Fitting a transformation: feature-based alignment

Wed, Feb 23
Prof. Kristen Grauman
UT-Austin

Announcements

• Reminder: Pset 2 due Wed March 2
• Midterm exam is Wed March 9
  (2 weeks from now)

Last time: Deformable contours

a.k.a. active contours, snakes

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

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?

Kristen Grauman

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:

\[ E_{\text{push}} = \sum_{i=0}^{n-1} \left( ||n_i - p||^2 \right) \]

Nearby points get pushed hardest

Kristen Grauman

Intelligent scissors

Another form of interactive segmentation:
Compute optimal paths from every point to the seed based on edge-related costs.

[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

Intelligent scissors

Beyond boundary snapping…

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

Boykov and Jolly (2001)

How to use this information?

Kristen Grauman
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

Graph cuts for interactive segmentation

- Adding hard constraints:
  - Add two additional nodes, object and background "terminals"
  - Let the edge weight to object or background terminal reflect similarity to the respective seed pixels.
Graph cuts for interactive segmentation

“Grab Cut”
- Loosely specify foreground region
- Iterated graph cut

“Grab Cut”
- Loosely specify foreground region
- Iterated graph cut

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

Motivation: Recognition
Figures from David Lowe
Motivation: medical image registration

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

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 operation:

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

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?

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

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

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

To convert to homogeneous coordinates:

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

Converting from homogeneous coordinates

\[
\begin{bmatrix}
 x/w \\
 y/w \\
 1
\end{bmatrix} \Rightarrow (x, y)
\]
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:
\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\]

Basic 2D Transformations
Basic 2D transformations as 3x3 matrices
- **Translate**
  \[
  \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
  \end{bmatrix}
  \]
- **Scale**
  \[
  \begin{bmatrix}
  a & 0 & 0 \\
  0 & b & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
- **Rotate**
  \[
  \begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]
- **Shear**
  \[
  \begin{bmatrix}
  1 & sb & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
  \end{bmatrix}
  \]

Translation
\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\]

2D Affine Transformations
Affine transformations are combinations of ...
- Linear transformations, and
- Translations
Parallel lines remain parallel

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”).

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

\[
\begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots
\end{bmatrix} =
\begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4 \\
  m_5 & m_6 \\
end{bmatrix}
\begin{bmatrix}
  x_1 \\
  y_1 \\
  x_2 \\
  y_2 \\
  \vdots
\end{bmatrix} +
\begin{bmatrix}
  t_1 \\
  t_2 \\
  t_3 \\
  t_4 \\
  \vdots
\end{bmatrix}
\]

An aside: Least Squares Example

Say we have a set of data points \((X_1, X'_1), (X_2, X'_2), (X_3, X'_3), \text{ etc.}\) (e.g. person's height vs. weight)

We want a nice compact formula (a line) to predict \(X'\)s from \(X\)s: \(X = ax + b = X'\)

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

What if the data is noisy?

\[
\begin{bmatrix}
  X_1 & 1 \\
  X_2 & 1 \\
  X_3 & 1 \\
  \vdots & \vdots
\end{bmatrix}
\begin{bmatrix}
  a \\
  b
\end{bmatrix} =
\begin{bmatrix}
  X'_1 \\
  X'_2 \\
  X'_3 \\
  \vdots
\end{bmatrix}
\]

\[
\text{min} \left\| Ax - B \right\|
\]

overconstrained

Fitting an affine transformation

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

\[
\begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots
\end{bmatrix} =
\begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4 \\
  m_5 & m_6 \\
end{bmatrix}
\begin{bmatrix}
  x_1 \\
  y_1 \\
  x_2 \\
  y_2 \\
  \vdots
\end{bmatrix} +
\begin{bmatrix}
  t_1 \\
  t_2 \\
  t_3 \\
  t_4 \\
  \vdots
\end{bmatrix}
\]

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 according to the geometric changes, and more robust descriptors.
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

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

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.

Today

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

- **RANSAC loop:**
  1. Randomly select a seed group of points on which to base transformation estimate (e.g., a group of matches)
  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

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

---

Least-squares fit

---

1. Randomly select minimal subset of points
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

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 hypothesize-and-verify loop

RANSAC for line fitting example

Uncontaminated sample

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

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
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