) + s^2$$

- Differentiating WRT s and setting to zero leads to

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

- This result can be substituted back into the parameteric equations for (x0,y0).
- To find the distance to the line we compute the differences

$$x - x_0 = \sin\theta(x\sin\theta - y\cos\theta + r)$$
$$y - y_0 = -\cos\theta(x\sin\theta - y\cos\theta + r)$$

- And enter into the distance formula to yield

$$d^2 = (x\sin\theta - y\cos\theta + r)^2$$

- We conclude that the distance of a point from the line is given by the expression of the line itself!

Edge detection: Differential operators for enhancement

Formalizing the edge model

- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$

- We note that $u(z)$ is just the integral of the one-dimensional unit impulse, i.e.,

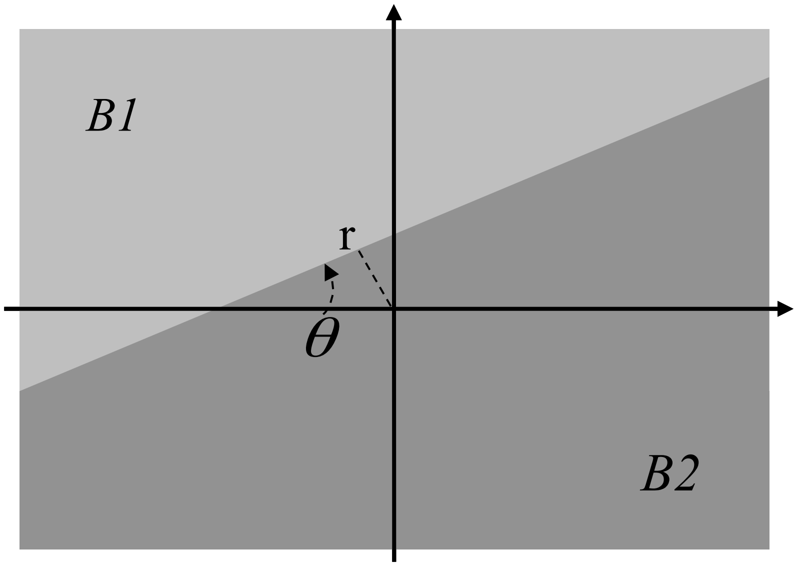
$$u(z) = \int_{-\infty}^{z} \delta(t)dt$$

- Suppose that the edge lies along the line

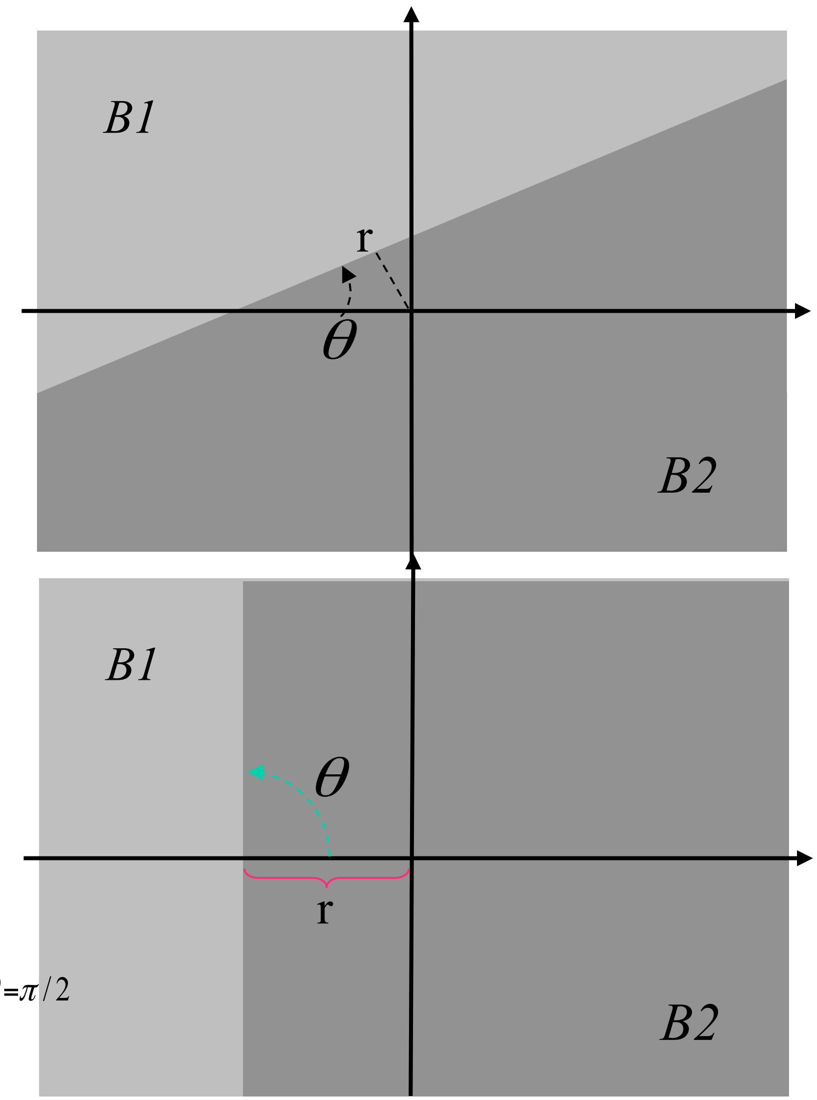
$$x \sin\theta - y\cos\theta + r = 0$$

- Then we write the image brightness as

$$E(x, y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$



Edge detection: Differential operators for enhancement



Formalizing the edge model

- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$

- We note that $u(z)$ is just the integral of the one-dimensional unit impulse, i.e.,

$$u(z) = \int_{-\infty}^{z} \delta(t)dt$$

- Suppose that the edge lies along the line

$$x\sin\theta - y\cos\theta + r = 0$$

- Then we write the image brightness as

$$E(x,y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)\big|_{\theta=\pi/2}$$
$$= B1 + (B2 - B1)u(x + r)$$

Edge detection: Differential operators for enhancement

Formalizing the edge model

- Keeping in mind the limitations of the ideal step edge model…
- We formalize the model as follows.
- Define the unit step function as

$$u(z) = \begin{cases} 1, z > 0 \\ 1/2, z = 0 \\ 0, z < 0 \end{cases}$$

- We note that *u(z)* is just the integral of the one-dimensional unit impulse, i.e.,

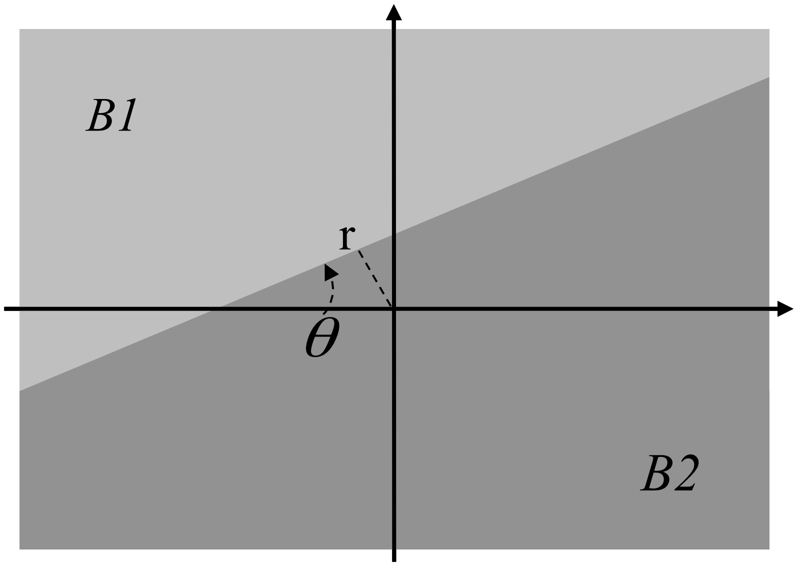
$$u(z) = \int_{-\infty}^{z} \delta(t)dt$$

- Suppose that the edge lies along the line

$$x \sin\theta - y\cos\theta + r = 0$$

- Then we write the image brightness as

$$E(x, y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$



Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x, y) = B1 + (B2 - B1)u(x \sin\theta - y \cos\theta + r)$$

Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x, y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$

- We can calculate the partial derivatives

$$\frac{\partial E}{\partial x} = \sin\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

Recall: The "chain rule", let

$h(x)=g[f(x)]$

then

$h'(x)=g'[f(x)]\,f'(x)$

Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x, y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$

- We can calculate the partial derivatives

$$\frac{\partial E}{\partial x} = \sin\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

- We define the brightness gradient as

$$\nabla E = \left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}\right)^{\mathrm{T}}$$

and note that it points along the direction of the edge transition.

Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x,y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$

- We can calculate the partial derivatives

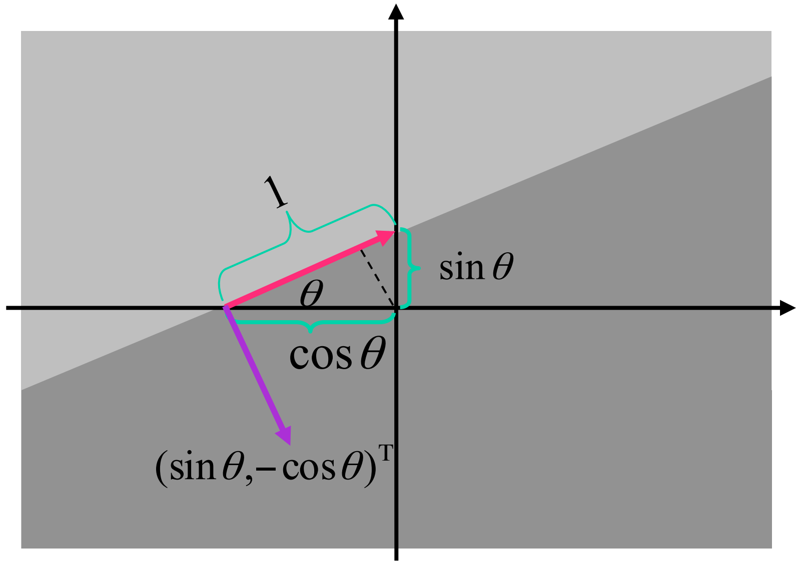
$$\frac{\partial E}{\partial x} = \sin\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

- We define the brightness gradient as

$$\nabla E = \left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}\right)^{\mathrm{T}}$$

and note that it points along the direction of the edge transition.



Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x,y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$

- We can calculate the partial derivatives

$$\frac{\partial E}{\partial x} = \sin\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

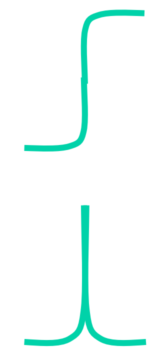
- We define the brightness gradient as

$$\nabla E = \left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}\right)^{T}$$

and note that it points along the direction of the edge transition.

$E(x,y)$

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2$$



- Calculating the squared gradient

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 = [(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)]^2$$

we see that it has magnitude proportional to the brightness jump as we cross the step.

Edge detection: Differential operators for enhancement

The gradient

- Considering our model of the brightness

$$E(x,y) = B1 + (B2 - B1)u(x\sin\theta - y\cos\theta + r)$$

- We can calculate the partial derivatives

$$\frac{\partial E}{\partial x} = \sin\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

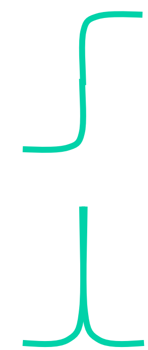
- We define the brightness gradient as

$$\nabla E = \left(\frac{\partial E}{\partial x}, \frac{\partial E}{\partial y}\right)^{\mathrm{T}}$$

and note that it points along the direction of the edge transition.

$E(x,y)$

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2$$



- Calculating the squared gradient

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 = [(B2 - B1)\delta(x\sin\theta - y\cos\theta + r)]^2$$

we see that it has magnitude proportional to the brightness jump as we cross the step.

- Remarks:
  - The response of this operator is independent of the edge orientation.
  - Calculation of the squared gradient is a nonlinear operation.

Edge detection: Differential operators for enhancement

The Laplacian

- Now let us consider the second (partial) derivatives

$$\frac{\partial E}{\partial x} = \sin\theta (B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial E}{\partial y} = -\cos\theta (B2 - B1)\delta(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial x \partial y} = -\sin\theta\cos\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

where we make use of the notation δ' for the unit doublet, the derivative of the unit impulse (introduced previously).

Edge detection: Differential operators for enhancement

The Laplacian

- Keeping in mind that we have

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

Edge detection: Differential operators for enhancement

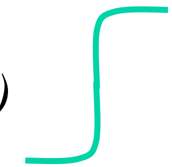
The Laplacian

- Keeping in mind that we have

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

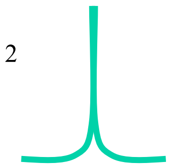
$E(x,y)$



- We define the Laplacian of the image as

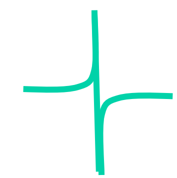
$$\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2$$



- Apparently, this operation will show a “zero-crossing” as we cross an edge.

$\nabla^2 E$



Edge detection: Differential operators for enhancement

The Laplacian

- Keeping in mind that we have

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

- We define the Laplacian of the image as

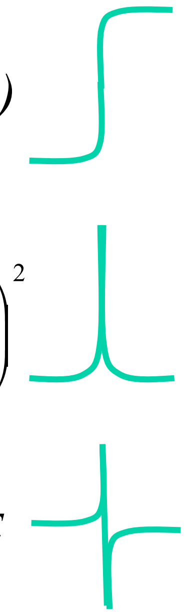
$$\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

- Apparently, this operation will show a “zero-crossing” as we cross an edge.
- We note that (like the squared gradient)
  - The response of the operator is independent of the edge orientation.

$E(x,y)$

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2$$

$\nabla^2 E$



Edge detection: Differential operators for enhancement

The Laplacian

- Keeping in mind that we have

$$\frac{\partial^2 E}{\partial x^2} = \sin^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

$$\frac{\partial^2 E}{\partial y^2} = \cos^2\theta(B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

- We define the Laplacian of the image as

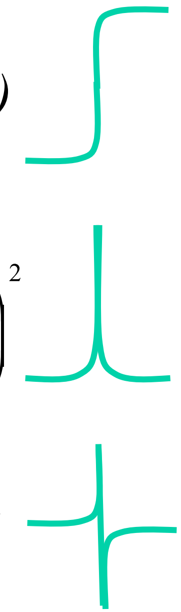
$$\nabla^2 E = \frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} = (B2 - B1)\delta'(x\sin\theta - y\cos\theta + r)$$

- Apparently, this operation will show a "zero-crossing" as we cross an edge.
- We note that (like the squared gradient)
  - The response of the operator is independent of the edge orientation.
- We note that (unlike the squared gradient)
  - The Laplacian is linear
  - The Laplacian preserves the sign of intensity change across the edge.

$E(x,y)$

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2$$

$\nabla^2 E$



Edge detection: Discrete approximations

The gradient

- Considering a 2x2 group of pixels

$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i,j}$	$E_{i+1,j}$

Edge detection: Discrete approximations

The gradient

- Considering a 2x2 group of pixels

$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i,j}$	$E_{i+1,j}$

- Using finite differences, we can then estimate the derivatives at the center of this group as

$$\frac{\partial E}{\partial x} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j+1}) + (E_{i+1,j} - E_{i,j})]$$

$$\frac{\partial E}{\partial y} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})]$$

Edge detection: Discrete approximations

The gradient

- Considering a 2x2 group of pixels

$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i,j}$	$E_{i+1,j}$

- Using finite differences, we can then estimate the derivatives at the center of this group as

$$\frac{\partial E}{\partial x} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j+1}) + (E_{i+1,j} - E_{i,j})]$$

$$\frac{\partial E}{\partial y} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})]$$

- Correspondingly, we calculate the squared gradient estimate as

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j})^2 + (E_{i,j+1} - E_{i+1,j})^2]$$

Edge detection: Discrete approximations

The gradient

- Considering a 2x2 group of pixels

$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i,j}$	$E_{i+1,j}$

- Using finite differences, we can then estimate the derivatives at the center of this group as

$$\frac{\partial E}{\partial x} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j+1}) + (E_{i+1,j} - E_{i,j})]$$

$$\frac{\partial E}{\partial y} \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i+1,j}) + (E_{i,j+1} - E_{i,j})]$$

- Correspondingly, we calculate the squared gradient estimate as

$$\left(\frac{\partial E}{\partial x}\right)^2 + \left(\frac{\partial E}{\partial y}\right)^2 \approx \frac{1}{2\varepsilon}[(E_{i+1,j+1} - E_{i,j})^2 + (E_{i,j+1} - E_{i+1,j})^2]$$

- Performing this calculation over an image of interest, we obtain large values where the image brightness is changing rapidly.
- We write the results in a new image array, in which the edges are strongly emphasized.

Edge detection: Discrete approximations

The Laplacian

- Considering a 3x3 group of pixels

$E_{i-1,j+1}$	$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i-1,j}$	$E_{i,j}$	$E_{i+1,j}$
$E_{i-1,j-1}$	$E_{i,j-1}$	$E_{i+1,j-1}$

Edge detection: Discrete approximations

The Laplacian

- Considering a 3x3 group of pixels

$E_{i-1,j+1}$	$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i-1,j}$	$E_{i,j}$	$E_{i+1,j}$
$E_{i-1,j-1}$	$E_{i,j-1}$	$E_{i+1,j-1}$

- Using finite differences, we estimate the Laplacian at the center of this group using

$$\frac{\partial^2 E}{\partial x^2} \approx \frac{1}{\varepsilon^2}(E_{i-1,j} - 2E_{i,j} + E_{i+1,j})$$

$$\frac{\partial^2 E}{\partial y^2} \approx \frac{1}{\varepsilon^2}(E_{i,j-1} - 2E_{i,j} + E_{i,j+1})$$

Edge detection: Discrete approximations

The Laplacian

- Considering a 3x3 group of pixels

$E_{i-1,j+1}$	$E_{i,j+1}$	$E_{i+1,j+1}$
$E_{i-1,j}$	$E_{i,j}$	$E_{i+1,j}$
$E_{i-1,j-1}$	$E_{i,j-1}$	$E_{i+1,j-1}$

- Using finite differences, we estimate the Laplacian at the center of this group using

$$\frac{\partial^2 E}{\partial x^2} \approx \frac{1}{\varepsilon^2}(E_{i-1,j} - 2E_{i,j} + E_{i+1,j})$$

$$\frac{\partial^2 E}{\partial y^2} \approx \frac{1}{\varepsilon^2}(E_{i,j-1} - 2E_{i,j} + E_{i,j+1})$$

to yield

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} \approx \frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j} + E_{i,j-1} + E_{i+1,j} + E_{i,j+1}) - E_{i,j}\right]$$

Edge detection: Discrete approximations

The Laplacian

- We notice that our discrete Laplacian operation

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} \approx \frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j} + E_{i,j-1} + E_{i+1,j} + Ei, j+1) - E_{i,j}\right]$$

essentially computes a average of the surrounding values and subtracts that of the centre.

Edge detection: Discrete approximations

The Laplacian

- We notice that our discrete Laplacian operation

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} \approx \frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j} + E_{i,j-1} + E_{i+1,j} + Ei,j+1) - E_{i,j}\right]$$

essentially computes a average of the surrounding values and subtracts that of the center.

- We further note that this operation can be mapped onto our local 3x3 mask as

$$\frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array}$$

- Indeed, we can use this mask as the (discrete) PSF in a (discrete) convolution to perform the necessary calculations.

Edge detection: Discrete approximations

The Laplacian

- We notice that our discrete Laplacian operation

$$\frac{\partial^2 E}{\partial x^2} + \frac{\partial^2 E}{\partial y^2} \approx \frac{4}{\varepsilon^2}\left[\frac{1}{4}(E_{i-1,j} + E_{i,j-1} + E_{i+1,j} + Ei, j+1) - E_{i,j}\right]$$

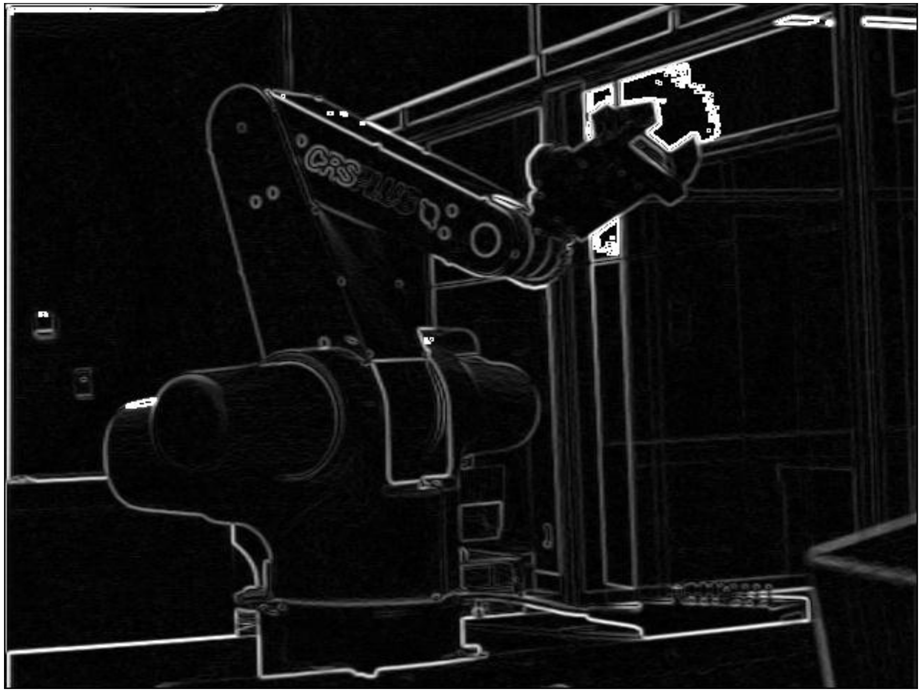
essentially computes a average of the surrounding values and subtracts that of the center.

- We further note that this operation can be mapped onto our local 3x3 mask as

$$\frac{1}{\varepsilon^2}\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

- Indeed, we can use this mask as the (discrete) PSF in a (discrete) convolution to perform the necessary calculations.
- Recall: Previously we noted that
  - the Laplacian is linear
  - And (more generally) differentiation is LSI

Edge detection: Example



Edge detection: Noise suppression

Local operators and noise

- In practice, application of the discrete operations that we have formulated can lead to poor results.
- Recalling that differentiation accentuates high frequency components of the image, we expect that (high frequency) noise will be accentuated as are the edges of interest.
- Our recourse is to rely on the observation that the edges of interest will (typically) have frequency components across a wider range of frequencies (especially) lower frequencies than the noise.

Edge detection: Noise suppression

Local operators and noise

- In practice, application of the discrete operations that we have formulated can lead to poor results.
- Recalling that differentiation accentuates high frequency components of the image, we expect that (high frequency) noise will be accentuated as are the edges of interest.
- Our recourse is to rely on the observation that the edges of interest will (typically) have frequency components across a wider range of frequencies (especially) lower frequencies than the noise.
- If this is the case, a useful noise suppression is to convolve the image with a Gaussian PSF

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

Edge detection: Noise suppression

Local operators and noise

- In practice, application of the discrete operations that we have formulated can lead to poor results.
- Recalling that differentiation accentuates high frequency components of the image, we expect that (high frequency) noise will be accentuated as are the edges of interest.
- Our recourse is to rely on the observation that the edges of interest will (typically) have frequency components across a wider range of frequencies (especially) lower frequencies than the noise.
- If this is the case, a useful noise suppression is to convolve the image with a Gaussian PSF

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

- Interestingly, recalling that
  - Derivatives can be implemented as convolutions
  - Convolution is associative

  we choose to combine the operations of noise suppression and smoothing via application of the PSFs

$$h_x(x,y) = -\frac{x}{2\pi\sigma^4} \exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

$$h_y(x,y) = -\frac{y}{2\pi\sigma^4} \exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

Edge detection: Noise suppression

Toward multiresolution analysis

- In applying the operators in practice,

$$h_x(x,y) = -\frac{x}{2\pi\sigma^4}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

$$h_y(x,y) = -\frac{y}{2\pi\sigma^4}\exp\left[-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)\right]$$

  we frequently will need to select values of the standard deviation so that the resulting PSF has a large spatial support.
- More generally, we may choose to incorporate the notion of multiresolution processing and detect edges using a range of values for the standard deviation.

Edge detection: Example

Edge detection: Localization

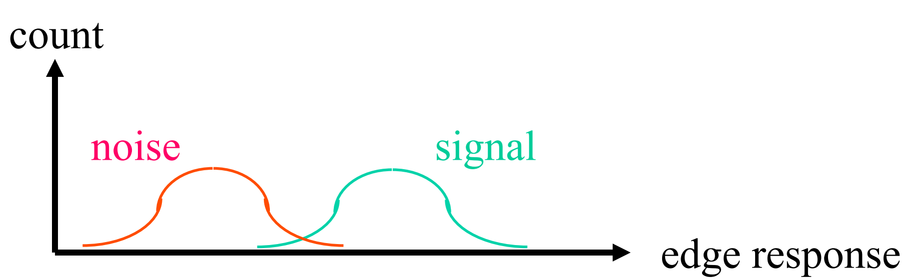
A matter of thresholding

- Having enhanced the edge loci (hopefully while ameliorating the effects of noise).
- We must localize the edges per se.
- In essence, this comes down to selecting a threshold for accepting an (enhanced) image value as corresponding to an edge (as opposed to noise).
- For the gradient magnitude, we seek a decision point above which we will declare a value as marking an edge location.
- For the Laplacian, we seek a transition magnitude across the zero-crossing above which we will declare a value as marking an edge location.
- Remark: Having good a priori models of what is an edge and what is noise in a particular situation can provide a principled basis for threshold selection.

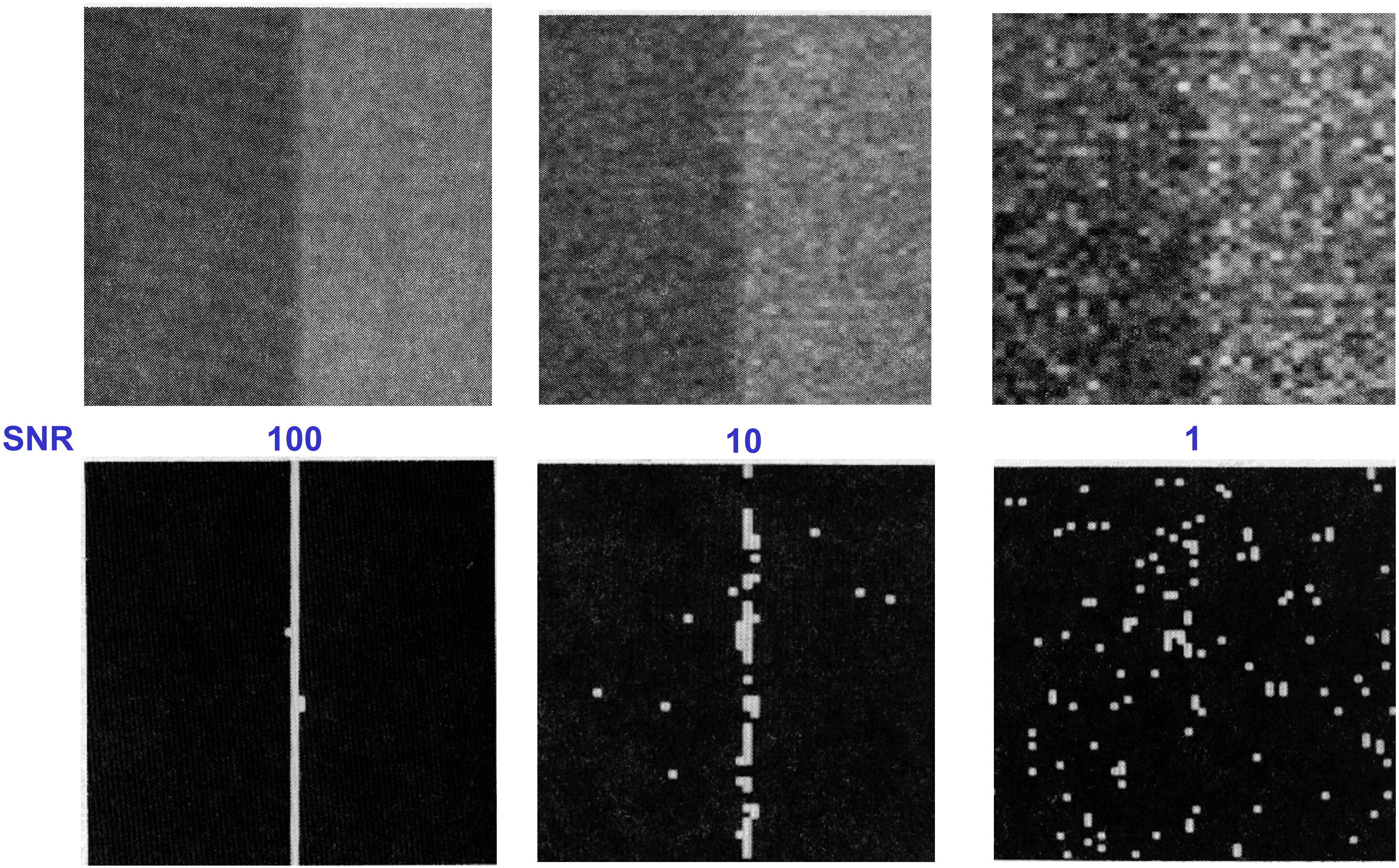
Edge detection: Localization

A matter of thresholding

- Having enhanced the edge loci (hopefully while ameliorating the effects of noise).
- We must localize the edges per se.
- In essence, this comes down to selecting a threshold for accepting an (enhanced) image value as corresponding to an edge (as opposed to noise).
- For the gradient magnitude, we seek a decision point above which we will declare a value as marking an edge location.
- For the Laplacian, we seek a transition magnitude across the zero-crossing above which we will declare a value as marking an edge location.
- Remark: Having good a priori models of what is an edge and what is noise in a particular situation can provide a principled basis for threshold selection.



Edge detection: Example



Edge detection: Additional examples presented in lecture

Edge detection: Recapitulation

Model
- Edges in the image appear as light/dark transitions; typically with physical meaning.
- The ideal step edge.
- But as corrupted by noise.

3 step process
- Suppress noise.
- Enhance edges.
- Locate edges.

Case studies
- (Squared) gradient.
- Laplacian.
- Lots of comparative examples.

Outline

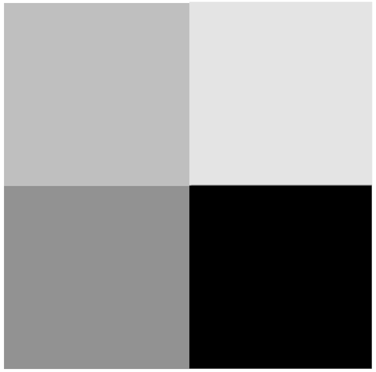
- Introduction
- Edge detection
- Corner detection
- Hough transform
- Deformable templates

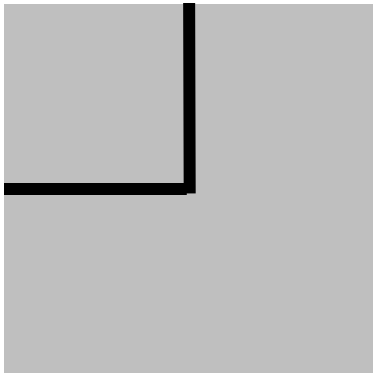
Corner detection: Basics

What is an corner

- A corner is an image location where two distinct image orientations occur in a local region.
- Physically, image corners tend to arise for similar reasons as edges (e.g., changes of reflectance, surface orientation).
- Corners are of interest for two main reasons

1. Corners provide constrain 2 degrees of freedom in a pattern's location.
2. Corners tend to persist across changes in viewpoint.





Corner detection: Differential analysis

The gradient

- Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives Ex and Ey, using subscript notation for partial derivatives

Corner detection: Differential analysis

The gradient

- Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives Ex and Ey, using subscript notation for partial derivatives
- We choose to accumulate these measures over a neighorhood via summation and construct the matrix

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

- We appeal to this matrix because it encapsulates the local orientation structure as captured by the gradients.

Corner detection: Differential analysis

The gradient

- Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives Ex and Ey, using subscript notation for partial derivatives
- We choose to accumulate these measures over a neighorhood via summation and construct the matrix

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

- We appeal to this matrix because it encapsulates the local orientation structure as captured by the gradients.
- To see this, note that we could choose a vector (x, y) which maximizes

$$\sum (E_x x + E_y y)^2 = \sum [(E_x, E_y) \bullet (x, y)]^2$$

as representing the local gradient direction.

Corner detection: Differential analysis

The gradient

- Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives Ex and Ey, using subscript notation for partial derivatives
- We choose to accumulate these measures over a neighorhood via summation and construct the matrix

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

- We appeal to this matrix because it encapsulates the local orientation structure as captured by the gradients.
- To see this, note that we could choose a vector (x, y) which maximizes

$$\sum (E_x x + E_y y)^2$$

as representing the local gradient direction.
- We can rewrite this as expression as

$$\begin{aligned}\sum (E_x x + E_y y)^2 &= \sum (E_x^2 x^2 + 2E_x E_y xy + E_y^2 y^2) \\ &= \left(\sum E_x^2\right)x^2 + 2\left(\sum E_x E_y\right)xy + \left(\sum E_y^2\right)y^2 \\ &= x\left(\sum E_x^2\right)x + 2x\left(\sum E_x E_y\right)y + y\left(\sum E_y^2\right)y \\ &= (x, y)\begin{pmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}\end{aligned}$$

Corner detection: Differential analysis

The gradient

- Given that we are concerned with local measures of orientation, one approach is to calculate the local spatial derivatives Ex and Ey, using subscript notation for partial derivatives
- We choose to accumulate these measures over a neighorhood via summation and construct the matrix

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

- We appeal to this matrix because it encapsulates the local orientation structure as captured by the gradients.
- To see this, note that we could choose a vector (x, y) which maximizes

$$\sum (E_x x + E_y y)^2$$

  as representing the local gradient direction.
- We can rewrite this as expression as

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

  which brings us back to our matrix of concern.

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}; C = R \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R^{-1}$$

Remark: Columns of **R** are the Eigenvectors of **C**.

where λ_1, λ_2 are the eigenvalues of the matrix.

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).
  - Recall the Schwarz inequality: $|\mathrm{a}||\mathrm{b}| \geq |\mathrm{a}^{\mathrm{T}}\mathrm{b}|$

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).
  - Recall the Schwarz inequality: $|\mathrm{a}||\mathrm{b}| \geq |\mathrm{a}^{\mathrm{T}}\mathrm{b}|$

$$\det\begin{pmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{pmatrix} = \sum E_x^2 \sum E_y^2 - \left(\sum E_x E_y\right)^2 \geq 0$$

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).
- Consider three cases

1. If the region of interest is perfectly uniform, then the gradients are identically zero: $\lambda_1 = \lambda_2 = 0$
2. If the region contains and ideal step edge, then there is only one gradient direction: $\lambda_1 > 0, \lambda_2 = 0$
3. If the region contains two orientations, then there are two gradient directions: $\lambda_1 \geq \lambda_2 > 0$

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

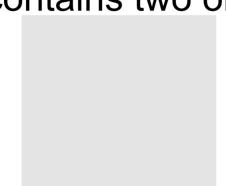
is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).
- Consider three cases

1. If the region of interest is perfectly uniform, then the gradients are identically zero: $\lambda_1 = \lambda_2 = 0$
2. If the region contains and ideal step edge, then there is only one gradient direction: $\lambda_1 > 0, \lambda_2 = 0$
3. If the region contains two orientations, then there are two gradient directions: $\lambda_1 \geq \lambda_2 > 0$



Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive (or perhaps 0 for degeneracies).
- Consider three cases

1. If the region of interest is perfectly uniform, then the gradients are identically zero: $\lambda_1 = \lambda_2 = 0$
2. If the region contains and ideal step edge, then there is only one gradient direction: $\lambda_1 > 0, \lambda_2 = 0$
3. If the region contains two orientations, then there are two gradient directions: $\lambda_1 \geq \lambda_2 > 0$

Corner detection: Differential analysis

Eigenvalues

- Because the matrix

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

is symmetric, we can diagonalize it by a rotation of the coordinates axes to yield a form

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

where λ_1, λ_2 are the eigenvalues of the matrix.

- Remark: Owing to the fact that our matrix is positive definite (e.g., the determinant is >0), we are guaranteed that both of the eigenvalues are positive.
- Consider three cases

1. If the region of interest is perfectly uniform, then the gradients are identically zero: $\lambda_1 = \lambda_2 = 0$
2. If the region contains and ideal step edge, then there is only one gradient direction: $\lambda_1 > 0, \lambda_2 = 0$
3. If the region contains two orientations, then there are two gradient directions: $\lambda_1 \geq \lambda_2 > 0$

- In conclusion
  - The eigenvalues capture edge strength.
  - The eigenvectors capture edge direction.

Corner detection: Differential analysis

What we have learned

- For the summed gradient matrix, **C**
  - The eigenvectors capture edge direction.
  - The eigenvalues capture edge strength.

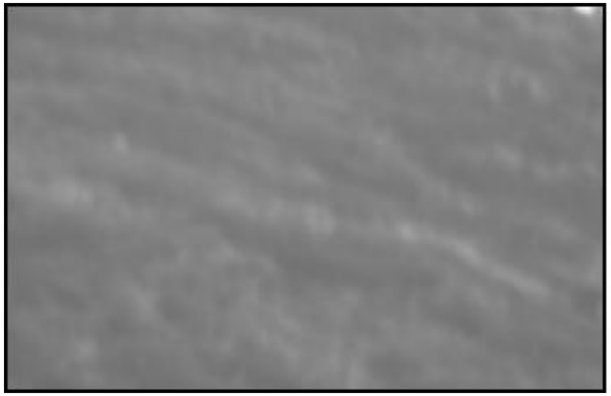
Resulting approach

- Detection of corners and lines
  - For each point in an image of interest
  - Construct the 2x2 summed image gradient matrix, **C**
  - Calculate the eigenvalues of **C**
  - If the eigenvalues are similar (nonzero) magnitude, then a corner is marked.
  - Also, when only one eigenvalues is nonzero, then a line can be marked.

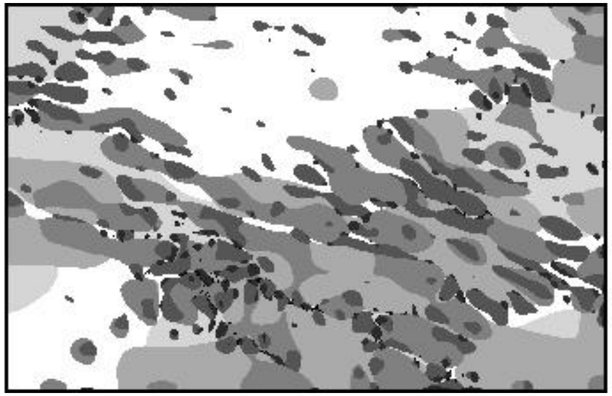
Corner detection: Exploiting local orientation estimates

An alternative image measurement

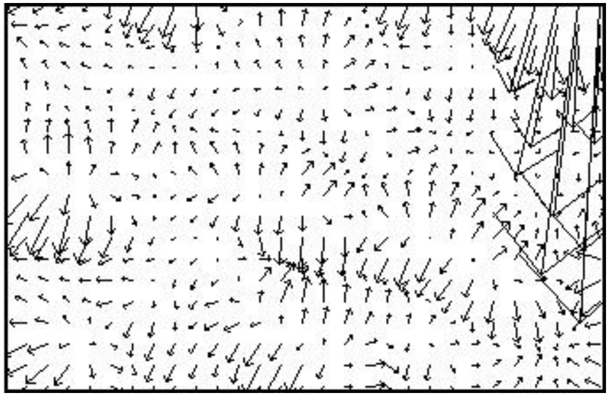
- Previously, we developed the ability to decompose images according to their local orientation structure
  - E.g., via convolution with Gabor filters
- As an application, we noted the abililty to determine the locally dominant orientation and its magnitude
  - E.g., by scanning across the oriented bandpass decomposition for (locally) largest magnitudes



Source image
(natural terrain)



Locally dominant scale
(darker intensity for finer scale)



Locally dominant orientation
(shown as normal vector)

Corner detection: Exploiting local orientation estimates

An alternative image measurement

- Previously, we developed the ability to decompose images according to their local orientation structure
  - E.g., via convolution with Gabor filters
- As an application, we noted the abililty to determine the locally dominant orientation and its magnitude
  - E.g., by scanning across the oriented bandpass decomposition for (locally) largest magnitudes
- We can exploit that analysis in the present context
  - Rather than construct our corner (and line) detection matrix with image gradients
  - Use the locally dominant orientation magnitudes as projected on the coordinate axes.
- Let
  - The dominant orientation be recovered as $(\cos w, \sin w)$
  - The corresponding response magnitude be given as r
  - Then replace Ex with $r\,(\cos w, \sin w).(1,0) = r \cos w$
  - And replace Ey with $r\,(\cos w, \sin w).(0,1) = r \sin w$
- This formulation has the advantage of uniformity of representation across levels of our system.

Corner detection: Exploiting local orientation estimates

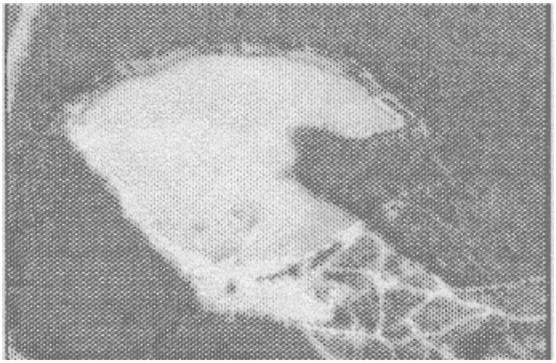
An alternative image measurement

- Previously, we developed the ability to decompose images according to their local orientation structure
  - E.g., via convolution with Gabor filters
- As an application, we noted the abililty to determine the locally dominant orientation and its magnitude
  - E.g., by scanning across the oriented bandpass decomposition for (locally) largest magnitudes
- We can exploit that analysis in the present context
  - Rather than construct our corner (and line) detection matrix with image gradients
  - Use the locally dominant orientation magnitudes as projected on the coordinate axes.
- Let
  - The dominant orientation be recovered as $(\cos w, \sin w)$
  - The corresponding response magnitude be given as r
  - Then replace Ex with r $(\cos w, \sin w).(1,0) = r \cos w$
  - And replace Ey with r $(\cos w, \sin w).(0,1) = r \sin w$
- This formulation has the advantage of uniformity of representation across levels of our system.

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

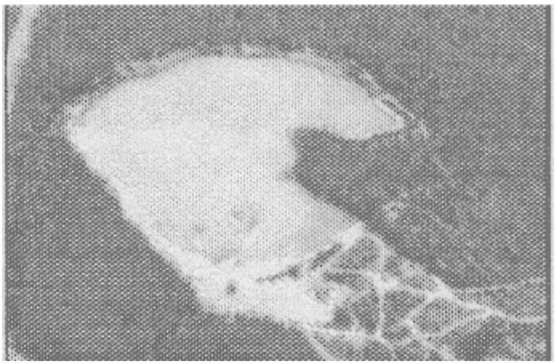
$$\begin{pmatrix} \sum r^2 \cos^2 w & \sum r^2 \cos w \sin w \\ \sum r^2 \cos w \sin w & \sum r^2 \sin^2 w \end{pmatrix}$$

Corner/line detection: Example



Source image

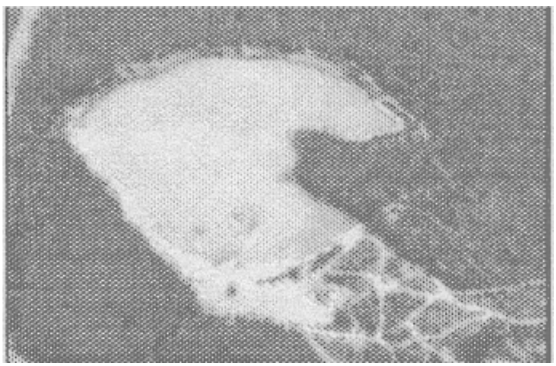
Corner/line detection: Example



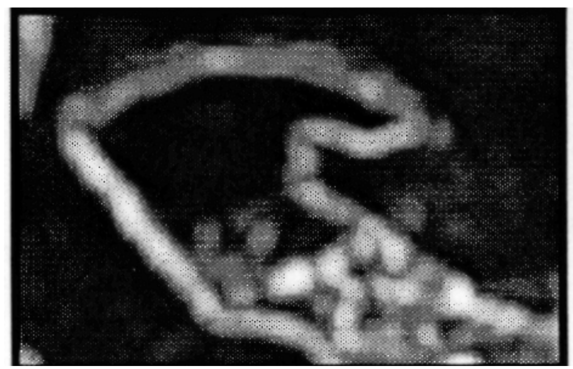
Source image

Filtered response magnitude at 4 orientations

Corner/line detection: Example



Source image



Detected lines

Detected corners

Filtered response magnitude at 4 orientations

Corner detection: Recapitulation

Model

- Image loci where multiple orientations are present.
- Local orientation structure captured by summed gradient matrix.
- Approach also captures line structure.

3 step process

- Recover local estimates of image orientation structure: direction and magnitude
- Accumulate local measures of orientation structure into summed gradient matrix.
- Perform eigenvalue decomposition.

Case studies

- Image gradient based measurements
- More general oriented filtering based measurements
- Natural image example.

Outline

- Introduction
- Edge detection
- Corner detection
- **Hough transform**
- Deformable templates

Detecting contour features: Beyond simplest features

Motivation

- So far, the features of interest (edges, corners) have been defined on a purely local basis.
- Now consider configurations of contours that correspond to more complicated geometries
  - Extended lines
  - Circles
  - Simply parameterized objects
- Two types of approach covered

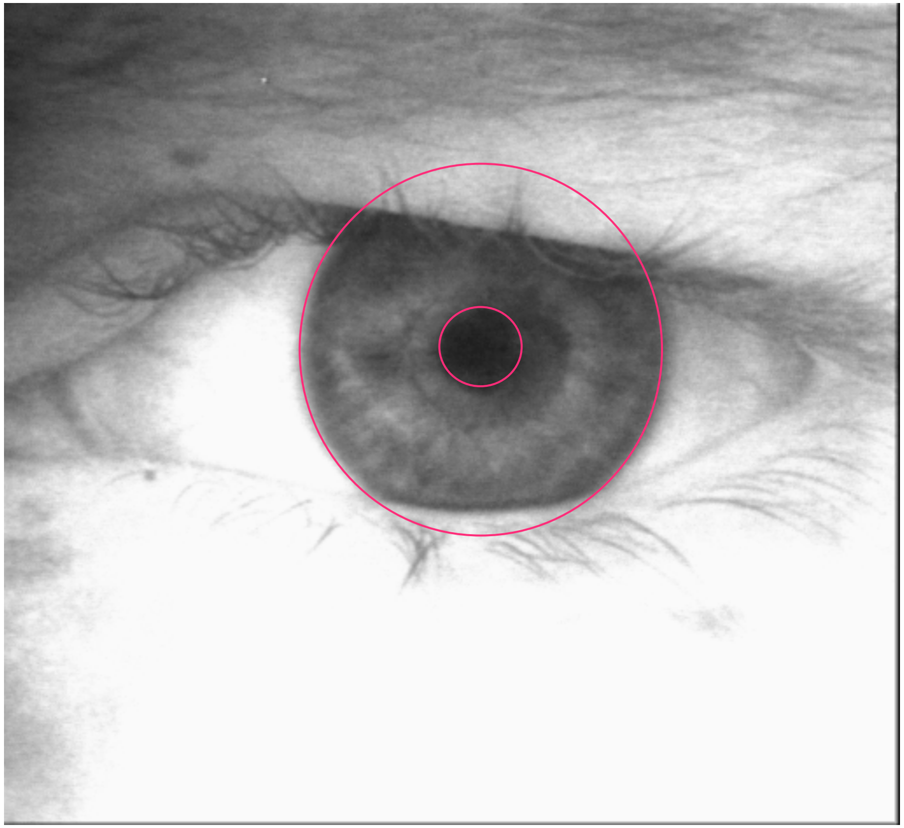
1. First extract edge features; then fit the model.
2. Fit the model more directly to an (enhanced) image.

Detecting contour features: Beyond simplest features

Motivation

- So far, the features of interest (edges, corners) have been defined on a purely local basis.
- Now consider configurations of contours that correspond to more complicated geometries
  - Extended lines
  - Circles
  - Simply parameterized objects
- Two types of approach covered

1. First extract edge features; then fit the model.
2. Fit the model more directly to an (enhanced) image.



Hough transform: Introduction

Basic idea

- The Hough transform was introduced to detect patterns of points in binary images.
- It thus corresponds to the class of techniques that assume edge (or some other primitive detection) already has marked points of interest in an image.
- The key idea:
  - transform a potentially difficult problem: Detection of a relatively complex pattern in the image domain
  - into a simpler problem of peak detection in the space of the pattern's parameters

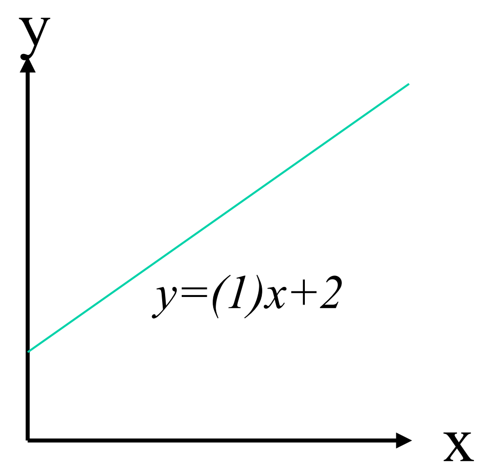
Hough transform: Introduction

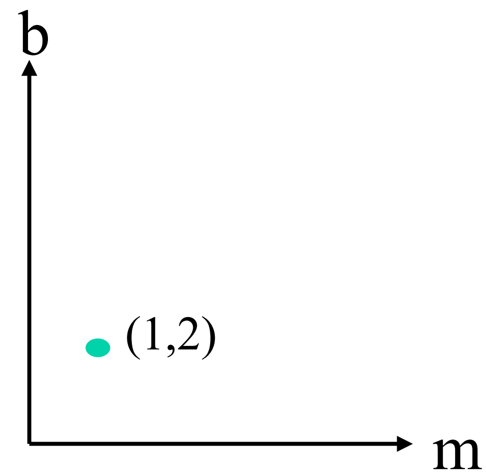
Basic idea

- The Hough transform was introduced to detect patterns of points in binary images.
- It thus corresponds to the class of techniques that assume edge (or some other primitive detection) already has marked points of interest in an image.
- The key idea:
  - transform a potentially difficult problem: Detection of a relatively complex pattern in the image domain
  - into a simpler problem of peak detection in the space of the pattern’s parameters

Example

- Detection of lines
  - Suppose we represent a line as $y = mx + b$
  - Move from the space of image position (x,y)
  - To the space of line parameters (m,b).



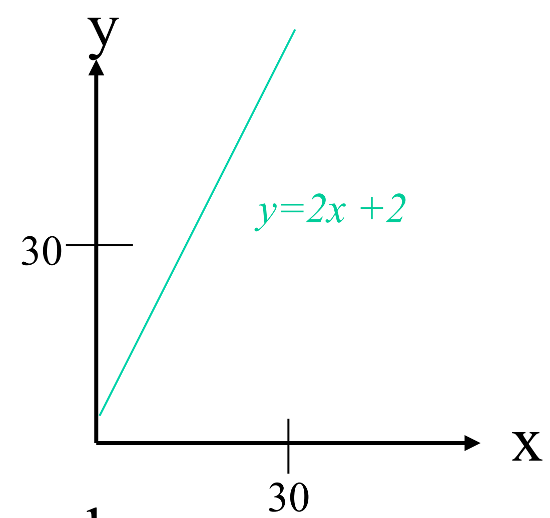


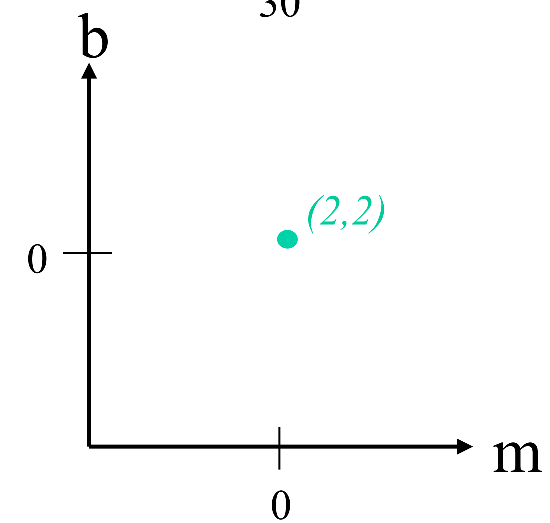
Hough transform: Line detection

A closer look (in 2 parts)

1. Transform line detection into line intersection

- Any line $y = mx + b$ is uniquely identified by a parameter pair (m,b).
- The line is represented by a point in the (m,b) plane (parameter space).



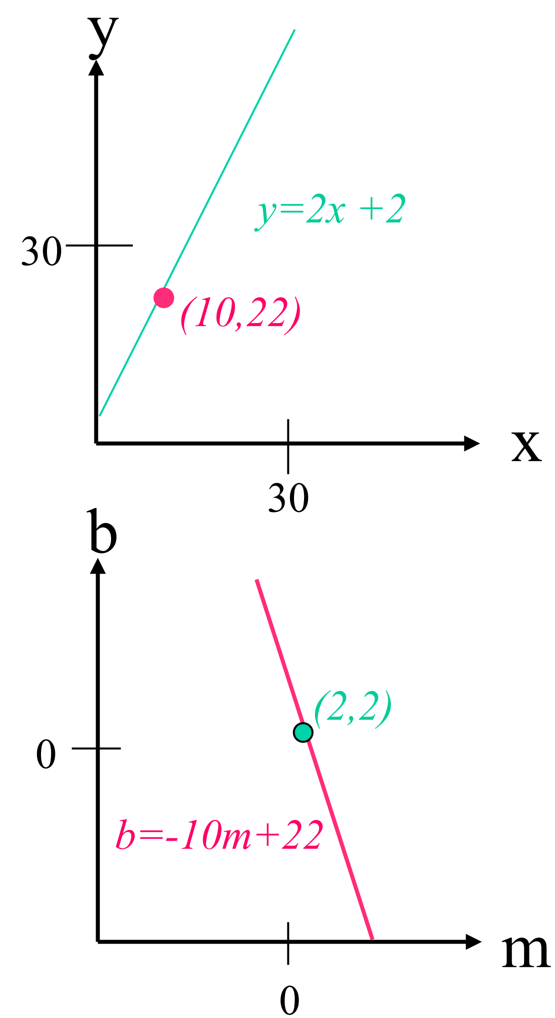


Hough transform: Line detection

A closer look (in 2 parts)

1. Transform line detection into line intersection

- Any line $y = mx + b$ is uniquely identified by a parameter pair (m,b).
- The line is represented by a point in the (m,b) plane (parameter space).
- Any point (x,y) in the image corresponds to a line $b = x(-m) + y$ in parameter space

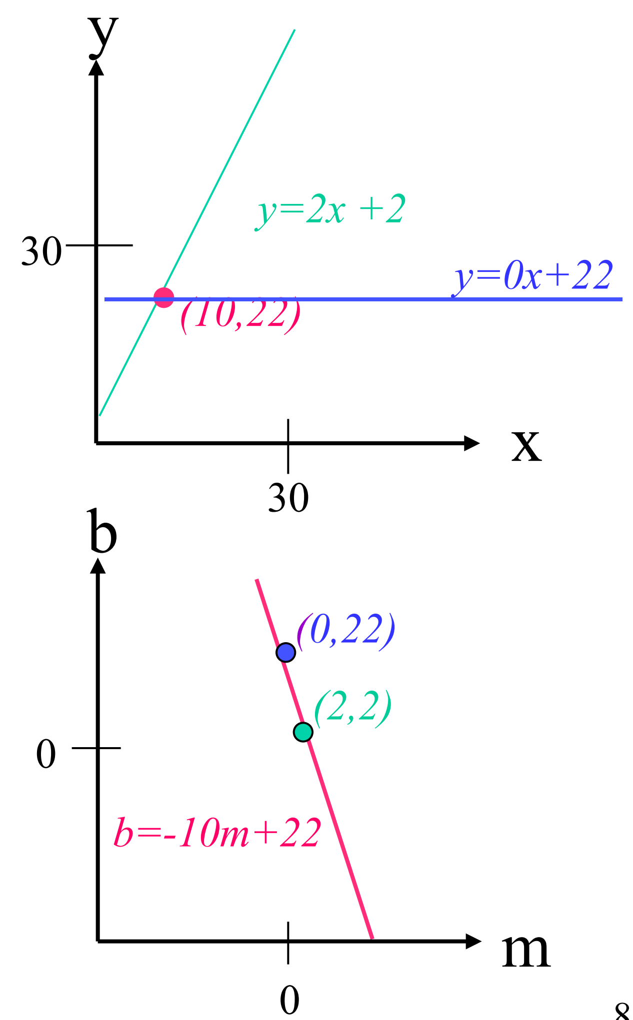


Hough transform: Line detection

A closer look (in 2 parts)

1. Transform line detection into line intersection

- Any line $y = mx + b$ is uniquely identified by a parameter pair (m,b).
- The line is represented by a point in the (m,b) plane (parameter space).
- Any point (x,y) in the image corresponds to a line $b = x(-m) + y$ in parameter space
  - As m and b vary, this captures all line through (x,y).

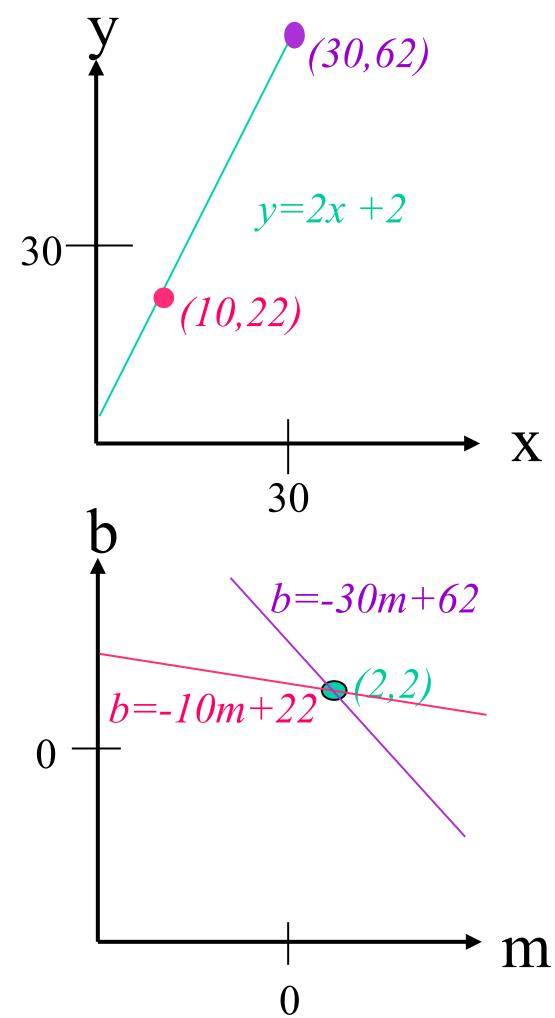


Hough transform: Line detection

A closer look (in 2 parts)

1. Transform line detection into line intersection

- Any line $y = mx + b$ is uniquely identified by a parameter pair (m,b).
- The line is represented by a point in the (m,b) plane (parameter space).
- Any point (x,y) in the image corresponds to a line $b = x(-m) + y$ in parameter space
  - As m and b vary, this captures all line through (x,y).
- So, a line defined by N collinear image points is identified in parameter space by the intersection of the N associated lines in parameter space.

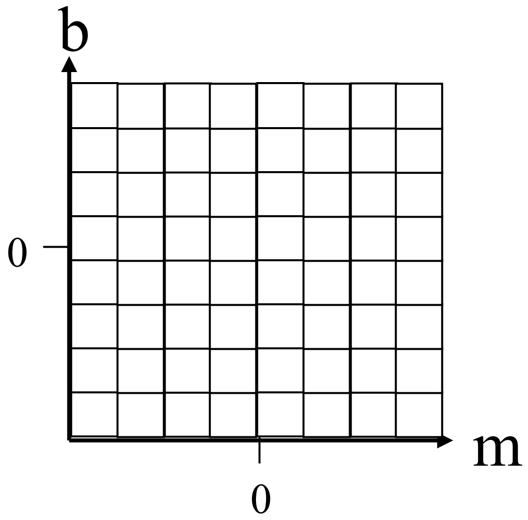


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the *(m,b)*-plane into a finite grid of cells.
- Associate a counter *c(m,b),* with each cell; initialize it to *0*.

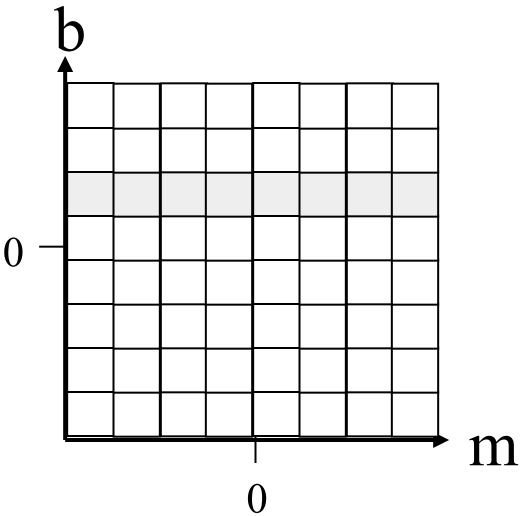


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the *(m,b)*-plane into a finite grid of cells.
- Associate a counter *c(m,b),* with each cell; initialize it to *0.*
- For each image point *p*, increment all counters on the corresponding line in parameter space.

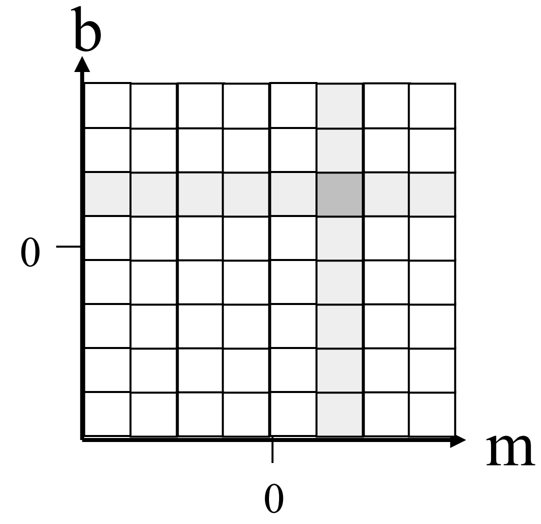


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the (m,b)-plane into a finite grid of cells.
- Associate a counter $c(m,b)$, with each cell; initialize it to 0.
- For each image point p, increment all counters on the corresponding line in parameter space.

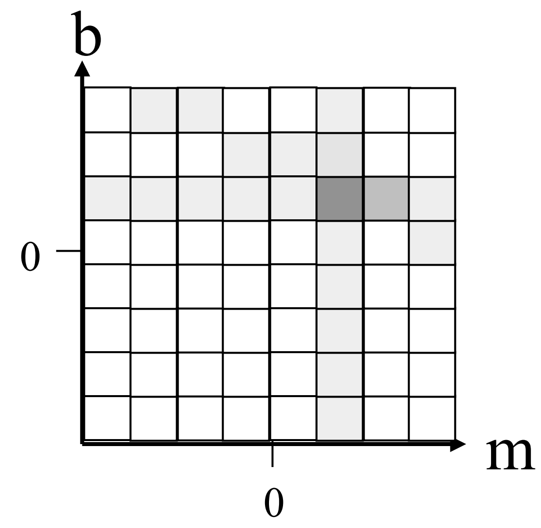


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the (m,b)-plane into a finite grid of cells.
- Associate a counter $c(m,b)$, with each cell; initialize it to 0.
- For each image point p, increment all counters on the corresponding line in parameter space.

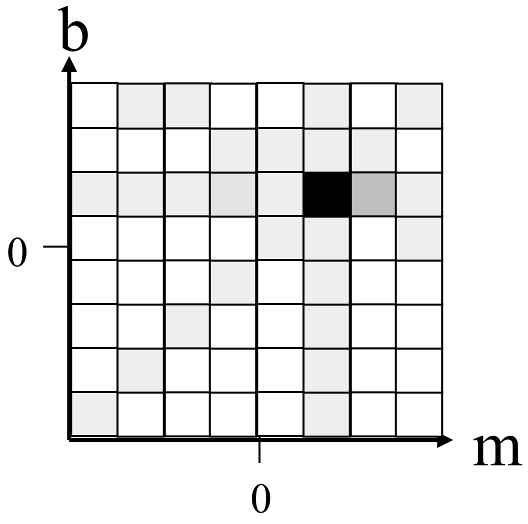


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the *(m,b)*-plane into a finite grid of cells.
- Associate a counter *c(m,b),* with each cell; initialize it to *0*.
- For each image point *p*, increment all counters on the corresponding line in parameter space.
- Note that for *N* image points, the corresponding N lines in parameter space must go through the "true" value of *(m,b)*.
- So, the line is identified with the parameters corresponding to the largest count, *c(m,b)*.

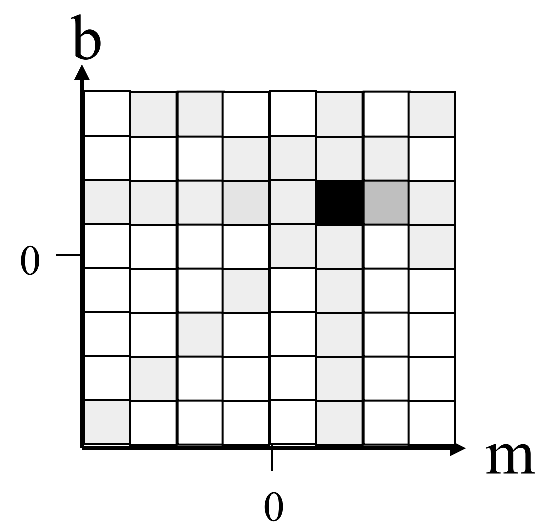


Hough transform: Line detection

A closer look (in 2 parts)

2. Transform line intersection into peak detection

- Divide the (m,b)-plane into a finite grid of cells.
- Associate a counter $c(m,b)$, with each cell; initialize it to 0.
- For each image point p, increment all counters on the corresponding line in parameter space.
- Note that for N image points, the corresponding N lines in parameter space must go through the "true" value of (m,b).
- So, the line is identified with the parameters corresponding to the largest count, $c(m,b)$.



Recap: The 2 parts

1. Transform line detection into line intersection
2. Transform line intersection into peak detection

Hough transform: Line detection

A few practical considerations

- In theory, the parameter values can take on any real value.
  - Must discretize while weighting precision vs. storage/processing requirements.
- Real images will contain pixels "incorrectly" marked as edges due to noise
  - Must select a threshold on a minimally acceptable value for $C(m,b)$.
- There may be multiple lines present in an image
  - The Hough can simultaneously detect all of these by returning all (m,b) pairs whose counter exceeds the threshold.

Hough transform: Line detection

Procedure

- Input: Binary image, $I(i,j)$ pixels marked 1 of edge has been detection; else 0.
- Output: (m,b) detected line parameters.

1. Discretize the parameter space (m,b) using sampling intervals dm, db, which yield precision suited to application, yet reasonable storage requirements; let the resulting number of values for b and m be B and M, respectively.
2. Let $C(m,b)$ be an integer array of counters; initialize all elements to 0.
3. For each pixel $I(i,j)=1$,

   For each $m=1..M$

   i) Let $b' = j - m\,i$

   ii) Find the index b closest to b'.

   iii) Increment $C(m,b)$ by 1.
4. Return all (m,b) where $C(m,b) > t$; t a threshold value.

Hough transform: Generalization

The generalized Hough transform

- We start by assuming that we are given a set of edge points $(x_i, y_i), i = 1..n$
- Suppose we represent a contour geometry of interest as a parametric expression of the form

$$g(x_i, y_i, \mathrm{p}) = 0$$

with $\mathbf{p}$ a parameter vector.

Hough transform: Generalization

The generalized Hough transform

- We start by assuming that we are given a set of edge points $(x_i, y_i), i = 1..n$
- Suppose we represent a contour geometry of interest as a parametric expression of the form

$$g(x_i, y_i, \mathrm{p}) = 0$$

e.g., for line: *$mx+b-y=0$*

with $\mathbf{p}$ a parameter vector.

Hough transform: Generalization

The generalized Hough transform

- We start by assuming that we are given a set of edge points $(x_i, y_i), i = 1..n$
- Suppose we represent a contour geometry of interest as a parametric expression of the form

$$g(x_i, y_i, \mathrm{p}) = 0$$

  with $\mathbf{p}$ a parameter vector.

- Let the characteristic function of g be

$$h(x_i, y_i, \mathbf{p}) = \begin{Bmatrix} 1, g(x_i, y_i, \mathbf{p}) = 0 \\ 0, \text{otherwise} \end{Bmatrix}$$

Hough transform: Generalization

The generalized Hough transform

- We start by assuming that we are given a set of edge points $(x_i, y_i), i = 1..n$
- Suppose we represent a contour geometry of interest as a parametric expression of the form

$$g(x_i, y_i, \mathrm{p}) = 0$$

with $\mathbf{p}$ a parameter vector.

- Let the characteristic function of g be

$$h(x_i, y_i, \mathbf{p}) = \begin{Bmatrix} 1, g(x_i, y_i, \mathbf{p}) = 0 \\ 0, \text{otherwise} \end{Bmatrix}$$

- Then we define the (generalized) Hough transform as

$$H(\mathbf{p}) = \sum_{i=1}^{n} h(x_i, y_i, \mathbf{p})$$

Hough transform: Generalization

The generalized Hough transform

- We start by assuming that we are given a set of edge points $(x_i, y_i), i = 1..n$
- Suppose we represent a contour geometry of interest as a parametric expression of the form

$$g(x_i, y_i, \mathrm{p}) = 0$$

with $\mathbf{p}$ a parameter vector.

- Let

$$h(x_i, y_i, \mathbf{p}) = \begin{cases} 1, g(x_i, y_i, \mathbf{p}) = 0 \\ 0, \text{otherwise} \end{cases}$$

- Then we define the (generalized) Hough transform as

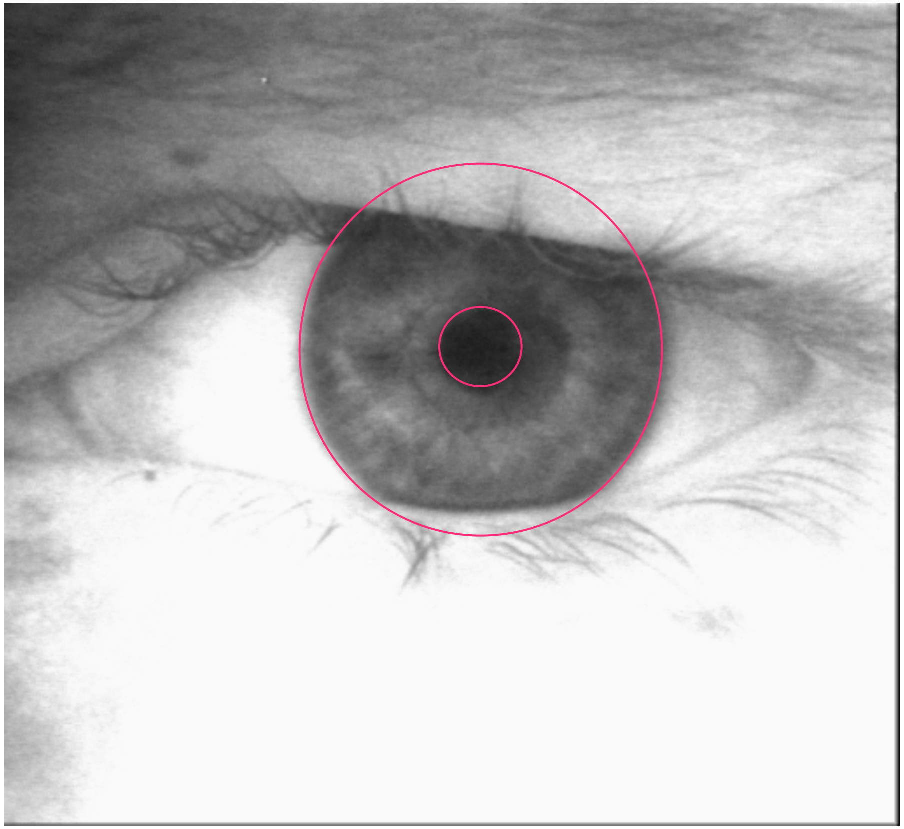
$$H(\mathbf{p}) = \sum_{i=1}^{n} h(x_i, y_i, \mathbf{p})$$

Example

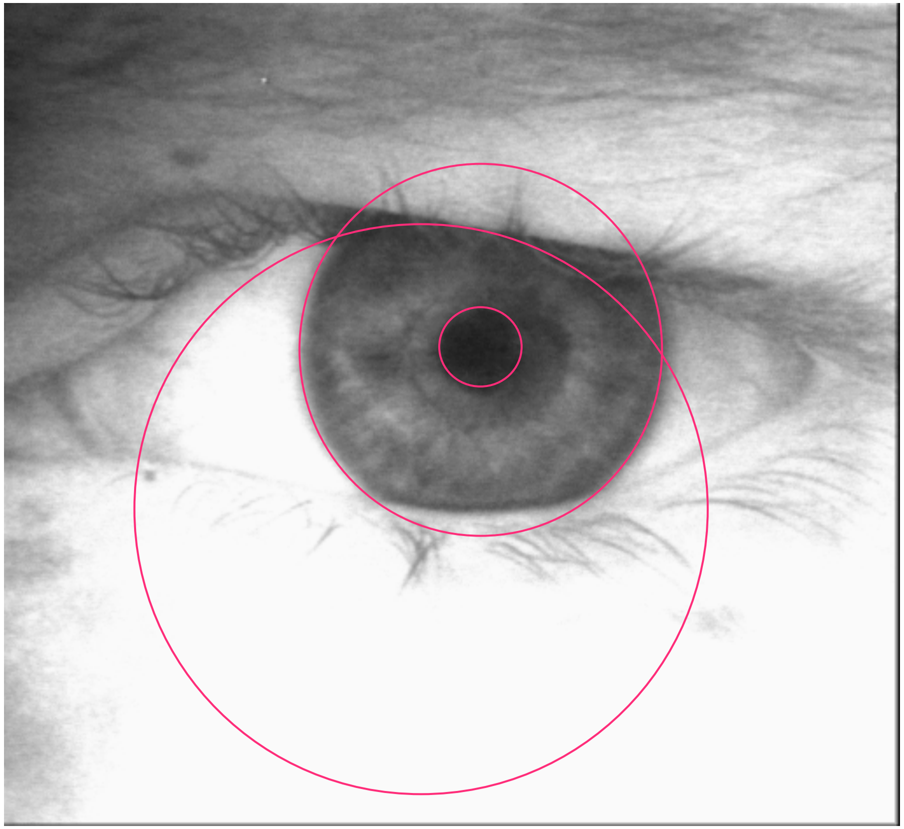
- For a circle, we might let $\mathbf{p}^{\mathrm{T}} = (x_c, y_c, r)$

$$g(x_i, y_i, x_c, y_c, r) = (x_i - x_c)^2 + (y_i - y_c)^2 - r^2$$

Hough transform: Examples



Hough transform: Examples



Hough transform: Remarks

Pluses

- The Hough can convert difficult model fitting problems into a simple histogramming operation.
- For a given image, it allows for simultaneous detection of multiple instances of a model.
- It is robust, to outliers in the data (marked edge pixels).

Hough transform: Remarks

Pluses

- The Hough can convert difficult model fitting problems into a simple histogramming operation.
- For a given image, it allows for simultaneous detection of multiple instances of a model.
- It is robust, to outliers in the data (marked edge pixels).

Minuses

- The required space can be large for even a moderate number of parameters, if precision is required.
- The time required in peak search grows rapidly with the number of parameters.
- In depending on detected edge pixels, it has no recourse to the original image data.

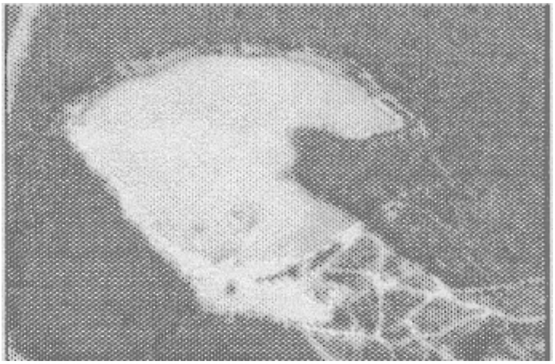
Outline

- Introduction
- Edge detection
- Corner detection
- Hough transform
- **Deformable templates**

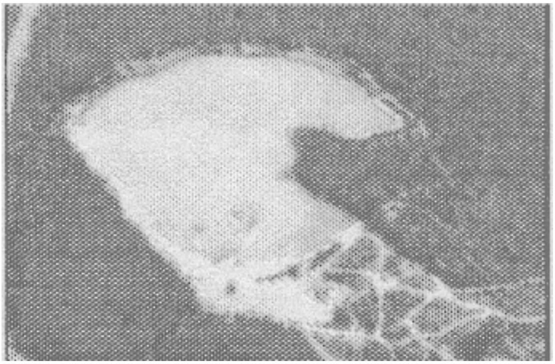
Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.



Deformable templates: Basic idea



Motivation

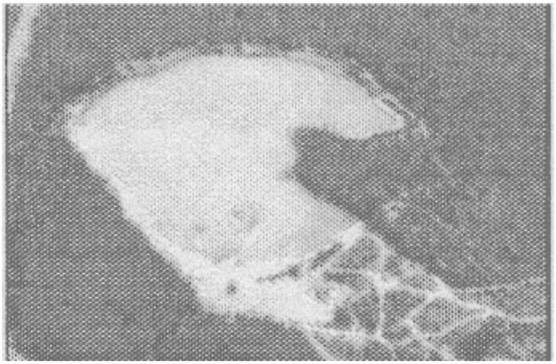
- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

1. Enhance the image to make those portions most likely to be important stand out.
    - e.g., high contrast contours, i.e., strong light/dark transitions

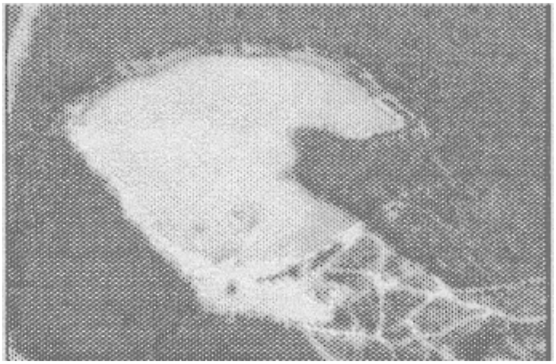


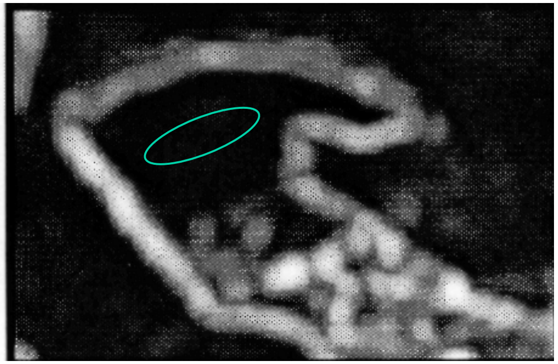
Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

1. Enhance the image to make those portions most likely to be important stand out.
    - e.g., high contrast contours, i.e., strong light/dark transitions.
2. Initialize and then incrementally adjust the parameters of the model so that the shape “deforms” so as to lie along the most enhanced pixels.



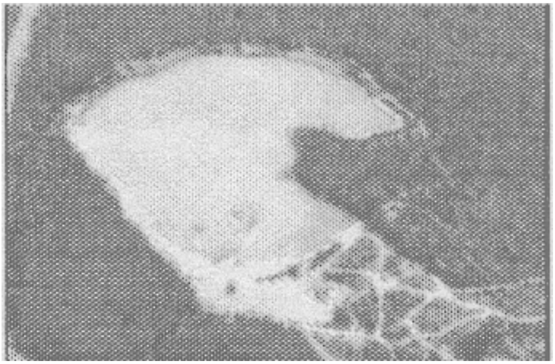


Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

1. Enhance the image to make those portions most likely to be important stand out.
    - e.g., high contrast contours, i.e., strong light/dark transitions
2. Initialize and then incrementally adjust the parameters of the model so that the shape “deforms” so as to lie along the most enhanced pixels.

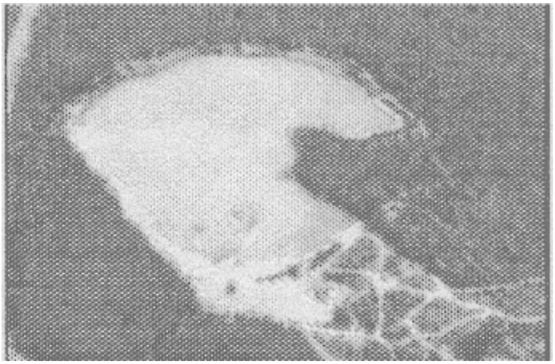


Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

1. Enhance the image to make those portions most likely to be important stand out.
   - e.g., high contrast contours (i.e., strong light/dark transitions)
2. Initialize and then incrementally adjust the parameters of the model so that the shape “deforms” so as to lie along the most enhanced pixels.

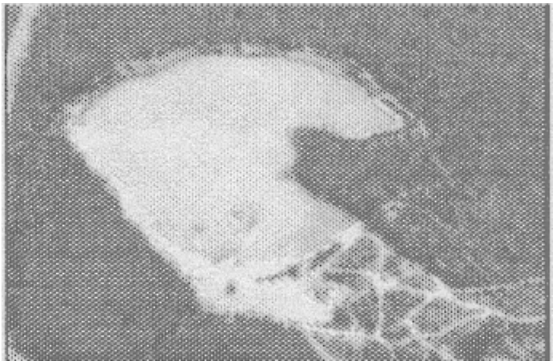


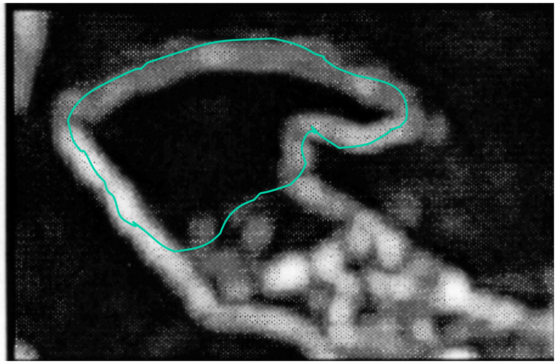
Deformable templates: Basic idea

Motivation

- Recent thinking has questioned the approach of fitting contour models in two discrete stages (first extract generic interesting points, then fit the model).
- At issue is the fact that much of the image data is discarded without knowing exactly what it will be used for.
- Instead, consider fitting a parameterized model more directly to the image data.
- Again, a two step process results, but without a hard up front decision about which parts of the data are relevant

1. Enhance the image to make those portions most likely to be important stand out.
   - e.g., high contrast contours (i.e, strong light/transitions)
2. Initialize and then incrementally adjust the parameters of the model so that the shape "deforms" so as to lie along the most enhanced pixels.





Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

  with s varying from 0 to 1.

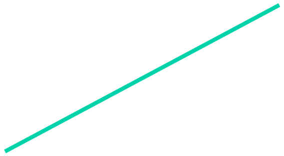
- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$
$$y(s) = ds^2 + es + f$$



with s varying from 0 to 1.

- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$
$$y(s) = ds^2 + es + f$$

with s varying from 0 to 1.

- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$
$$y(s) = ds^2 + es + f$$

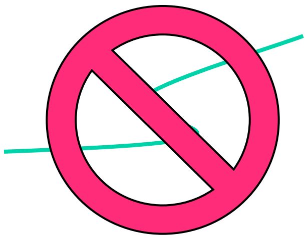
  with s varying from 0 to 1.
- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$
$$y(s) = ds^2 + es + f$$



with s varying from 0 to 1.

- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

with s varying from 0 to 1.

- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Deformable templates: Formalization

Template model

- For illustration, we consider a simple contour model, that of a parabolic segment.

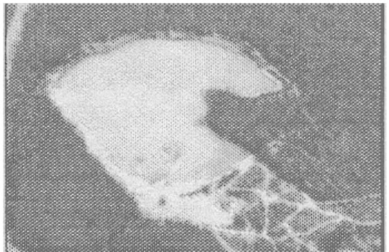
$$x(s) = as^2 + bs + c$$

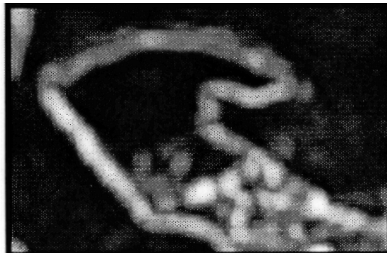
$$y(s) = ds^2 + es + f$$

  with s varying from 0 to 1.
- This contour model varies in shape (deforms) as we vary the parameters a, b, c, d, e, f.
- We will seek to vary the parameters so that the final shape lies along high contrast image points.

Energy image

- We seek to enhance those portions of the image that have high contrast contours.
- Various approaches could be considered (e.g., perhaps most simply, image gradient magnitude).
- We choose to use an image derived by taking the magnitude of the strongest response in our oriented bandpass image decomposition (as derived from Gabor filters).
- We refer to this image, I, as the energy image.
  - It has most energy along high contrast contours.





Deformable templates: Formalization

Fitting the model

- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.

Deformable templates: Formalization

Fitting the model

- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x,y)\,ds \right]$$

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

Deformable templates: Formalization

Fitting the model

- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x,y)\,ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent

Deformable templates: Formalization

Fitting the model

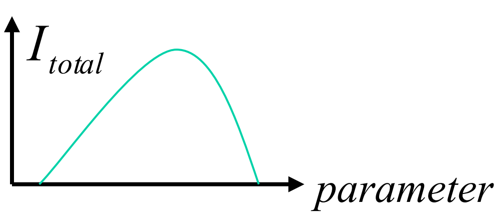
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x,y)\,ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

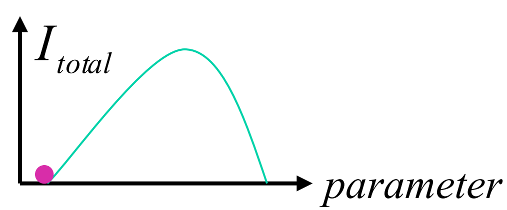
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

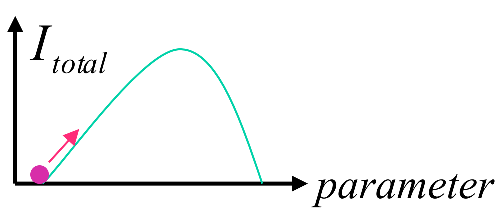
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

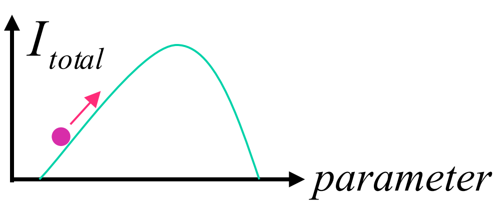
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x,y)\,ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

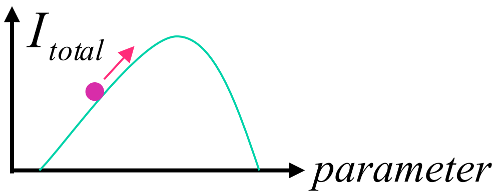
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

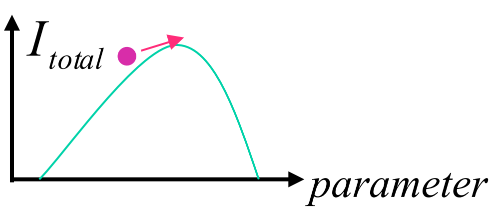
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x,y)\,ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model

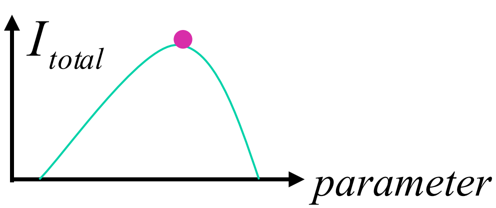
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- We seek to adjust automatically the template parameters so that it lies along the largest values in the energy.
- So, we seek to maximize the total of the energy image values that lie along the contour defined by the template
  - Let

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

  - We seek

$$\max_{a,b,c,d,e,f} I_{total}$$

- To solve, we adjust the parameter values iteratively, by moving in parameter space along the gradient direction until a local maximum is reached.
  - A procedure known as gradient ascent



Deformable templates: Formalization

Fitting the model (cont.)

- For our specific template parameterization

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

at each iteration we increment, e.g., a according to

$$a^{new} = a^{old} + \Delta a; \qquad \Delta a = \frac{\partial I_{total}}{\partial a}$$

Deformable templates: Formalization

Fitting the model (cont.)

- For our specific template parameterization

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

at each iteration we increment, e.g., a according to

$$a^{new} = a^{old} + \Delta a; \qquad \Delta a = \frac{\partial I_{total}}{\partial a}$$

- More specifically, recalling that

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

and applying the chain rule yields

$$\Delta a = \int_0^1 \frac{\partial I}{\partial x} \frac{\partial x}{\partial a} ds$$

or

$$\Delta a = \int_0^1 \frac{\partial I}{\partial x} s^2 ds$$

Deformable templates: Formalization

Fitting the model (cont.)

- For our specific template parameterization

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

at each iteration we increment, e.g., b according to

$$b^{new} = b^{old} + \Delta b; \qquad \Delta b = \frac{\partial I_{total}}{\partial b}$$

- More specifically, recalling that

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

and applying the chain rule yields

$$\Delta b = \int_0^1 \frac{\partial I}{\partial x} \frac{\partial x}{\partial b} ds$$

or

$$\Delta b = \int_0^1 \frac{\partial I}{\partial x} s \; ds$$

Deformable templates: Formalization

Fitting the model (cont.)

- For our specific template parameterization

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

at each iteration we increment, e.g., c according to

$$c^{new} = c^{old} + \Delta c; \qquad \Delta c = \frac{\partial I_{total}}{\partial c}$$

- More specifically, recalling that

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

and applying the chain rule yields

$$\Delta c = \int_0^1 \frac{\partial I}{\partial x} \frac{\partial x}{\partial c} ds$$

or

$$\Delta c = \int_0^1 \frac{\partial I}{\partial x} \, ds$$

Deformable templates: Formalization

Fitting the model (cont.)

- For our specific template parameterization

$$x(s) = as^2 + bs + c$$

$$y(s) = ds^2 + es + f$$

at each iteration we increment, e.g., c according to

$$c^{new} = c^{old} + \Delta c; \qquad \Delta c = \frac{\partial I_{total}}{\partial c}$$

- More specifically, recalling that

$$I_{total} = \left[\int_0^1 I(x, y) ds \right]$$

and applying the chain rule yields

$$\Delta c = \int_0^1 \frac{\partial I}{\partial x} \frac{\partial x}{\partial c} ds$$

or

$$\Delta c = \int_0^1 \frac{\partial I}{\partial x} \, ds$$

And similarly for d, e and f.

Deformable templates: Key components

Template model

- Parameterized contour model.
- For example, parabolic segment.

Image representation

- Energy image that enhances high contrast (strong light/dark transition) loci in the image
- For example, maximal magnitude of response in oriented bandpass image representation.

Fitting the model

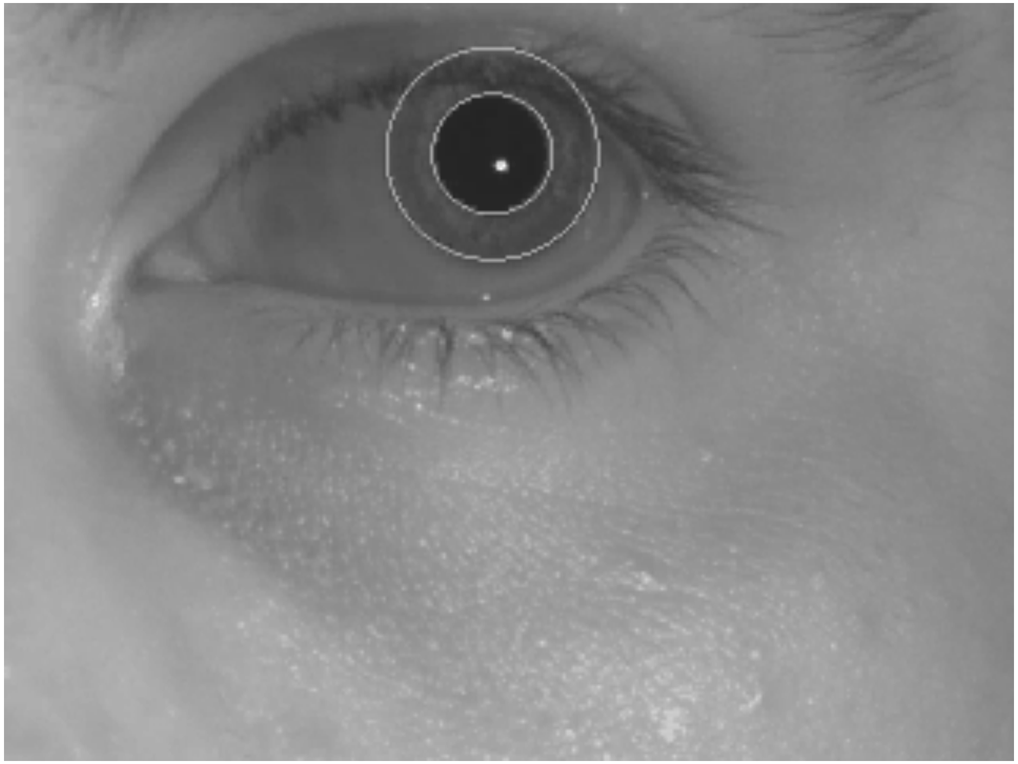
- The model is initialized with a starting set of parameter values, perhaps provided by hand.
- Adjust via gradient ascent the template parameters so that it lies along the largest values in the energy.

Deformable templates: Examples

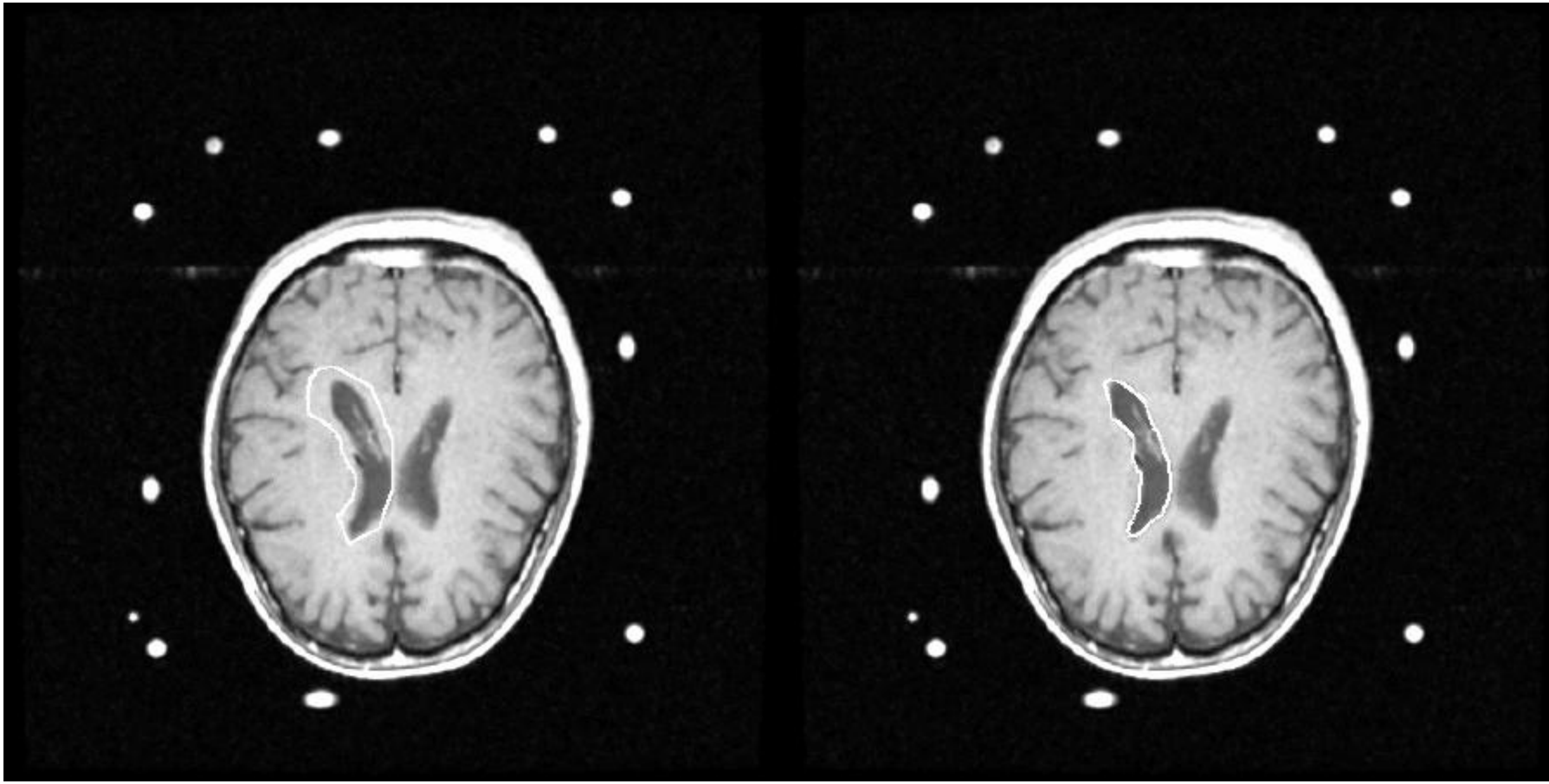
Deformable templates: Examples

Deformable templates: Examples

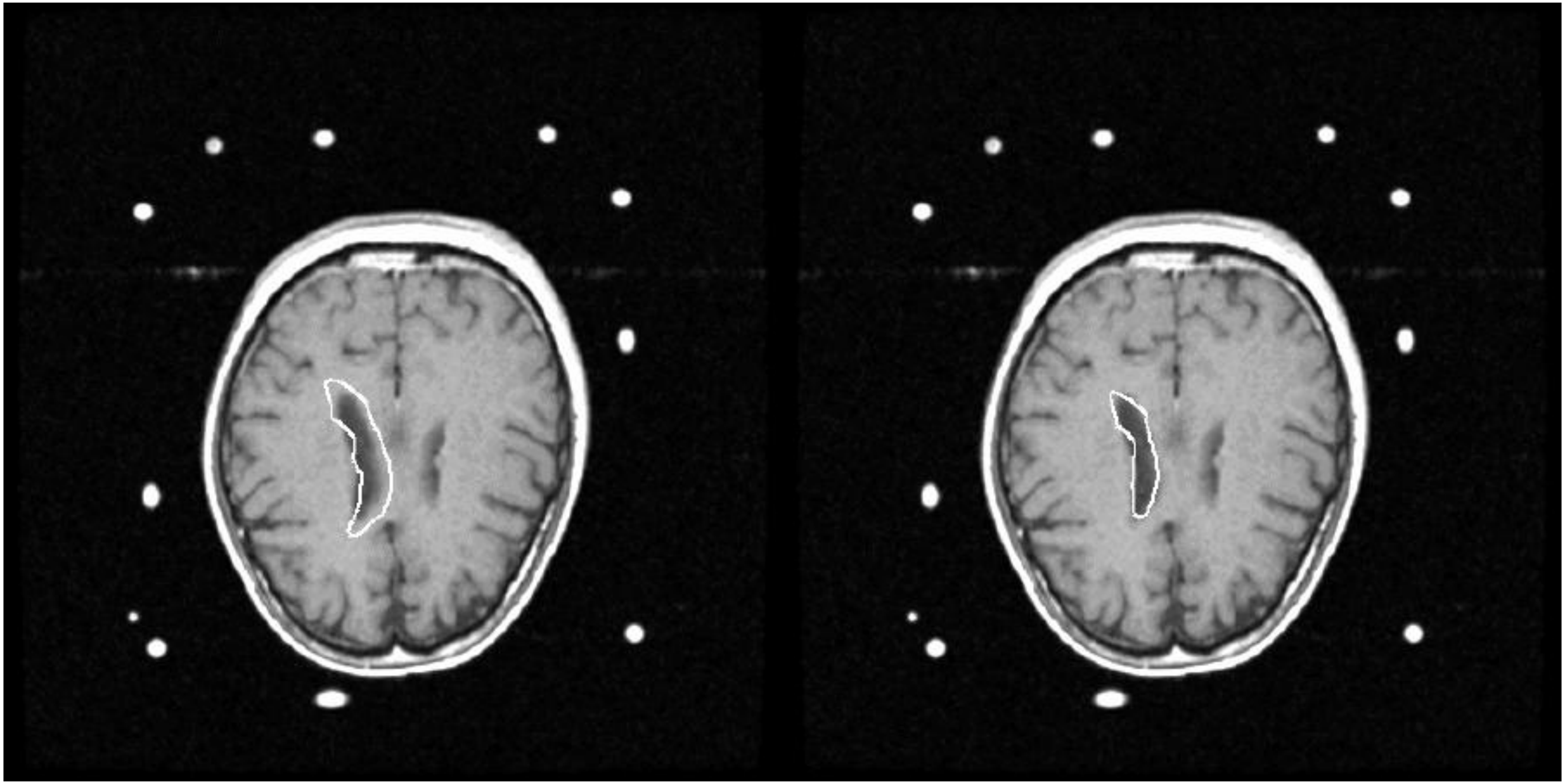
Deformable templates: Examples



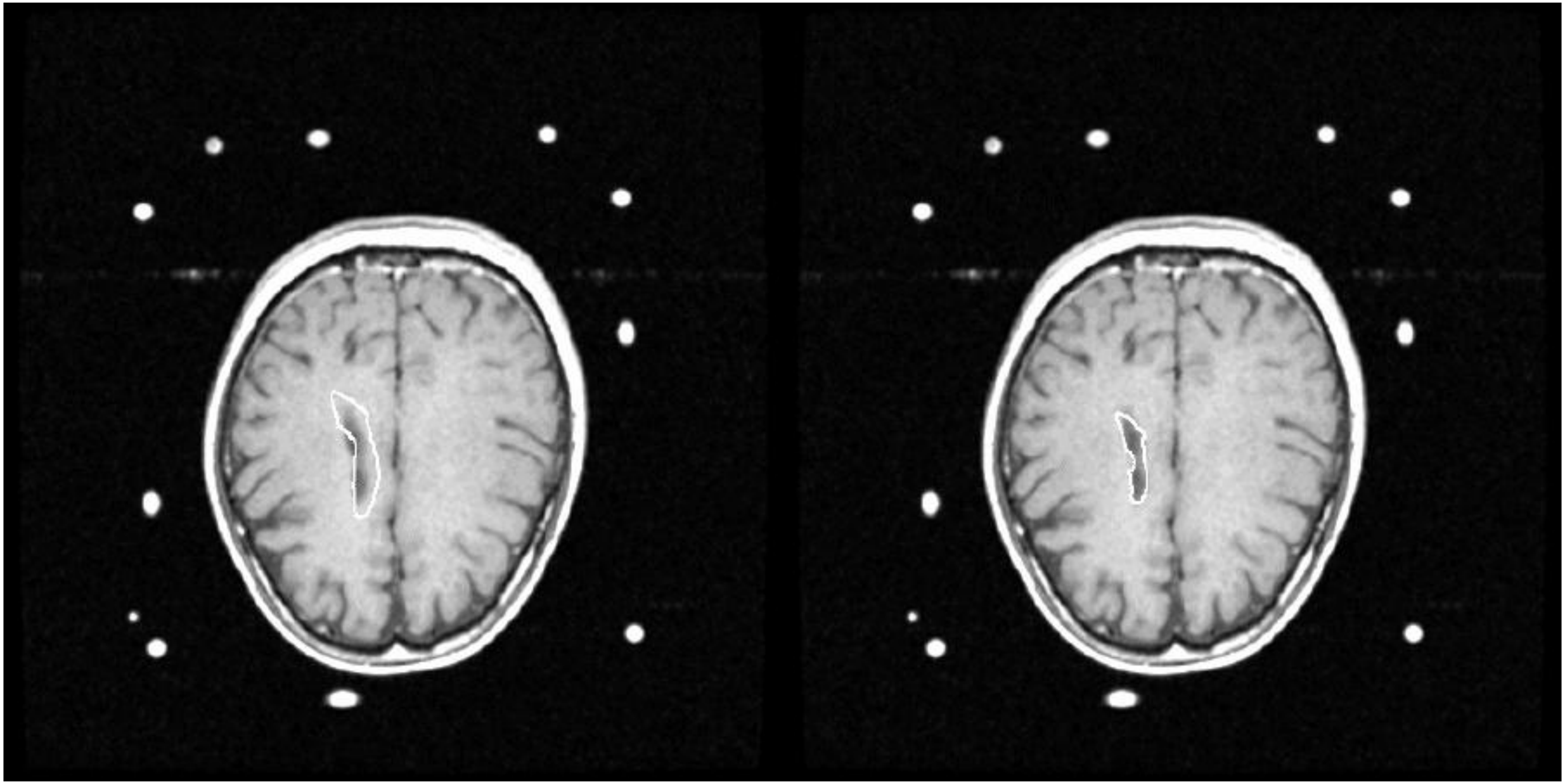
Deformable templates: Examples



Deformable templates: Examples



Deformable templates: Examples



Deformable templates: Remarks

Pluses

- Make use of all the available image data.
- Can accommodate fairly complex contour models.
- Can be intuitive in HCI.

Deformable templates: Remarks

Pluses

- Make use of all the available image data.
- Can accommodate fairly complex contour models.
- Can be intuitive in HCI.

Minuses

- The iterative fit can be slow to converge.
- Prone to local minima.
- Model construction can require careful design.

Summary

- **Introduction**
- **Edge detection**
- **Corner detection**
- **Hough transform**
- **Deformable templates**