Menu February 25, 2016

Topics:
   SIFT (cont.)

Reading:
   Today:   Paper: “Distinctive Image Features from Scale-Invariant Keypoints” Forsyth & Ponce (2nd ed.) 5.4, 10.4.2
   Next:    Forsyth & Ponce (2nd ed.) 10.1, 10.2

Handouts:
   Assignment 5: Local Invariant Features

Reminders:
   Assignment 4 due now
   Middle-of-term survey: https://survey.ubc.ca/s/cpsc-425/
   piazza: https://piazza.com/ubc.ca/winterterm22015/cpsc425/
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Today’s “Fun” Example: Recognizing Panoramas

Figure credit: Matthew Brown & David Lowe
Lecture 12: Re-cap

- We motivated SIFT for identifying locally distinct features in an image.

- SIFT features are invariant to translation, rotation, and scale; robust to 3D pose and illumination.

- First step in SIFT feature generation:
  1. Scale-space representation and local extrema detection
     - use DoG pyramid
     - 3 scales/octave, down-sample by factor of 2 each octave
Build Scale-Space Pyramid

- All scales must be examined to identify scale-invariant features
- An efficient function is to compute the Difference of Gaussian (DOG) pyramid

![Diagram of building a scale-space pyramid]

- Resample
- Blur
- Subtract
Scale Space Processed One Octave at a Time
Keypoint Localization

Detect maxima and minima of Difference of Gaussian in scale space
Keypoint Localization

- After keypoints are detected, we remove those that have low contrast or are poorly localized along an edge.

How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?
Keypoint Localization

- After keypoints are detected, we remove those that have low contrast or are poorly localized along an edge.

  How do we decide whether a keypoint is poorly localized, say along an edge, vs. well-localized?

- Lowe suggests computing the ratio of the eigenvalues of $\mathbf{C}$ (recall Harris corners) and checking if it is greater than a threshold.

  Aside: The ratio can be computed efficiently in fewer than 20 floating point operations, using a trick involving the trace and determinant of $\mathbf{C}$ - no need to explicitly compute the eigenvalues.
Keypoint Localization

Example:

(a) 233 × 189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principal curvatures
Select Canonical Orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates \((x, y, \text{scale}, \text{orientation})\)
Keypoint Description

- We have seen how to assign a location, scale, and orientation to each keypoint — keypoint detection

- The next step is to compute a keypoint descriptor — should be robust to local shape distortions, changes in illumination or 3D viewpoint

- Keypoint detection is not the same as keypoint description — e.g. some applications skip keypoint detection and extract SIFT descriptors on a regularly spaced grid
SIFT Descriptor

- Thresholded image gradients are sampled over $16 \times 16$ array of locations in scale space
- Create array of orientation histograms
- 8 orientations $\times$ $4 \times 4$ histogram array
SIFT Descriptor

How many dimensions are there in a SIFT descriptor?

(Hint: This diagram shows a 2 x 2 histogram array but the actual descriptor uses a 4 x 4 histogram array)
Descriptor is normalized to unit length (i.e. magnitude of 1) to reduce the effects of illumination change

— if brightness values are scaled (multiplied) by a constant, the gradients are scaled by the same constant, and the normalization cancels the change
SIFT Descriptor

- Descriptor is normalized to unit length (i.e. magnitude of 1) to reduce the effects of illumination change

  — if brightness values are scaled (multiplied) by a constant, the gradients are scaled by the same constant, and the normalization cancels the change

  — if brightness values are increased/decreased by a constant, the gradients do not change
Feature Stability to Noise

- Match features after random change in image scale & orientation, with differing levels of image noise
- Find nearest neighbour in database of 30,000 features

![Graph showing feature stability to noise]
Feature Stability to Affine Change

- Match features after random change in image scale & orientation, with 2% image noise and affine distortion
- Find nearest neighbour in database of 30,000 features
Distinctiveness of Features

- Vary size of database of features, with 30 degree affine change, 2% image noise
- Measure % correct for single nearest neighbour match

![Graph showing the distinctiveness of features](image)
Summary

Four steps to SIFT feature generation:

1. Scale-space representation and local extrema detection
   — use DoG pyramid
   — 3 scales/octave, down-sample by factor of 2 each octave

2. Keypoint localization
   — select stable keypoints (threshold on magnitude of extremum, ratio of principal curvatures)

3. Keypoint orientation assignment
   — based on histogram of local image gradient directions

4. Keypoint descriptor
   — (a more detailed) histogram of local gradient directions
   — vector with $8 \times (4 \times 4) = 128$ dimensions
   — vector normalized (to unit length)
Framework for Next Topic

Problem:
Object recognition in the presence of clutter and occlusion

Key Idea(s):
- Match keypoints independently to a database of known keypoints built from “training” examples
- Identify clusters of (at least) 3 matches that agree on object identity and similarity transformation
- Verify clusters using an affine transformation

Practical Detail(s):
There will be a large percentage of outliers
Object recognition requires us to first match each keypoint independently to the database of keypoints.

Many features will not have any correct match in the database because they arise from background clutter.

It would be useful to have a way to discard features that do not have any good match.
Probability of Correct Match

- Compare ratio of distance of nearest neighbour to second nearest neighbour (from different object)
- Threshold of 0.8 provides excellent separation
Nearest-Neighbour Matching to Feature Database

- Hypotheses are generated by approximate nearest neighbour matching of each feature to vectors in the database
  - Use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1,000 while finding nearest neighbour (of interest) 95% of the time
Identifying consistent features

- We have matched keypoints to a database of known keypoints extracted from training images.

- Next we identify clusters of at least 3 features that agree on an object and its pose. A typical image contains 2,000+ features → detecting less than 1% inliers among 99% outliers!

- Lowe’s solution uses the generalized Hough transform: vote for each potential match according to model ID and pose, insert into multiple bins to allow for error in similarity approximation. (More on Hough transforms later.)
Model Verification

1. Examine all clusters with at least 3 features
2. Perform least-squares affine fit to model
3. Discard outliers and perform top-down check for additional features
4. Evaluate probability that match is correct
   - Use Bayesian model, with probability that features would arise by chance if object was not present (Lowe, CVPR 01)
Solution for Affine Parameters

- Affine transform of \([x, y]\) to \([u, v]\):

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix} = \begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4
\end{bmatrix} \begin{bmatrix}
  x \\
  y
\end{bmatrix} + \begin{bmatrix}
  t_x \\
  t_y
\end{bmatrix}
\]

- Rewrite to solve for transform parameters:

\[
\begin{bmatrix}
  x & y & 0 & 0 & 1 & 0 \\
  0 & 0 & x & y & 0 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots 
\end{bmatrix} \begin{bmatrix}
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  t_x \\
  t_y
\end{bmatrix} = \begin{bmatrix}
  u \\
  v \\
  \vdots
\end{bmatrix}
\]
Suppose we have $k \geq 3$ matches, $[x_i, y_i]$ to $[u_i, v_i]$, $i = 1, 2, \ldots, k$.

Then,

$$
\begin{bmatrix}
x_1 & y_1 & 0 & 0 & 1 & 0 \\
0 & 0 & x_1 & y_1 & 0 & 1 \\
x_2 & y_2 & 0 & 0 & 1 & 0 \\
0 & 0 & x_2 & y_2 & 0 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_k & y_k & 0 & 0 & 1 & 0 \\
0 & 0 & x_k & y_k & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
m_1 \\
m_2 \\
m_3 \\
m_4 \\
t_x \\
t_y \\
\end{bmatrix}
=
\begin{bmatrix}
u_1 \\
v_1 \\
u_2 \\
v_2 \\
\vdots \\
u_k \\
v_k \\
\end{bmatrix}
$$
3D Object Recognition

- Extract outlines with background subtraction
3D Object Recognition (cont’d)

- Only 3 keys are needed for recognition, so extra keys provide robustness
Recognition Under Occlusion
Location Recognition
Example 1: Sony Aibo (Evolution Robotics)

SIFT usage:

- Recognize charging station
- Communicate with visual cards
Summary of object recognition with SIFT

- Match each keypoint independently to database of known keypoints extracted from “training” examples
  — use fast (approximate) nearest neighbour matching
  — threshold based on ratio of distances to best and to second best match

- Identify clusters of (at least) 3 matches that agree on an object and a similarity pose
  — use generalized Hough transform

- Check each cluster found by performing detailed geometric fit of affine transformation to the model
  — accept/reject interpretation accordingly