Lecture 4.2
Feature matching

Trym Vegard Haavardsholm
Overview of point feature matching

1. Detect a set of distinct feature points
2. Define a patch around each point
3. Extract and normalize the patch
4. Compute a local descriptor
5. Match local descriptors

\[ d(f_A, f_B) < T \]
Distance between descriptors

- Define distance function that compares two descriptors

  - $L_1$ distance (SAD):
    
    $$d(f_a, f_b) = \sum |f_a - f_b|$$

  - $L_2$ distance (SSD):
    
    $$d(f_a, f_b) = \sum (f_a - f_b)^2$$

  - Hamming distance:
    
    $$d(f_a, f_b) = \sum \text{XOR}(f_a, f_b)$$
At which threshold do we get a good match?

- The distance threshold affects performance
Evaluating matching performance

\[
\text{recall} = \frac{\text{# true positives}}{\text{# matching features (positives)}}
\]

\[
1 - \text{“precision”} = \frac{\text{# false positives}}{\text{# unmatched features (negatives)}}
\]

![Graph showing recall and precision](image)
Evaluating matching performance

ROC curve ("Receiver Operator Characteristic")

AUC: Area Under the Curve

# true positives
# matching features (positives)

"recall"

true positive rate

false positive rate

# false positives
# unmatched features (negatives)

1 - "precision"
Matching strategy

• Compare all

• Take the closest
  – Or $k$ closest
  – And/or within a (low) thresholded distance
Matching strategy

- Compare all
- Take the closest
  - Or $k$ closest
  - And/or within a (low) thresholded distance
- Choose the $N$ best putative matches
Which matches are good?

• Can get good scores for ambiguous or incorrect matches
Nearest Neighbour Distance Ratio

- For a descriptor $f_a$ in $I_a$, take the two closest descriptors $f_b^1$ and $f_b^2$ in $I_b$
- Perform ratio test: $d(f_a, f_b^1) / d(f_a, f_b^2)$
  - Low distance ratio: $f_b^1$ can be a good match
  - High distance ratio: $f_b^1$ can be an ambiguous or incorrect match
Nearest Neighbour Distance Ratio

- For a descriptor $f_a$ in $I_a$, take the two closest descriptors $f_b^1$ and $f_b^2$ in $I_b$
- Perform ratio test: $d(f_a, f_b^1) / d(f_a, f_b^2)$
  - Low distance ratio: $f_b^1$ can be a good match
  - High distance ratio: $f_b^1$ can be an ambiguous or incorrect match
Nearest Neighbour Distance Ratio

- For a descriptor $f_a$ in $I_a$, take the two closest descriptors $f_b^1$ and $f_b^2$ in $I_b$
- Perform ratio test: $\frac{d(f_a, f_b^1)}{d(f_a, f_b^2)}$
  - Low distance ratio: $f_b^1$ can be a good match
  - High distance ratio: $f_b^1$ can be an ambiguous or incorrect match

Threshold of 0.8 provides good separation

Example: Holmenkollen
Example: Holmenkollen
Cross check test

• Choose matches \((f_a, f_b)\) so that
  - \(f_b\) is the best match for \(f_a\) in \(I_b\)
  - And \(f_a\) is the best match for \(f_b\) in \(I_a\)

• Alternative to ratio test
Cross check test

- Choose matches \((f_a, f_b)\) so that
  - \(f_b\) is the best match for \(f_a\) in \(I_b\)
  - And \(f_a\) is the best match for \(f_b\) in \(I_a\)
- Alternative to ratio test
Matching algorithms

- Comparing all features works well for small sets of images
  - Brute force: BFMatcher in OpenCV

- When the number of features is large, an indexing structure is required
  - For example a k-d tree
  - Training an indexing structure takes time, but accelerates matching
  - FlannBasedMatcher in OpenCV
Summary

• Matching keypoints
  – Comparing local patches in canonical scale and orientation

• Feature descriptors
  – Robust, distinctive and efficient

• Descriptor types
  – HoG descriptors
  – Binary descriptors

• Putative matching
  – Closest match, distance ratio, cross check

• Next lecture
  – Matches that fit a model