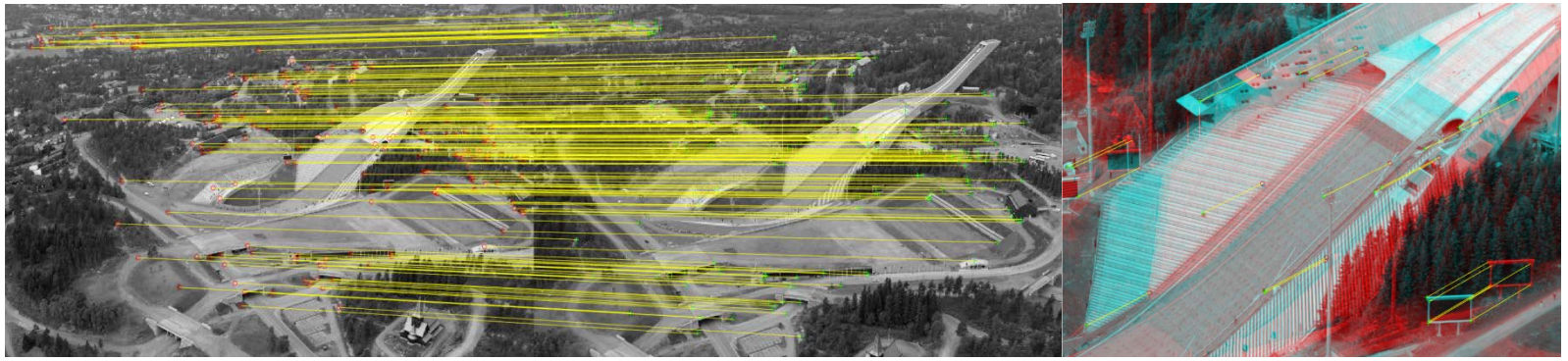
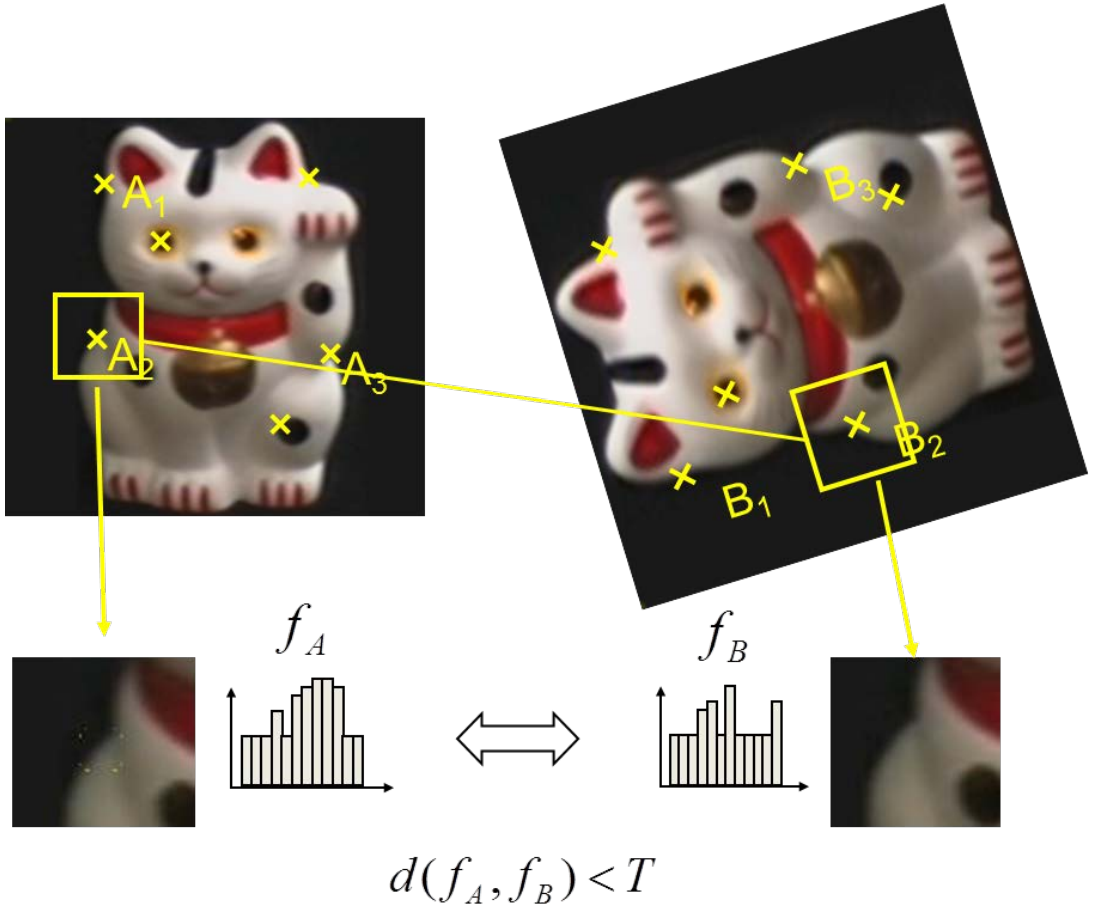


# Lecture 4.2
# Feature matching

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

# Distance between descriptors

- Define distance function that compares two descriptors
  - $L_1$ distance (SAD):

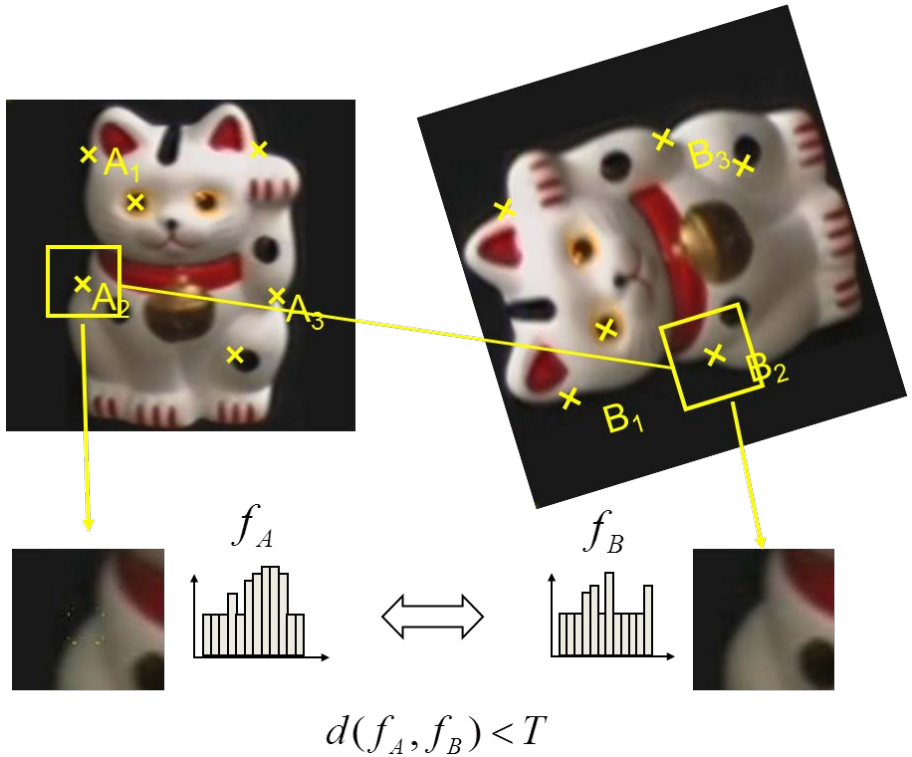
$$d(f_a, f_b) = \sum |f_a - f_b|$$

  - $L_2$ distance (SSD):

$$d(f_a, f_b) = \sum (f_a - f_b)^2$$

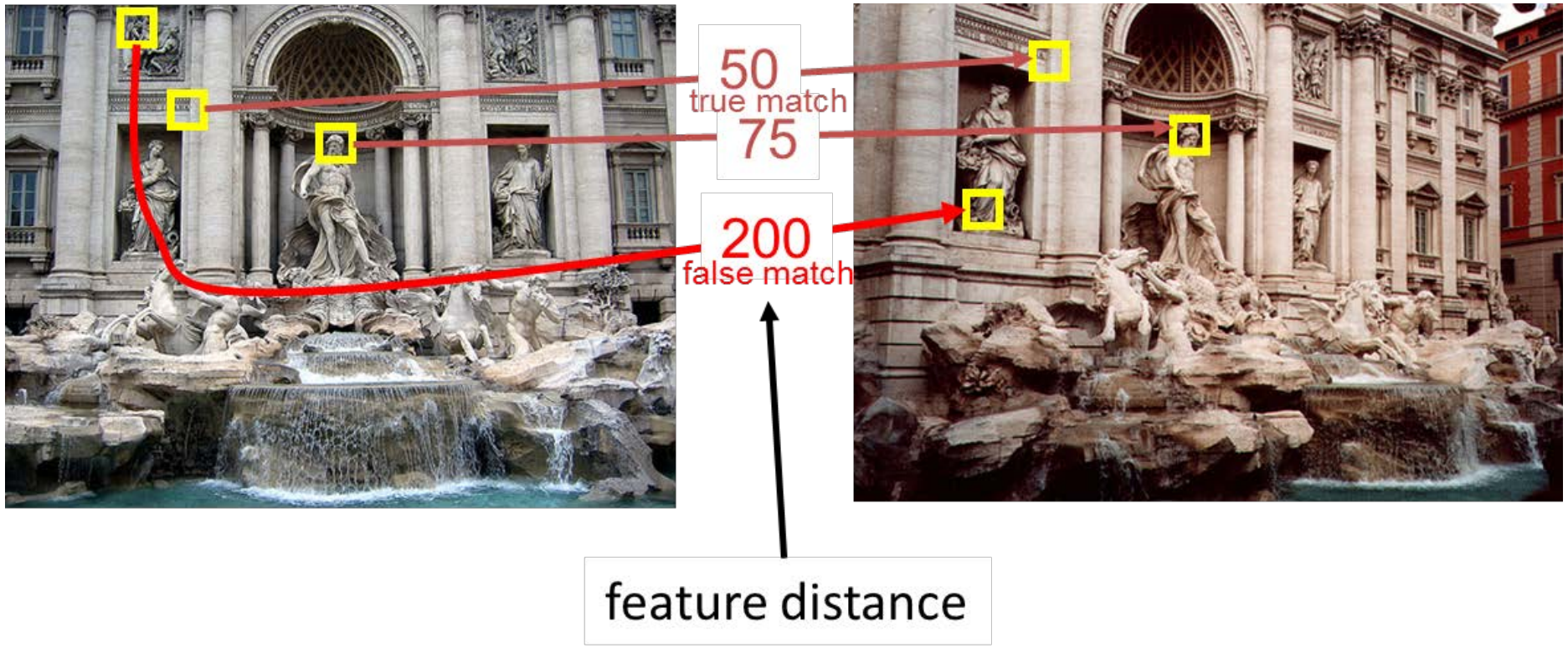
  - Hamming distance:

$$d(f_a, f_b) = \sum \mathrm{XOR}(f_a, f_b)$$



# At which threshold do we get a good match?

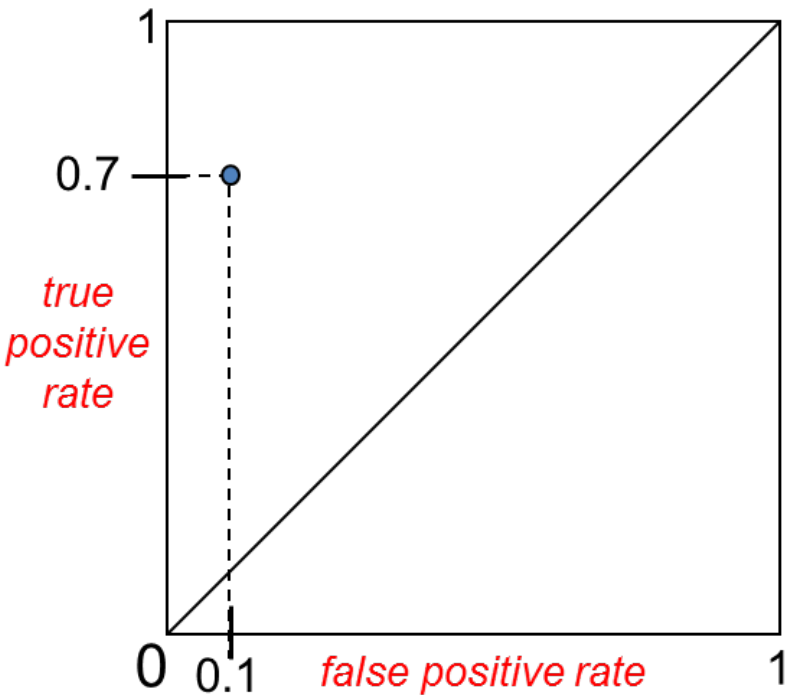
- The distance threshold affects performance



# Evaluating matching performance

$$\frac{\text{\# true positives}}{\text{\# matching features (positives)}}$$

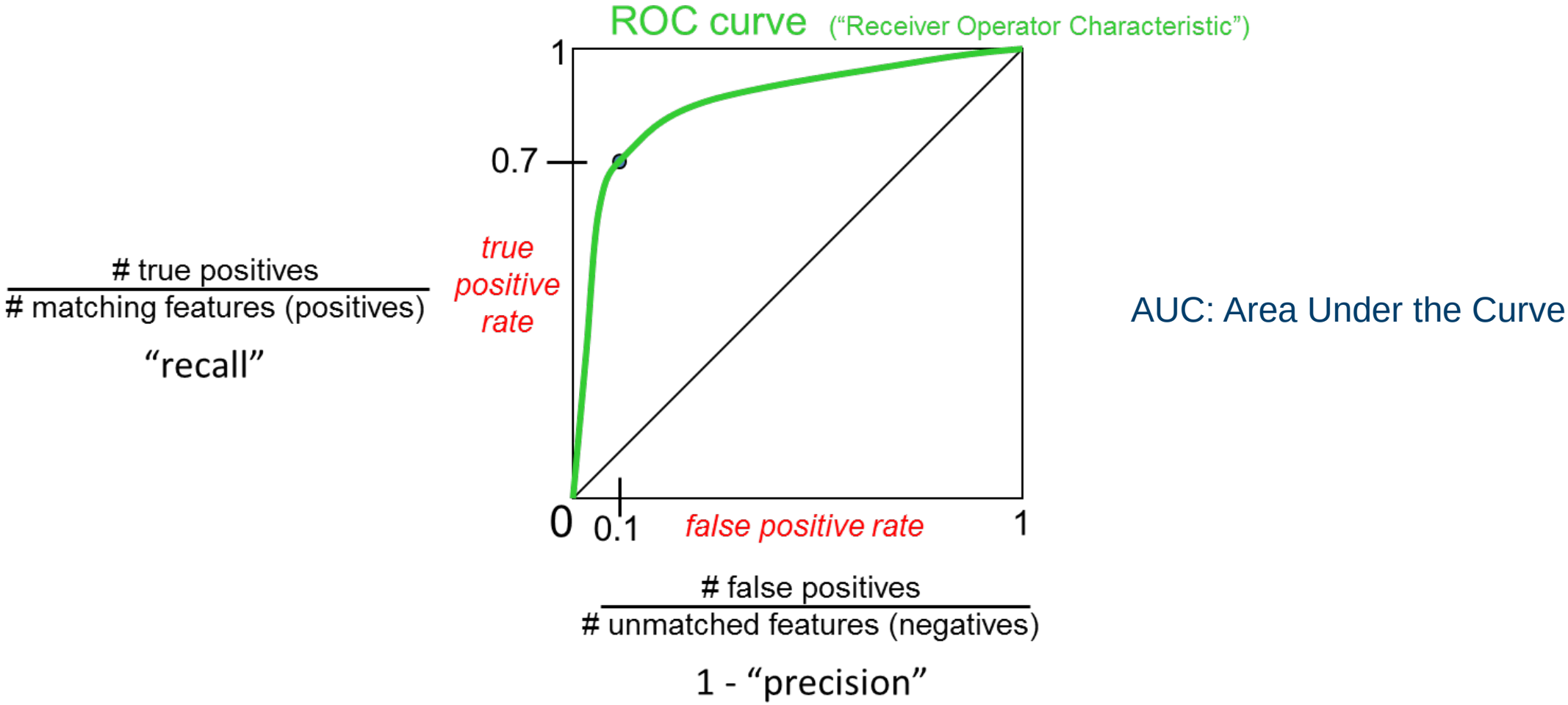
"recall"




$$\frac{\text{\# false positives}}{\text{\# unmatched features (negatives)}}$$

1 - "precision"

# Evaluating matching performance

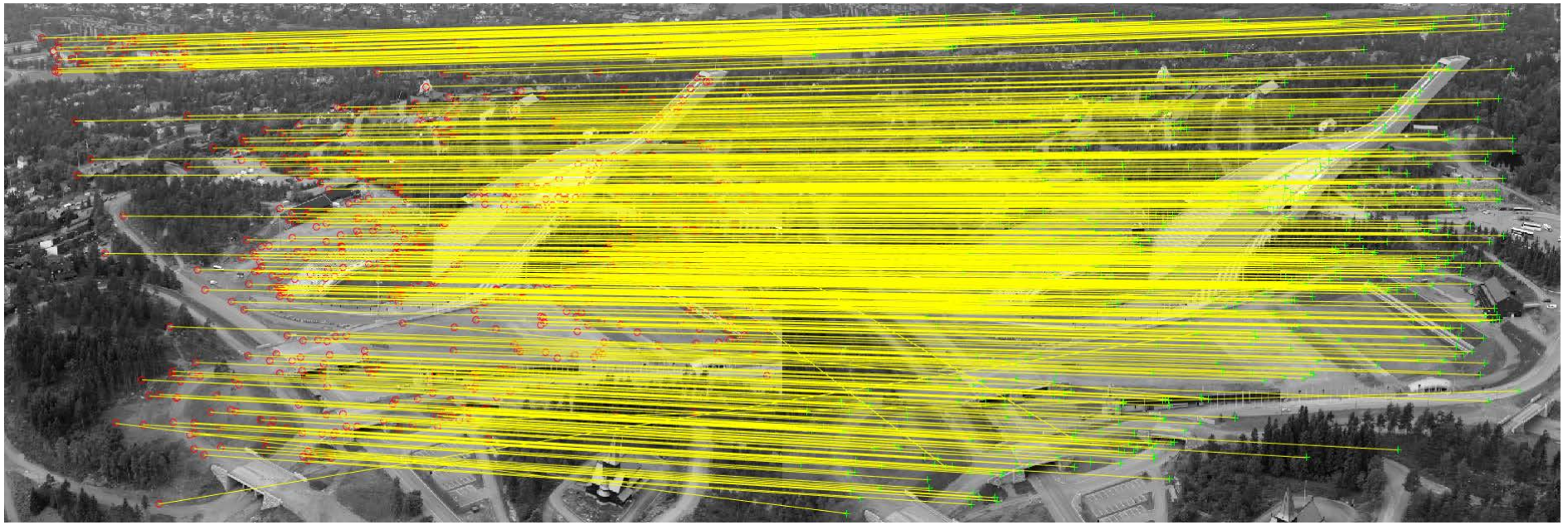



AUC: Area Under the Curve

# Matching strategy

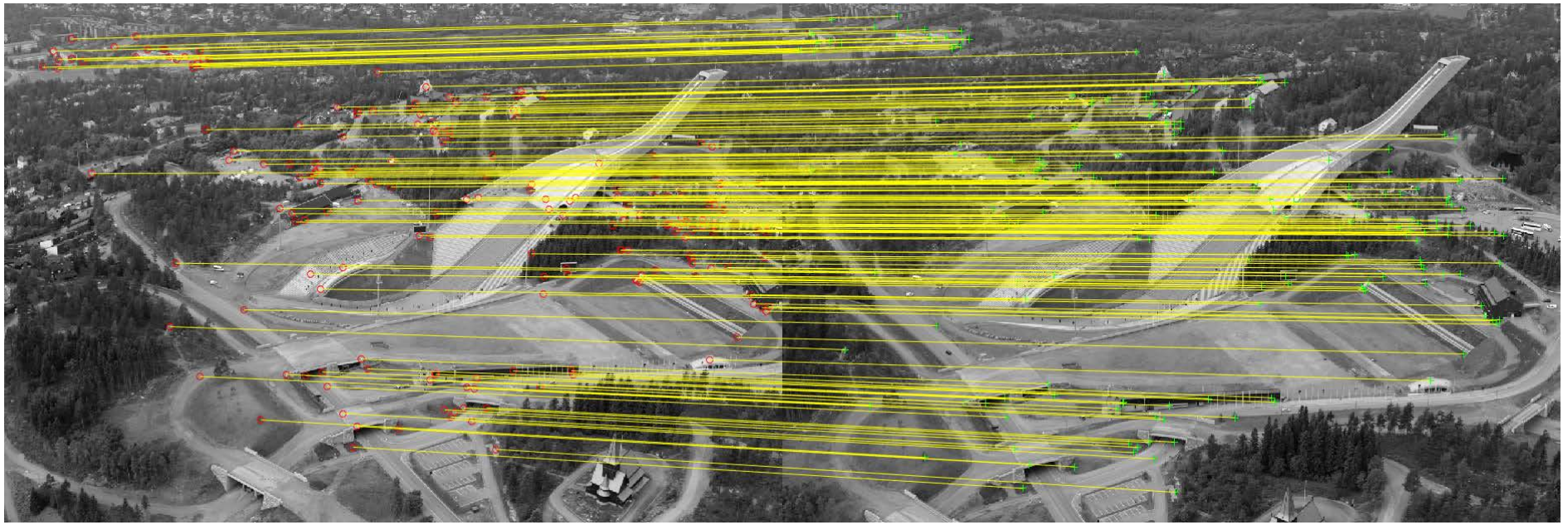
- Compare all

- Take the closest
  - Or *k* closest
  - And/or within a (low) thresholded distance



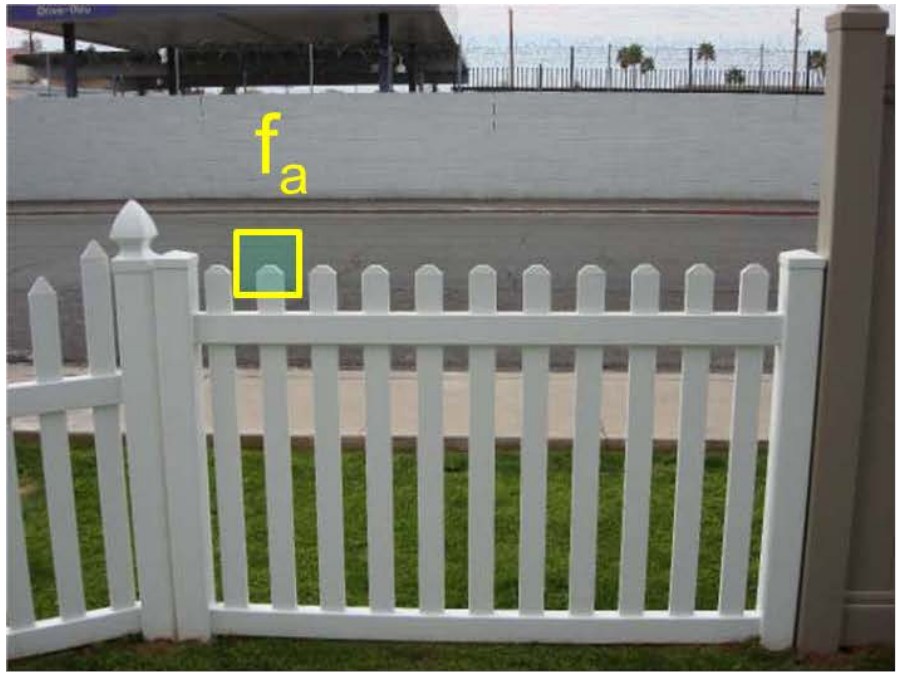
# Matching strategy

- Compare all
- Take the closest
  - Or *k* closest
  - And/or within a (low) thresholded distance
- Choose the *N* best *putative* matches

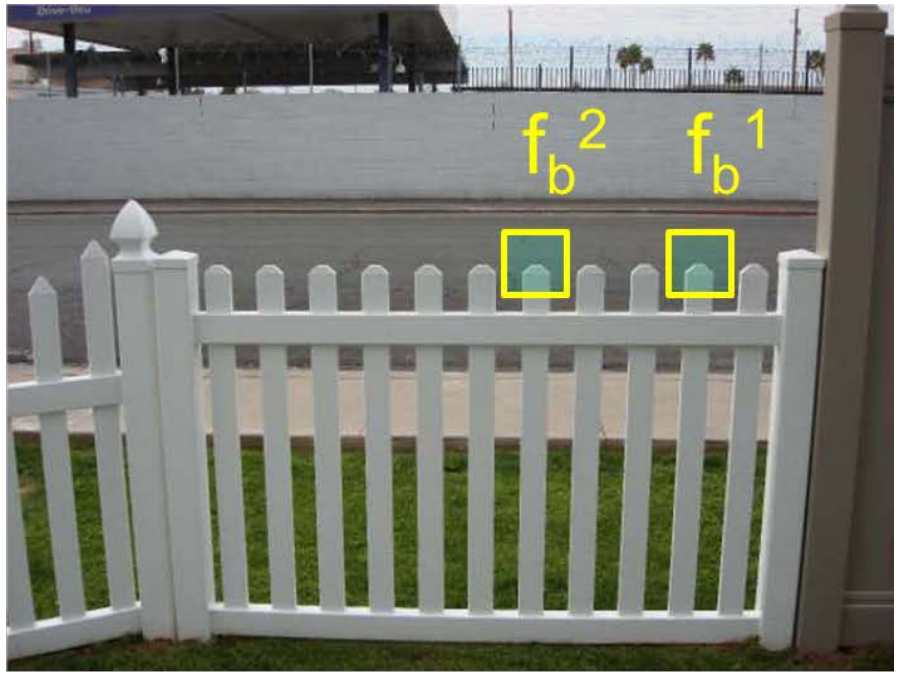


# Which matches are good?

- Can get good scores for ambiguous or incorrect matches



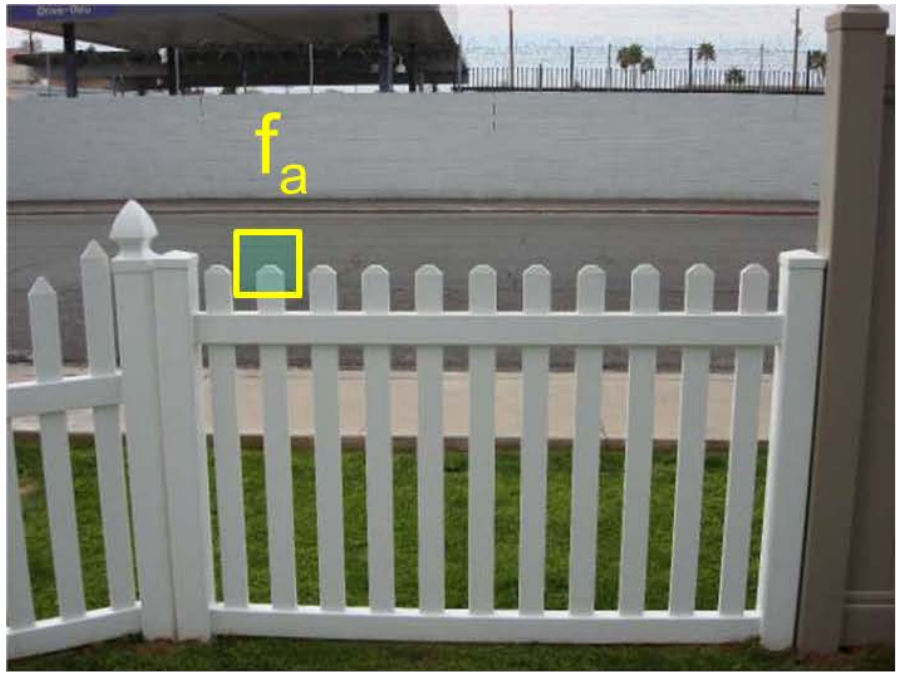

$I_a$



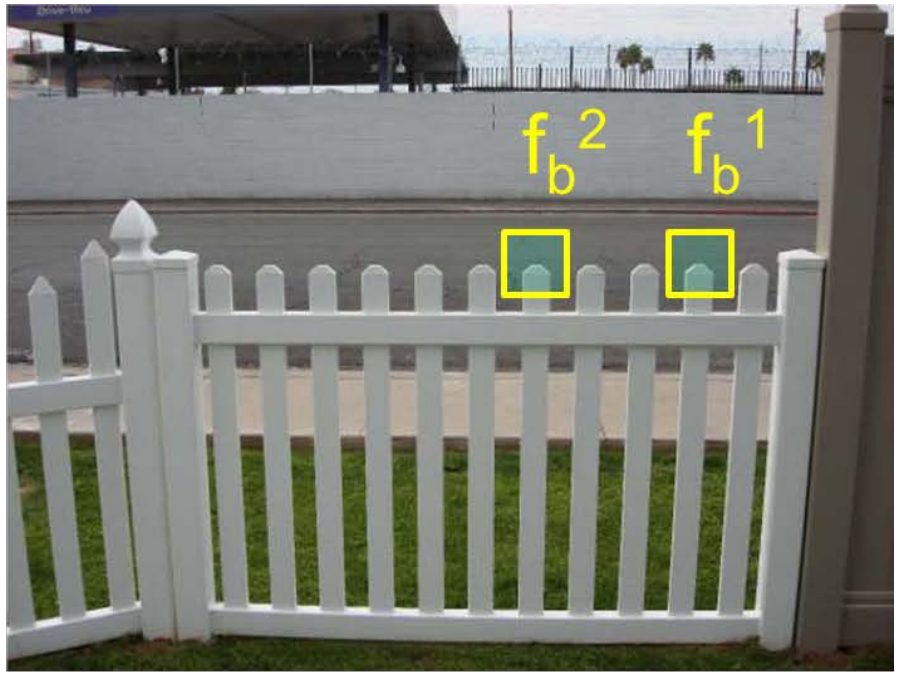

$I_b$

# 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




$I_a$



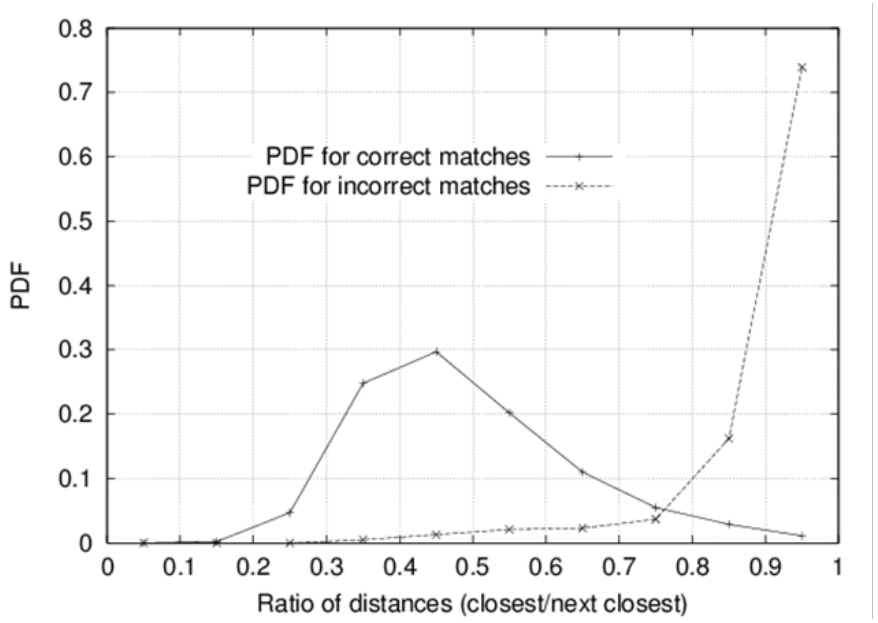

$I_b$

# 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

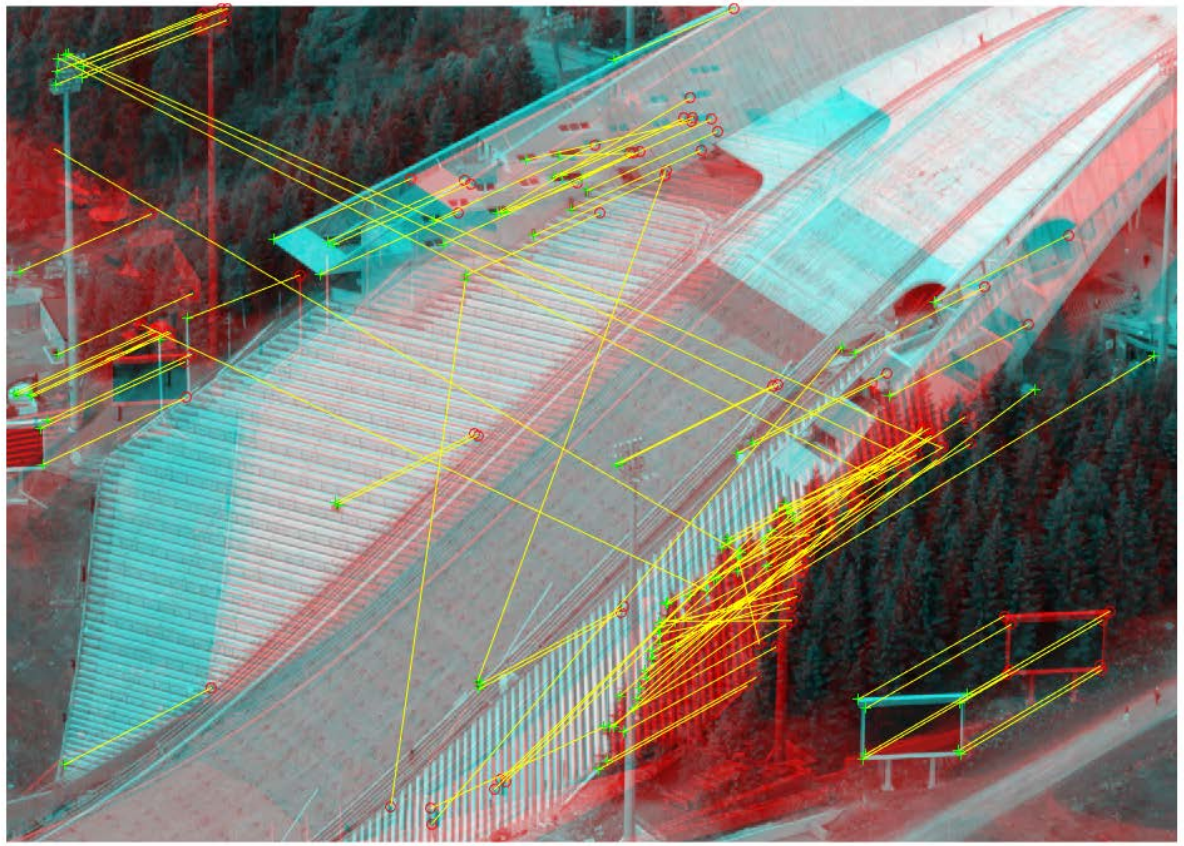
- For a descriptor $f_a$ in $I_a$, take the two closest descriptors $f_b^1$ and $f_b^2$ in $I_b$
- Perform ratio test: $d(f_a, f_b^1) \,/\, d(f_a, f_b^2)$
  - Low distance ratio: $f_b^1$ can be a good match
  - High distance ratio: $f_b^1$ can be an ambiguous or incorrect match



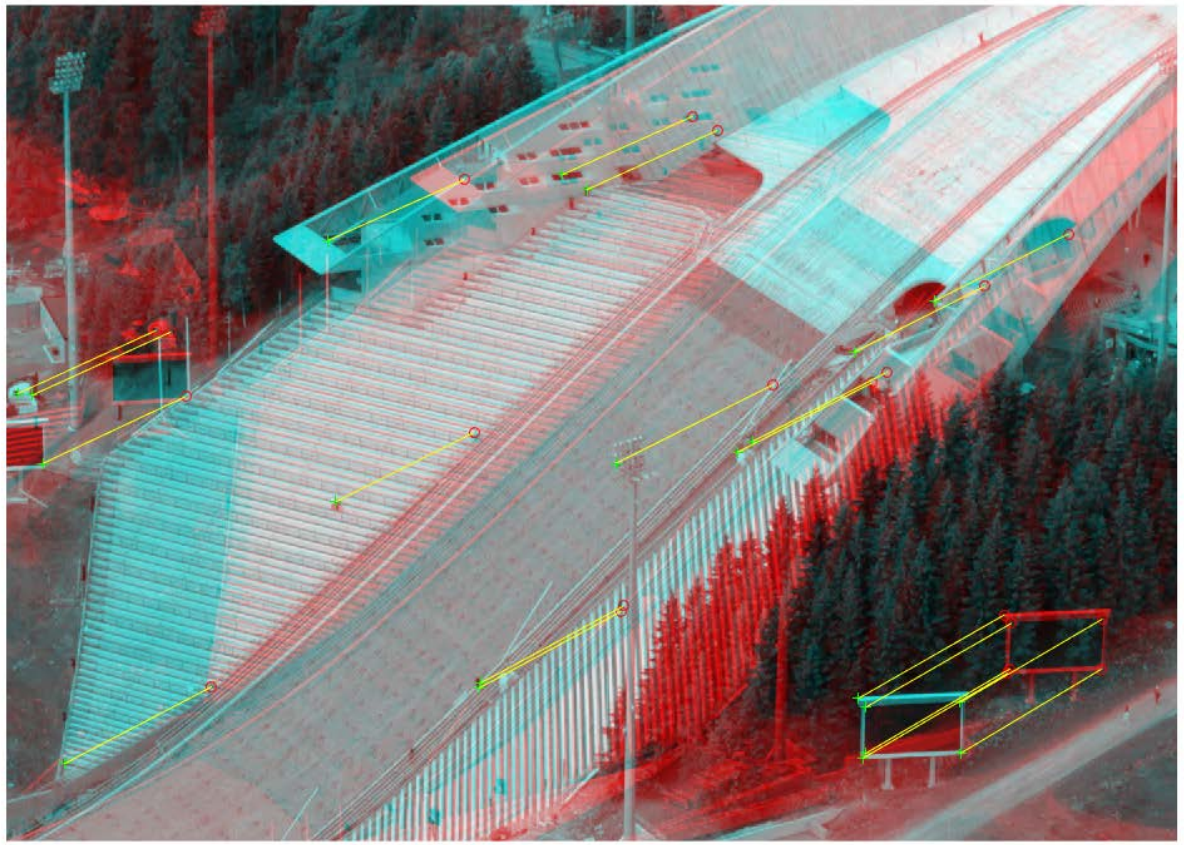

**Threshold of 0.8 provides good separation**

David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004

# Example: Holmenkollen

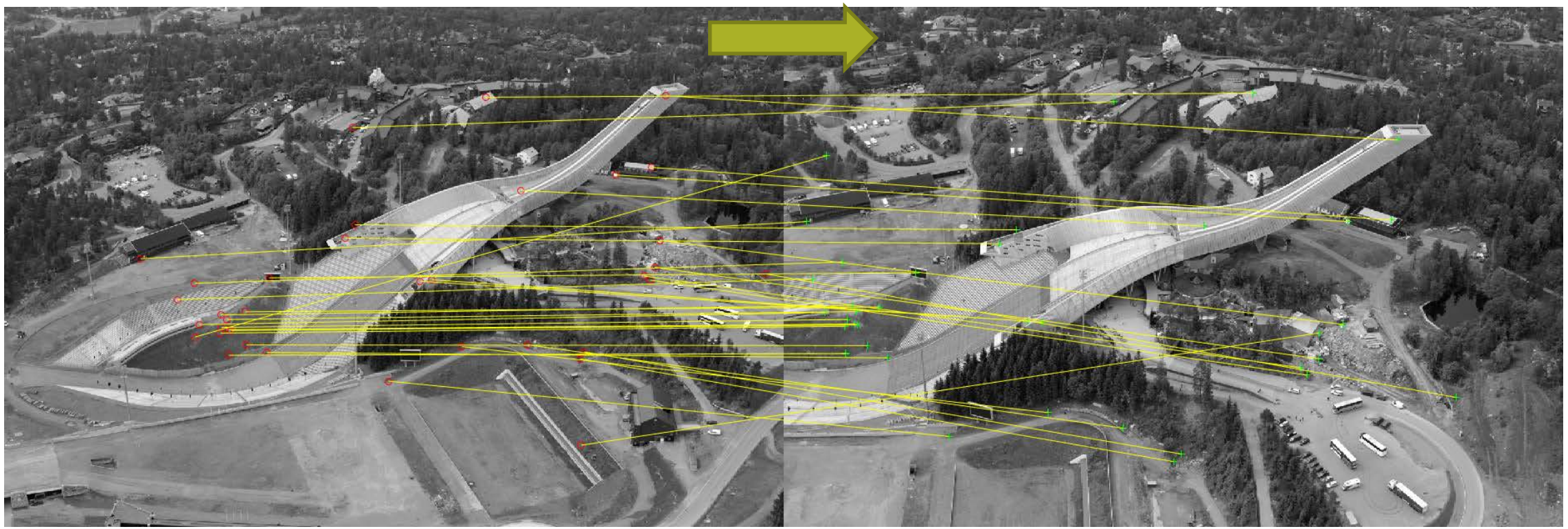


# Example: Holmenkollen



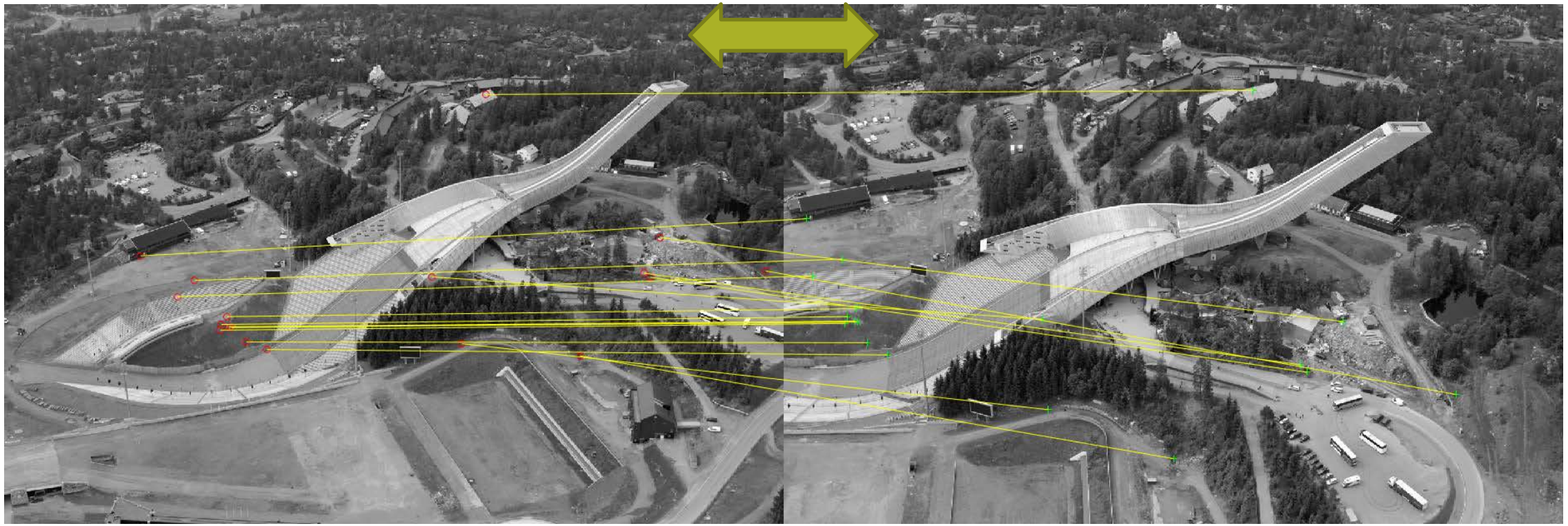
# Cross check test

- 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