Fitting a transformation: Feature-based alignment

April 30th, 2019

Yong Jae Lee
UC Davis

Generalized Hough Transform

• What if we want to detect arbitrary shapes?

Intuition:

Now suppose those colors encode gradient directions...

Generalized Hough Transform

• Define a model shape by its boundary points and a reference point.

Offline procedure:

At each boundary point, compute displacement vector: \( r = a - p_t \).

Store these vectors in a table indexed by gradient orientation \( \theta \).

(Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980)
Generalized Hough Transform

**Detection procedure:**
For each edge point:
- Use its gradient orientation $\theta$ to index into stored table
- Use retrieved $r$ vectors to vote for reference point

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by matched local patterns.

Source: L. Lazebnik

B. Leibe, A. Leonardis, and B. Schiele, *Combined Object Categorization and Segmentation with an Implicit Shape Model*, ECCV Workshop on Statistical Learning in Computer Vision 2004

Generalized Hough for object detection

- Instead of indexing displacements by gradient orientation, index by matched local patterns.

Source: L. Lazebnik

B. Leibe, A. Leonardis, and B. Schiele, *Combined Object Categorization and Segmentation with an Implicit Shape Model*, ECCV Workshop on Statistical Learning in Computer Vision 2004
Summary

- **Fitting** problems require finding any supporting evidence for a model, even within clutter and missing features
  - associate features with an explicit model

- **Voting** approaches, such as the Hough transform, make it possible to find likely model parameters without searching all combinations of features
  - Hough transform approach for lines, circles, ..., arbitrary shapes defined by a set of boundary points, recognition from patches

Today

- 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

- 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:

$$\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} = \begin{bmatrix}
  x \\
  y
\end{bmatrix} M$$

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:
\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= 
\begin{bmatrix}
  a & 0 \\
  b & 0
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]
scaling matrix \( S \)

What transformations can be represented with a 2x2 matrix?

2D Scaling?
\[
\begin{align*}
x' &= s_x \cdot x \\
y' &= s_y \cdot y
\end{align*}
\]
\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix}
= 
\begin{bmatrix}
s_x & 0 \\
0 & s_y
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

2D Rotate around (0,0)?
\[
\begin{align*}
x' &= \cos\theta \cdot x - \sin\theta \cdot y \\
y' &= \sin\theta \cdot x + \cos\theta \cdot y
\end{align*}
\]
\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix}
= 
\begin{bmatrix}
\cos\theta & -\sin\theta \\
\sin\theta & \cos\theta
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]

2D Shear?
\[
\begin{align*}
x' &= x + sh_x \cdot y \\
y' &= sh_y \cdot x + y
\end{align*}
\]
\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix}
= 
\begin{bmatrix}
1 & sh_x \\
sh_y & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix}
\]
What transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?
\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\
  y
\end{pmatrix}
\]

2D Mirror over (0,0)?
\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\
  y
\end{pmatrix}
\]

2D Translation?
\[
\begin{pmatrix}
  x' = x + t_x \\
  y' = y + t_y
\end{pmatrix}
\]

2D Linear Transformations
\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\
  y
\end{pmatrix}
\]

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

Converting from homogeneous coordinates:
\[
\begin{pmatrix} x \\
  y \\
  w
\end{pmatrix} \Rightarrow (x/w, y/w)
\]
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:

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

Translation

Homogeneous Coordinates

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

Source: Alyosha Efros

Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

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

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

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

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

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

Source: Alyosha Efros
2D Affine Transformations

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

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

Parallel lines remain parallel

Today

• 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

• 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?

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

An aside: Least Squares Example

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

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

\[Xa + b = X'\]

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?

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

\[
\min \|Ax - b\|
\]

overconstrained
Fitting an affine transformation

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

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

Slide credit: Kristen Grauman

Fitting an affine transformation

\[
\begin{bmatrix}
  x_i & y_i & 0 & 0 & 1 & 0 \\
  0 & 0 & x_i & y_i & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  m_1 & \cdots & m_i & \cdots \\
  m_2 & \cdots & m_i & \cdots
\end{bmatrix}
= \begin{bmatrix}
  x'_i \\
  y'_i
\end{bmatrix}
\begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

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

Affine: # correspondences?

How many correspondences needed for affine?

Slide credit: Alyosha Efros
Fitting an affine transformation

\[
\begin{bmatrix}
  x_0 & y_0 & 0 & 0 & 1 & 0 \\
  0 & 0 & x_0 & y_0 & 0 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  t_0 & t_1 & t_2 & t_3 & t_4 & t_5
\end{bmatrix}
\begin{bmatrix}
  m_0 \\
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  m_5
\end{bmatrix}
= \begin{bmatrix}
  x'_0 \\
  x'_1 \\
  x'_2 \\
  y'_0 \\
  y'_1 \\
  y'_2
\end{bmatrix}
\]

• 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
Today

• 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
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  
  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

Source: R. Raguram, Lana Lazebnik
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

Source: R. Raguram Lana Lazebnik
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

Source: R. Raguram, Lana Lazebnik
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

Source: R. Raguram

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)
Today

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

Coming up:
alignment and image stitching

Questions?
See you Thursday!