

Example



Edge points

Strongest lines



Lane Detection and Departure Warning





Edge Linking Rationale

- Edge maps are still in an *image* format
- * *Image* to *data structure* transform
- Two issues
 - □ *Identity*: there are so many edge points, which ones should be grouped together?
 - Representation: now that a group of edge pixels are identified, how best to represent them?



The Canny edge detector



Problem: pixels along this edge didn't survive the thresholding

thinning (non-maximum suppression)



Hysteresis thresholding

Check that maximum value of gradient value is sufficiently large

drop-outs? use hysteresis

> use a high threshold to start edge curves and a low threshold to continue them.





Hysteresis thresholding



original image



high threshold (strong edges)



low threshold (weak edges)



hysteresis threshold



Object boundaries vs. edges











Background

Texture

Edge detection is just the beginning...

image

human segmentation

gradient magnitude

Source: I ff azebnik



Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Much more on segmentation later in term

Identity

Measurement space clustering

- **u** curve fitting
- □ *global* technique
- Image space grouping
 - □ tracing or following
 - with known templates
 - local technique



Intuition

Q: If several points fall on the same line, what "commonality" is there?



$$(x_o, y_o) \cdot (\cos \theta, \sin \theta) = \rho$$
$$x_o \cos \theta + y_o \sin \theta = \rho$$



Measurement Space Clustering

Example: Hough transform



Duality of Representation

- Image spacea line
 - a point

- Measurement space
 - a point
 - a sinusoidal curve

$$\rho = x_o \cos \theta + y_o \sin \theta$$

= $\sqrt{x_o^2 + y_o^2} (\frac{x_o}{\sqrt{x_o^2 + y_o^2}} \cos \theta + \frac{y_o}{\sqrt{x_o^2 + y_o^2}} \sin \theta)$
= $\sqrt{x_o^2 + y_o^2} \cos(\theta - \alpha)$
where $\alpha = \tan^{-1} \frac{y_o}{x_o}$





A *voting* (evidence accumulation) scheme

- * A point votes for all lines it is on
- All points (on a single line) vote for the single line they are on
- Tolerate a certain degree of occlusion
- * Must know the parametric form



Hough Transform Algorithm

- Select a parametric form
- Quantize measurement space
- For each edge pixel, increment all cells satisfying the parametric form
- Locate maximum in the measurement space

 $\rho - \theta$

 θ : min : 0°, max : 359°, *inc* : 1° ρ : min : 0, max : $N\sqrt{2}$, *inc* : 1 px1

for $\theta = 0$ to 360 inc 1 $\rho = x_o \cos \theta + y_o \sin \theta$ $(\rho, \theta) + +$ end

Example





Accumulator array































General 3D Measurement Space





Hough Transform (cont.)

Theoretically, Hough transform can be constructed for any parametric curve

- a curve with n parameters
- n-dimensional measurement space
- (n-1)-dimensional surfaces for each image point
- □ highly computationally intensive if n>3
- used mainly for lines, circles, ellipses, etc.











(d)



(f)



(a)

(b)









Sometimes edge detectors find the boundary pretty well





Sometimes not well at all





At times we want to find a complete bounding contour of an object:

At other times we want to find an internal or partial contour. E.g., the best path between two points:





acobs



Which of these two paths is better?

How do we decide how good a path is?





Example: edgels to line segments to contours



Original image

Contours derived from edgels





Desired properties of an image contour:

- Contour should be near/on edges
 Strength of gradient
- Contour should be smooth (good continuation)
 - Low curvature



Active Contours (deformable contours, snakes)







- Points, corners, lines, circles, etc., do not characterize well many objects, especially non-man-made ones
- We want other ways to describe and represent objects and image regions: Contour representations
- In particular, *active contours* are contour representations that conform to the (2D) shape by combining geometry and physics to make elastic, deformable shape models
 - These are often used to <u>track</u> contours in time, so the shape deforms to stay with the changing object

Active Contours

- Given an initial contour estimate, find the best match to the image data evolve the contour to fit the object boundary
 - □ This is an optimization problem
 - Often uses dynamic programming, or something similar, in its solution
 - > Iterates until final solution, or until a time limit
 - □ Visual evidence (support) for the contour can come from edges, corners, detected features, or even user input
- Current best contour fit can be the initial estimate for the subsequent frame (e.g., in tracking over time)
- Active contours are particularly useful when dealing with deformable (non-rigid) objects and surfaces

□ These are not easily described by edges, corners, etc.



Active Contours





- Applications:
 - Object segmentation (for object recognition, medical imaging, etc.)
 - Tracking through time
 - Region selection (e.g., in Photoshop) human in the loop



Contour tracking examples

- http://www.youtube.com/watch?v=laiykNbPkgg
- http://www.youtube.com/watch?v=5se69vcbqxA
- http://www.youtube.com/watch?v=ARIZzcE11Es
- http://www.youtube.com/watch?v=OFTDqGLa2p0



Illusory contours


Partial contours

Active contours can deal with occluded or missing image data





Active contours

- Think of an active contour as an elastic band, with an initial default (low energy) shape, that gets pulled or pushed to be near image positions that satisfy various criteria
 - Be near high gradients, detected points, user input, etc.
 - Don't get stretched too much
 - □ Keep a smooth shape
- How is the current contour adjusted to find the new contour at each iteration?
 - Define a cost function ("energy" function) that says how good a possible configuration is.
 - Seek next configuration that minimizes that cost function.



Energy minimization framework

Framework: energy minimization

- Bending and stretching curve = more energy
- Good features = less energy
- Curve evolves to minimize energy

Parametric representation of the curve v(s) = (x(s), y(s))

* Minimize an energy function on v(s)

$$E_{total} = E_{internal} + E_{external} + E_{constraint}$$



Energy minimization framewo

$$E_{total} = E_{internal} + E_{external} + E_{constraint}$$

- A good fit between the current deformable contour and the target shape in the image will yield a low value for this cost (energy) function
 - □ Internal energy: encourage prior shape preferences: e.g., smoothness, elasticity, particular known shape.
 - **External** energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.
 - Constraint energy: allow for specific (often user-specified) constraints that alter the contour locally



Energy minimization

The energy functional typically consists of three terms:

$$\mathcal{E} = \int \left[\mathcal{E}_{int} \left(\mathbf{v}(s) \right) + \mathcal{E}_{img} \left(\mathbf{v}(s) \right) + \mathcal{E}_{con} \left(\mathbf{v}(s) \right) \right] ds$$

$$\int \left\{ \begin{array}{c} \uparrow \\ \text{Total Internal (contour) energy} \\ \uparrow \\ \\ \varepsilon_{int} \left(\mathbf{v}(s) \right) = \left(\alpha(s) \| \mathbf{v}_{s}(s) \|^{2} + \beta(s) \| \mathbf{v}_{ss}(s) \|^{2} \right) / 2 \end{array} \right\} ds$$

$$\mathcal{E}_{int} \left(\mathbf{v}(s) \right) = \left(\alpha(s) \| \mathbf{v}_{s}(s) \|^{2} + \beta(s) \| \mathbf{v}_{ss}(s) \|^{2} \right) / 2$$
Minimize length and curvature of contour

$$\varepsilon_{img} = -w \cdot \left\| \nabla I(x, y) \right\|^2$$

Maximize gradient along contour (Minimize the negative of this)

$$\varepsilon_{con} = \frac{k}{\left\|\mathbf{v} - \mathbf{x}\right\|^2}$$

Negative spring constraint (repulsion)



Examples





Internal model is too "tight"



Examples





Examples







Corpus callosum example





Corpus callosum example





Lips example













Active contours: pros and cons

Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in "subjective" contours
- Flexibility in how energy function is defined, weighted.

Cons:

- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information



Devil in the Details

Snake: an energy minimizing spline

subject to

□ internal forces (*template shape*)

- resisting stretching and compression
 - maintain natural length
- resisting bending
 - maintain natural curvature
- resisting twisting
 - maintain natural torsion (for 3D snake)

external forces (*shape detector*)

> attract a snake to lines, edges, corners, etc.



Physics Law

* A snake's final position and shape influenced by

- □ balance of all applied forces
- □ total potential energy is minimum
- a dynamic sequence is played out which is based on physics principle





$$c^{(0)}(s) = (x^{(0)}(s), y^{(0)}(s))$$

$$s = 0$$

$$s = 1$$

Internal energy

- resisting stretching and compression

$$E_1 = \int (c_s(s) - c_s^{(0)}(s))^2 ds$$

$$- \text{ resisting bending} \\ E_2 = \int (c_{ss}(s) - c_{ss}^{(0)}(s))^2 ds \qquad E_{\text{int}} = \int_{-\beta}^{\alpha} (c_s(s) - c_s^{(0)}(s))^2 ds \\ + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 ds$$



Programming Methods

Comparison

There is a nontrivial prior shape (e.g., lip) represented by c ⁽⁰⁾

The prior shape is "zero"

Shrink-wrap models

$$E_{\rm int} = \int_{-\infty}^{\alpha (c_s(s) - c_s^{(0)}(s))^2} + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 ds$$

$$\varepsilon_{\text{int}}(\mathbf{v}(s)) = \left(\alpha(s) \|\mathbf{v}_s(s)\|^2 + \beta(s) \|\mathbf{v}_{ss}(s)\|^2\right) / 2$$





$$E = l | (x_o, y_o) - (x(s_o), y(s_o)) |^2$$

- attach the snake to a bright line (not used in hw)

$$E = -\int I(c(s))ds$$

- attach the snake to an edge

$$E = -\int (\nabla I(c(s)))^2 ds$$



- Treated as an minimization problem, we are looking for a function c(s) or f(s,t) that minimizes the total energy (int+ext)
- Intuitively,
 - small internal energy, less stretching, bending, twisting, closer to the natural resting state
 - small external energy, confirming to external constraints (e.g., close to attachment points, image contours, etc.)



- For those of you who are mathematics-gifted, you probably recognize this as a calculus of variation problem
- The solution is the Euler equation (a partial differential equation)
- The energy expression is a "functional"
- Need a function to give the extremal value of the "functional"



Calculus

- □ function
- □ locations (extremums of function)
- derivatives
- ordinary equations



Variational Calculus

- □ functional
- functions (extremums of functional)
- variational derivatives
- partial differential equations



For those of you who are physics-gifted, you probably recognize this as a generalized force problem

Again, the solution is based on the Euler equation (a partial differential equation) of variational derivatives



Math Detail

Need to maintain

- Length (no stretching)
- Curvature (no bending)

Both arc length and curvature are vectors!





Math Detail

- Most generally, allowing both translation and rotation (a rigid-body motion) that doesn't deform the shape
- Tangent and curvature vectors do not have to line up (under rotation), but their magnitude should be maintained
- Turn out the math becomes very messy



Simpler formulation: translation only (or small rotation)
Vectors should line up



Minimize

$$E_{total} = E_{int} + E_{ext}$$

= $\int \alpha (|c_s(s)| - |c_s^{(0)}(s)|)^2 + \beta (|c_{ss}(s)| - |c_{ss}^{(0)}(s)|)^2 - \delta \bigotimes(s)) - (\nabla I(c(s)))^2 ds$

Simplify (translation only, vector lined up)

$$E_{total} = E_{int} + E_{ext}$$

= $\int \alpha (c_s(s) - c_s^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \delta I \sum s) - (\nabla I(c(s)))^2 ds$

Discretize

$$c_{s}(s) = c_{i+1} - c_{i} = (x_{i+1} - x_{i}, y_{i+1} - y_{i})$$

$$c_{ss}(s) = c_{i+1} - 2c_{i} + c_{i-1} = (x_{i+1} - 2x_{i} + x_{i-1}, y_{i+1} - 2y_{i} + y_{i-1})$$

$$E(c) \Longrightarrow E(x_{o}, y_{o}, \dots, x_{n-1}, y_{n-1})$$





- Turn a variational calculus problem into a standard calculus problem
- 2n variables
- 2n equations (linear equations)
- Can solve a (very sparse) matrix equation of AX=B using Matlab A\B (or iterative)
- Sparsity comes from 1st and 2nd order derivative approximation using only neighboring points minimize



How to Get There?

Caveat

- □ This is not a numerical analysis course
- Options: Newton's method, line search and trust-region method, etc.
- Gradient descent
 - Go in the negative gradient direction in "some" length (step size)
 - How to tune the step size?

♦ What is the gradient direction? For n points, 2n variables $(x,y) \begin{bmatrix} \frac{\partial E}{\partial c_1} & \dots & \frac{\partial E}{\partial c_k} \end{bmatrix}^T$



Minimize

$$E_{total} = E_{int} + E_{ext}$$

= $\int \alpha (c_s(s) - c_s^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \delta V(s)) - (\nabla I(c(s)))^2 ds$

Discretize

$$c_{s}(s) = c_{i+1} - c_{i} = (x_{i+1} - x_{i}, y_{i+1} - y_{i})$$

$$c_{ss}(s) = c_{i+1} - 2c_{i} + c_{i-1} = (x_{i+1} - 2x_{i} + x_{i-1}, y_{i+1} - 2y_{i} + y_{i-1})$$
For a particular c_{i} :

$$(c_{s}(s) - c_{s}^{(0)}(s))^{2} = ([x_{i+1} - x_{i}, y_{i+1} - y_{i}] - [x^{(0)}_{i+1} - x^{(0)}_{i}, y^{(0)}_{i+1} - y^{(0)}_{i}])^{2}$$

$$= [(x_{i+1} - x_{i}) - (x^{(0)}_{i+1} - x^{(0)}_{i}), (y_{i+1} - y_{i}) - (y^{(0)}_{i+1} - y^{(0)}_{i})]^{2}$$

$$= ((x_{i+1} - x_{i}) - (x^{(0)}_{i+1} - x^{(0)}_{i}))^{2} + ((y_{i+1} - y_{i}) - (y^{(0)}_{i+1} - y^{(0)}_{i}))^{2}$$

$$\frac{\partial (c_{s}(s) - c_{s}^{(0)}(s))^{2}}{\partial x_{k}} = 2[-((x_{k+1} - x_{k}) - (x^{(0)}_{k+1} - x^{(0)}_{k})) + (x_{k} - x_{k-1}) - (x^{(0)}_{k} - x^{(0)}_{k-1})] + \cdots$$

$$\uparrow$$
1st derivatives of x_{k+1} and x_{k} involve x_{k}
Pattern: -1, 2, -1



Minimize

$$E_{total} = E_{int} + E_{ext}$$

= $\int \alpha (c_s(s) - c_s^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \sum_{ss} (c_s(s)) - (\nabla I(c(s)))^2 ds$

Discretize

$$c_{s}(s) = c_{i+1} - c_{i} = (x_{i+1} - x_{i}, y_{i+1} - y_{i})$$

$$c_{ss}(s) = c_{i+1} - 2c_{i} + c_{i-1} = (x_{i+1} - 2x_{i} + x_{i-1}, y_{i+1} - 2y_{i} + y_{i-1})$$

For a particular x_i :

$$(c_{ss}(s) - c_{ss}^{(0)}(s))^{2} = ([x_{i+1} - 2x_{i} + x_{i-1}, y_{i+1} - 2y_{i} + y_{i-1}] - [x^{(0)}_{i+1} - 2x^{(0)}_{i} + x^{(0)}_{i-1}, y^{(0)}_{i+1} - 2y^{(0)}_{i} + y^{(0)}_{i-1}])^{2}$$

$$= [(x_{i+1} - 2x_{i} + x_{i-1}) - (x^{(0)}_{i+1} - 2x^{(0)}_{i} + x^{(0)}_{i-1}), (y_{i+1} - 2y_{i} + y_{i-1}) - (y^{(0)}_{i+1} - 2y^{(0)}_{i} + y^{(0)}_{i-1})]^{2}$$

$$= ((x_{i+1} - 2x_{i} + x_{i-1}) - (x^{(0)}_{i+1} - 2x^{(0)}_{i} + x^{(0)}_{i-1}))^{2} + ((y_{i+1} - 2y_{i} + y_{i-1}) - (y^{(0)}_{i+1} - 2y^{(0)}_{i} + y^{(0)}_{i-1}))^{2}$$

$$\frac{\partial (c_{ss}(s) - c_{ss}^{(0)}(s))^{2}}{\partial x_{k}} =$$

$$2[((x_{k+2} - 2x_{k+1} + x_{k}) - (x^{(0)}_{k+2} - 2x^{(0)}_{k+1} + x^{(0)}_{k}))] +$$

$$2[-2((x_{k+1} - 2x_{k} + x_{k-1}) - (x^{(0)}_{k+1} - 2x^{(0)}_{k} + x^{(0)}_{k-1})] +$$

 $2[((x_k - 2x_{k-1} + x_{k-2}) - (x^{(0)}_k - 2x^{(0)}_{k-1} + x^{(0)}_{k-2}))]$

2nd derivatives of x_{k+1} , x_k and x_{k-1} involve x_k Pattern: 1, -4, 6, -4, 1



Minimize

$$E_{total} = E_{int} + E_{ext}$$

= $\int \alpha (c_s(s) - c_s^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \delta i \sum s (s) - (\nabla I(c(s)))^2 ds$

Discretize

$$c_{s}(s) = c_{i+1} - c_{i} = (x_{i+1} - x_{i}, y_{i+1} - y_{i})$$

$$c_{ss}(s) = c_{i+1} - 2c_{i} + c_{i-1} = (x_{i+1} - 2x_{i} + x_{i-1}, y_{i+1} - 2y_{i} + y_{i-1})$$
For a particular c_{i} :
$$(\nabla I(c(s)))^{2} = [I_{x}(x_{i}, y_{i}), I_{y}(x_{i}, y_{i})]^{2} = (I_{x}(x_{i}, y_{i}))^{2} + (I_{y}(x_{i}, y_{i}))^{2}$$

$$\frac{\partial (\nabla I(c(s)))^{2}}{\partial x_{k}} = 2[I_{x}(x_{k}, y_{k})\frac{\partial I_{x}}{\partial x_{k}} + I_{y}(x_{k}, y_{k})\frac{\partial I_{y}}{\partial x_{k}}] = 2[I_{x}(x_{k}, y_{k}), I_{y}(x_{k}, y_{k})][\frac{\partial I_{x}}{\partial x_{k}}, \frac{\partial I_{y}}{\partial x_{k}}]^{T}$$



Minimize

$$E_{total} = E_{int} + E_{ext}$$

$$= \int \alpha (c_s(s) - c_s^{(0)}(s))^2 + \beta (c_{ss}(s) - c_{ss}^{(0)}(s))^2 - \delta I(c(s)) - (\nabla I(c(s)))^2 ds$$

$$\frac{\partial (\nabla I(c(s)))^2}{\partial x_k} = 2[I_x(x_k, y_k)\frac{\partial I_x}{\partial x_k} + I_y(x_k, y_k)\frac{\partial I_y}{\partial x_k}] = 2[I_x(x_k, y_k), I_y(x_k, y_k)][\frac{\partial I_x}{\partial x_k}, \frac{\partial I_y}{\partial x_k}]^T$$

- Derivative of E (potential) is a gradient (force) field
- Minimization go in the negative gradient direction
- Pull the snake in the direction
 - Large gradient
 - Large increase in gradient
 - around a node



Details

- * α(-1,2,-1) +β (1 -4, 6, -4, 1) = (1, -5, 8, -5, 1) (if α=β)
 - □ Not diagonally dominant, need conditioning (regularization)
- Resulting in gradient of form AX-AX₀-B
 A is pentdiagonal matrix of the form (1, -5, 8, -5, 1) (if α=β)

 \Box AX₀ has internal energy term

- > Template (x⁽⁰⁾,y⁽⁰⁾)
 - Fixed (original template) or varying (last time instant)
 - Has the form of -AX⁽⁰⁾ or -AX^(t-1) or some combination
- B External energy term <- varying</p>

$$[I_x(x_k, y_k), I_y(x_k, y_k)][\frac{\partial I_x}{\partial x_k}, \frac{\partial I_y}{\partial x_k}]^T$$



The equation represents balance of forces!





Caveat:

Snake needs good initial position
Provided by initial interactive placement
Smooth images to enlarge "potential field"
Snake won't move if

- Gradient is zero or
- Change of gradient is zero



Numerical Methods - Iterative

Using Euler's method: expressions AX-AX₀-B are gradient
 Explicit Euler: $-\lambda(AX_{t-1} - AX_0 - B_{t-1}) = (X_t - X_{t-1})$ $X_t = X_{t-1} - \lambda(AX_{t-1} - AX_0 - B_{t-1})$

Mixed Euler:
$$-\lambda(AX_{t} - AX_{0} - B_{t}) = (X_{t} - X_{t-1})$$

$$-\lambda(AX_{t} - AX_{0} - B_{t-1}) = (X_{t} - X_{t-1})$$

$$(\lambda A + I)X_{t} = \lambda AX_{0} + \lambda B_{t-1} + X_{t-1}$$

$$X_{t} = (\lambda A + I)^{-1}(\lambda AX_{0} + \lambda B_{t-1} + X_{t-1})$$

Justification for mixed Euler:

- \Box B cannot be evaluated without knowing X_t, so use values at X_{t-1}
- \Box A can be easily inverted, so use X_t
- □ Your choice, explicit Euler for simplicity



Numerical Methods - Direct

Should result in a sparse, pentadiagonal matrix

*****AX = B, solve with

□ Direct method $\mathbf{X} = inv(\mathbf{A}) * \mathbf{B}$ (preferred for small system < 20 points and GOOD initialization)

Caveats:

▲ A can be numerically ill-conditioned (not diagonally dominant – the |diagonal element| is larger than the sum of |off-diagonal elements|)

Fix: Regularization (a topic to be discussed more later)

- $\texttt{Minimize} \parallel AX-B \parallel^{2} + w \parallel X \parallel^{2}$
- (A'A+wI) X = A'B or X = inv(A'A+wI)*A'B


Numerical Methods





Direct or Iterative

No iteration

- Initial state must be close to final state (because external energy is position dependent), image smoothing is important
- Require good template and update of templates

Iterative

- Initial state does not have to be close to final state
- External energy terms must be updated through out

