Fitting a transformation:
Feature-based alignment
May 2nd, 2017
Yong Jae Lee
UC Davis

Given: initial contour (model) near desired object
a.k.a. active contours, snakes

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'87]

Last time: Deformable contours

Last time: Deformable contours
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

\[ \sum_{i=1}^{\infty} \| n_{i} \|^2 \cdot p \cdot E \]  

Nearby points get pushed hardest

Intelligent scissors

Another form of interactive segmentation:

Compute optimal paths from every point to the seed based on edge-related costs.

Figure 2: Image demonstrating how the interactive segment assigns and overlaps (from object boundary to the first point) means (from source mean). The path of the first point is shown in white. Line-source segments from previous free point positions \( x_i \) and \( x_j \) are shown in green.

Intelligent scissors

http://rivit.cs.byu.edu/Eric/Enzo.html

Slide credit: 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
    - similarity is inversely proportional to difference in color and position

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:
  \[ \text{cut}(A, B) = \sum_{p \in A, q \in B} w_{p,q} \]

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

Recall: Segmentation by Graph Cuts

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

Let the edge weight to object or background terminal reflect similarity to the respective seed pixels.

\[ D_o(x) = -|I_o - I'| \]
\[ D_b(x) = -|I_b - I'| \]
Graph cuts for interactive segmentation

\[ E(L) = \sum_p -D_p(L_p) + \sum_{p \to q} w_{pq} \delta(L_p \neq L_q) \]

\[ L_p \in \{s, t\} \]

Another interaction modality: specify bounding box

"Grab Cut"

- Loosely specify foreground region
- Iterated graph cut

User initialization

K-means for learning colour distributions

Graph cuts to infer the segmentation

Rother et al. (2004)
“Grab Cut”

- Loosely specify foreground region
- Iterated graph cut


Gaussian Mixture Model (typically 5-8 components)

“Grab Cut”


Today

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

Slide credit: Kristen Grauman
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”).

Parametric (global) warping

Examples of parametric warps:

- translation
- rotation
- aspect
- affine
- perspective

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:

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
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 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)?
\[ x' = \cos \theta \cdot x - \sin \theta \cdot y \]
\[ y' = \sin \theta \cdot x + \cos \theta \cdot y \]

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

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:

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\]

homogeneous image coordinates

Converting from homogeneous coordinates:

\[\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)\]

---

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

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

\[
(\Delta x, \Delta y, 1) = (x' + t_x, y' + t_y, 1)
\]

---

Source: Alyosha Efros
Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

Translate
\[
\begin{pmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{pmatrix}
\]

Scale
\[
\begin{pmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

Rotate
\[
\begin{pmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

Shear
\[
\begin{pmatrix}
1 & \alpha & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

2D Affine Transformations

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

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

Fitting an affine transformation

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

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  m_{11} & m_{21} & x_1 \\
  m_{12} & m_{22} & y_1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix} +
\begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]
An aside: Least Squares Example

Say we have a set of data points \((X_1, X_1'), (X_2, X_2'), (X_3, X_3'), \ldots\) (e.g. person's height vs. weight)

We want a nice compact formula (a line) to predict \(X_1'\) from \(X_1\):

\[ X_1 a + b = X_1' \]

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

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

What if the data is noisy?

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

\[\text{overconstrained}\]

\[\text{min} \| \mathbf{a} - 8 \mathbf{f} \| \]

Fitting an affine transformation

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

\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix}
= 
\begin{bmatrix}
m_1 & m_2 & s_1 & i_1 \\
m_3 & m_4 & s_2 & i_2
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
s_1 \\
i_1
\end{bmatrix}
\]

• 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

- 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

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

Fitting an affine transformation

Example from UBC SIFT Demo

Fitting an affine transformation

Example from UBC SIFT Demo
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

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

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 (e.g., a group of match)
  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

Slide credit: Kristen Grauman

RANSAC for line fitting example

Source: R. Raguram

RANSAC for line fitting example

Source: R. Raguram

Least-squares fit
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
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

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesis-and-verify loop

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

Source: R. Raguram, Lana Lazebnik

Source: R. Raguram
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!