Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary

Main idea: elastic band is iteratively adjusted so as to
• be near image positions with high gradients, and
• satisfy shape “preferences” or contour priors

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV'1987]
Last time: Deformable contours

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

Today

• Interactive segmentation
• Feature-based alignment
  – 2D transformations
  – Affine fit
  – RANSAC

Interactive forces

How can we implement such an interactive force with deformable contours?
Interactive forces

- An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
- Modify external energy term to include a term such that

\[ \sum_{i} (-E_i) = -\sum_{i} \rho \cdot ||n_i|| \]

Nearby points get pushed hardest

Adapted by Devi Parikh from Kristen Grauman

Beyond boundary snapping...

- Another form of interactive guidance: specify regions
- Usually taken to suggest foreground/background color distributions

Boykov and Jolly (2001)

Recall: Images as graphs

- Fully-connected graph
  - node for every pixel
  - link between every pair of pixels, \( p,q \)
  - similarity \( w_{pq} \) for each link

\( w_{pq} \) is inversely proportional to difference in color and position

Steve Seitz
Recall: Segmentation by Graph Cuts

Break graph into segments
- Delete links that cross between segments
- Easiest to break links that have low similarity
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments

Link Cut
- set of links whose removal makes a graph disconnected
- cost of a cut: \[ \text{cut}(A, B) = \sum_{p \in A, q \in B} w_{pq} \]

Find minimum cut
- gives you a segmentation
- fast algorithms exist for doing this

Graph cuts for interactive segmentation

Adding hard constraints:
Add two additional nodes, object and background “terminals”
Link each pixel
- To both terminals
- To its neighboring pixels
Graph cuts for interactive segmentation

Adding hard constraints:
Let the edge weight to object or background terminal reflect distance to the respective seed pixels.

\[ D_s(s) = |I_s - I' | \]
\[ D_s(t) = |I_s - I' | \]

Boykov and Jolly (2001)

Graph cuts for interactive segmentation

\[ E(L) = \sum_{p} D_s(L_p) + \sum_{p \in \gamma \in \gamma} w_{\gamma} \cdot \delta(L_p \neq L_p) \]

Another interaction modality: specify bounding box

Slide credit: Kristen Grauman
"Grab Cut"

- Loosely specify foreground region
- Iterated graph cut

User initialization

K-means for learning color distributions

Graph cuts to infer the segmentation

Today

- Interactive segmentation
- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC

Motivation: Recognition

Figures from David Lowe

Motivation: medical image registration

Slide credit: Kristen Grauman
Motivation: mosaics

(In detail next week)


Alignment problem

• We have previously considered how to fit a model to image evidence
  – e.g., a line to edge points, or a snake to a deforming contour
• In alignment, we will fit the parameters of some transformation according to a set of matching feature pairs (“correspondences”).

\[ T \]

Slide credit: Kristen Grauman

Parametric (global) warping

Examples of parametric warps:

- Translation
- Rotation
- Aspect
- Affine
- Perspective

Source: Alyosha Efros
Parametric (global) warping

Transformation $T$ is a coordinate-changing machine:

$$p' = T(p)$$

What does it mean that $T$ is global?
- Is the same for any point $p$
- Can be described by just a few numbers (parameters)

Let's represent $T$ as a matrix:

$$p' = Mp$$

Scaling

Scaling a coordinate means multiplying each of its components by a scalar.

Uniform scaling means this scalar is the same for all components:

Non-uniform scaling: different scalars per component:
Scaling

Scaling operation:

\[ x' = ax \]
\[ y' = by \]

Or, in matrix form:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  a & 0 \\
  0 & b
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

scaling matrix \( S \)

What transformations can be represented with a 2x2 matrix?

2D Scaling?

\[ x' = s_x \cdot x \]
\[ y' = s_y \cdot y \]

2D Rotate around \((0,0)\)?

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  \cos \Theta & -\sin \Theta \\
  \sin \Theta & \cos \Theta
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

2D Shear?

\[ x' = x + s_h \cdot y \]
\[ y' = s_h \cdot x + y \]

2D Mirror about Y axis?

\[ x' = -x \]
\[ y' = y \]

2D Mirror over \((0,0)\)?

\[ x' = -x \]
\[ y' = -y \]

2D Translation?

\[ x' = x + t_x \]
\[ y' = y + t_y \]

NO!
2D Linear Transformations

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} = \begin{bmatrix}
    a & b \\
    c & d
\end{bmatrix} \begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

Only linear 2D transformations can be represented with a 2x2 matrix.
Linear transformations are combinations of …
• Scale,
• Rotation,
• Shear, and
• Mirror

Homogeneous coordinates

Convenient coordinate system to represent many useful transformations

To convert to homogeneous coordinates:
\[(x, y) \Rightarrow \begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

Converting from homogeneous coordinates:
\[
\begin{bmatrix}
    x \\
    y \\
    w
\end{bmatrix} \Rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)
\]

Homogeneous Coordinates

Q: How can we represent 2d translation as a 3x3 matrix using homogeneous coordinates?

\[
x' = x + t_x \\
y' = y + t_y
\]

A: Using the rightmost column:

\[
\text{Translation} = \begin{bmatrix}
    1 & 0 & t_x \\
    0 & 1 & t_y \\
    0 & 0 & 1
\end{bmatrix}
\]
Translation
Homogeneous Coordinates

\[
\begin{pmatrix}
x' \\ y' \\ 1
\end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

\[t_x = 2\]
\[t_y = 1\]

Basic 2D Transformations
Basic 2D transformations as 3x3 matrices

\[
\begin{pmatrix}
x' \\ y' \\ 1
\end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Translate

\[
\begin{pmatrix}
x' \\ y' \\ 1
\end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Rotate

\[
\begin{pmatrix}
x' \\ y' \\ 1
\end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Scale

\[
\begin{pmatrix}
x' \\ y' \\ 1
\end{pmatrix} = \begin{pmatrix} ax & cy & 0 \\ bx & dy & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Shear

2D Affine Transformations

Affine transformations are combinations of …
- Linear transformations, and
- Translations

Parallel lines remain parallel
Today

- Interactive segmentation
- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC

Alignment problem

- We have previously considered how to fit a model to image evidence
  - e.g., a line to edge points, or a snake to a deforming contour
- In alignment, we will fit the parameters of some transformation according to a set of matching feature pairs (“correspondences”).

Image alignment

- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where extracted features agree
    - Can be verified using pixel-based alignment

Slide credit: Kristen Grauman
Fitting an affine transformation

• Assuming we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  m_1 & m_2 & x_i & t_x \\
  m_3 & m_4 & y_i & t_y
\end{bmatrix}
\begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix} +
\begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

An aside: Least Squares Example

Say we have a set of data points \((X1,X1'), (X2,X2'), (X3,X3'),\) etc. (e.g. person’s height vs. weight)

We want a nice compact formula (a line) to predict \(X'\) from \(X\):

\[Xa + b = X'\]

We want to find \(a\) and \(b\)

How many \((X,X')\) pairs do we need?

\[
\begin{align*}
X_1a + b &= X_1' \\
X_2a + b &= X_2'
\end{align*}
\]

What if the data is noisy?

\[
\begin{bmatrix}
X_1 & 1 \\
X_2 & 1 \\
\vdots & \vdots
\end{bmatrix}\begin{bmatrix}
a \\
b
\end{bmatrix} =
\begin{bmatrix}
X_1' \\
X_2' \\
\vdots
\end{bmatrix}
\]

\[
\begin{bmatrix}
X_1 & 1 & X_1' \\
X_2 & 1 & X_2' \\
\vdots & \vdots & \vdots
\end{bmatrix}
\]

\[
\begin{bmatrix}
a \\
b
\end{bmatrix} =
\min \|Ax - \mathbf{B}\|
\]

overconstrained

Source: Alyosha Efros
Fitting an affine transformation

\[
\begin{bmatrix}
  x_1 & y_1 & 0 & 0 & 1 & 0 \\
  0 & 0 & x_1 & y_1 & 0 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  t_1 & t_2
\end{bmatrix} \begin{bmatrix}
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  m_5 \\
  m_6
\end{bmatrix} = \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  \vdots \\
  t'_1
\end{bmatrix}
\]

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?

Affine: # correspondences?

How many correspondences needed for affine?

Fitting an affine transformation

\[
\begin{bmatrix}
  x_1 & y_1 & 0 & 0 & 1 & 0 \\
  0 & 0 & x_1 & y_1 & 0 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  t_1 & t_2
\end{bmatrix} \begin{bmatrix}
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  m_5 \\
  m_6
\end{bmatrix} = \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  \vdots \\
  t'_1
\end{bmatrix}
\]

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for \((x_{new}, y_{new})\) ?
- Where do the matches come from?
What are the correspondences?

- Compare content in local patches, find best matches. 
  e.g., simplest approach: scan with template, and compute SSD 
  or correlation between list of pixel intensities in the patch 
- Later in the course: how to select regions using more 
  robust descriptors.

Fitting an affine transformation

Figures from David Lowe, ICCV 1999

Example from UBC SIFT Demo
Today

- Interactive segmentation
- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC
Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn’t belong to the line we are fitting.

Outliers affect least squares fit

![Graph showing outliers affecting least squares fit](Slide credit: Kristen Grauman)

Outliers affect least squares fit

![Graph showing outliers affecting least squares fit](Slide credit: Kristen Grauman)
RANSAC

• RANdom Sample Consensus

• Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

• Intuition: if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.

RANSAC: General form

• RANSAC loop:
  1. Randomly select a seed group of points on which to base transformation estimate
  2. Compute transformation from seed group
  3. Find inliers to this transformation
  4. If the number of inliers is sufficiently large, re-compute estimate of transformation on all of the inliers

• Keep the transformation with the largest number of inliers

RANSAC for line fitting example

Source: R. Raguram, Lana Lazebnik
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model

Least-squares fit

Source: R. Raguram Lana Lazebnik
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop
RANSAC for line fitting example

Source: R. Raguram, Lana Lazebnik
RANSAC for line fitting

Repeat \( N \) times:
- Draw \( s \) points uniformly at random
- Fit line to these \( s \) points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than \( t \))
- If there are \( d \) or more inliers, accept the line and refit using all inliers

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  - Can’t always get a good initialization of the model based on the minimum number of samples

Today

- Interactive segmentation
- Feature-based alignment
  - 2D transformations
  - Affine fit
  - RANSAC
Coming up: alignment and image stitching

Questions?
See you Thursday!