COSC470 Feature Matching

COSC470: Special Topic
Computer Vision | 3D Reconstruction
Steven Mills

Feature Matching

- Once we have detected features in each image, we need to match them
- This is the *correspondence problem*, and is one of the biggest issues in computer vision
- We want to find pairs of features, one in each of two images, that correspond to the same real-world point
- We do this on the basis of *feature descriptors*

Feature Description

- We typically describe features with a vector
- This vector contains values extracted from a region of the image around the features
- Given two features, f_1 and f_2, with descriptors d_1 and d_2, we say they are similar if
$$\|d_2 - d_2\|$$
is small, for some distance function $\|x\|$
- Match features with smallest distance

Distance Functions

- There are many ways to measure $\|d_1 - d_2\|$
  - Usually use sum of squared distances (SSD)
  - Sum of absolute distances (SAD)
  - Normalised Cross Correlation (NCC)

$$\mathrm{SSD}(d_1, d_2) = \sum_{i=1}^{n} (d_{1,i} - d_{2,i})^2$$

$$\mathrm{SAD}(d_1, d_2) = \sum_{i=1}^{n} |d_{1,i} - d_{2,i}|$$

$$\mathrm{NCC}(d_1, d_2) = \frac{1}{n} \sum_{i=1}^{n} \frac{(d_{1,i} - \overline{d_1})(d_{2,i} - \overline{d_2})}{\sigma_1 \sigma_2}$$

n is the number of elements in each descriptor

$d_{1,i}$ is the ith element of d_1

$\overline{d_1}$ is the average of all the elements of d_1

σ_1 is the standard deviation of the elements of d_1

Image Patch Descriptors

- A simple descriptor is the image patch around a feature
- This can change with
  - Viewpoint/angle
  - Lighting
  - Distance/scale
- We want to find more *invariant* descriptors

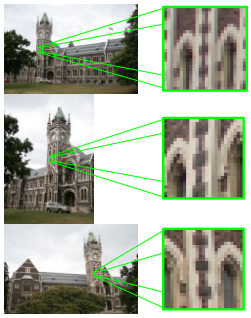
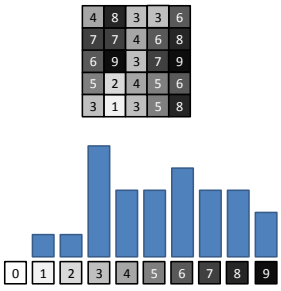


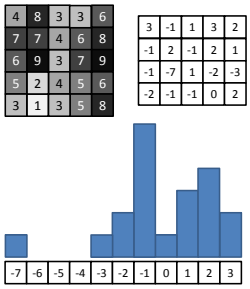
Image Histograms

- Invariant to
  - Translation
  - Rotation in the image
  - Other small rotations
- But not with respect to
  - Lighting (can used normalised histograms)
  - Scale (distance)
  - Big rotations out of the image plane



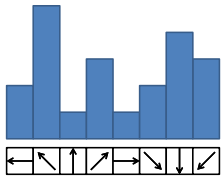
Gradient-Based Descriptors

- Image intensity varies a lot with lighting
- Gradients are usually more robust to this
- Can make a histogram of gradients as a more robust descriptor
- Use gradient direction/ magnitude/components



SIFT Descriptors

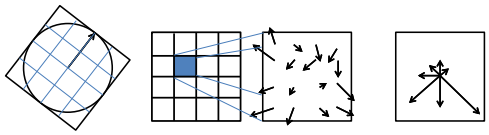
- SIFT uses DoG features
- These have a scale
- Can also estimate their orientation
  - Make a histogram of gradient directions
  - The biggest peak is the feature's orientation
  - Other peaks at least 80% as big are new features



SIFT Descriptors

- Can now find a patch around the feature
  - Aligned to orientation
  - Of the right scale
- Split this into a 4x4 grid
  - Make a histogram of gradients in each cell
  - This uses 8 bins – so 128 (4x4x8) values
- Normalise to fit in bytes
- The size of the cell depends on scale
  - Take a fixed number of samples
- The boundaries of the cell/feature patches may cause problems
  - Use centre-weighted averages

SIFT Descriptors



Detected Feature | Correct for orientation | Sample gradients in each cell | Gradient histogram for each cell

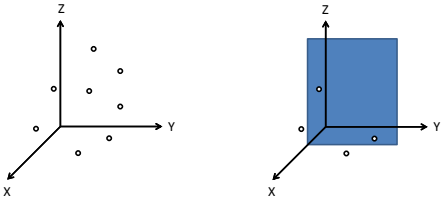
Matching Descriptors

- The descriptors are vectors, so subtract
- The length of this difference is low if the features are similar
- Usually use squared length as it is easier to compute
- Naïve matching is slow
- Suppose we have n features in each image, represented by vectors of length d
- $O(n^2)$ matches, each using $O(d)$ operations
- n = 10,000; d=128, gives $O(10^{10})$ operations

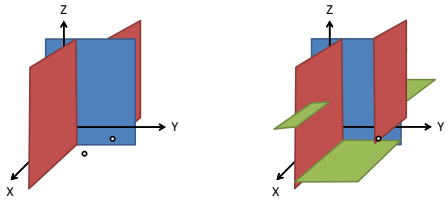
kd-Tree Based Matching

- It is n that gets big
- Use a tree-based search to find matches quickly
- These often give only *approximate* nearest neighbours for faster processing
- But matching becomes $O(n \log n)$, which is much better than $O(n^2)$
- k-d Trees often used:
  - A k-d tree is a binary tree for multi-dimensional data
  - Pick a dimension
  - Split around some point in that dimension to greater/less than halves
  - Repeat until leaves are small enough that splitting isn't needed

kd-Trees



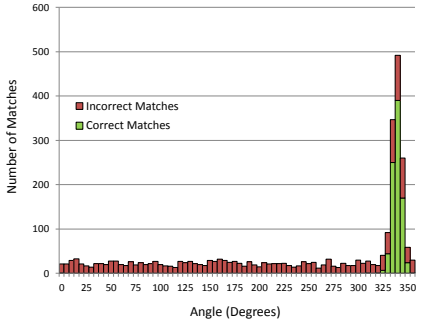
kd-Trees



Filtering Matches

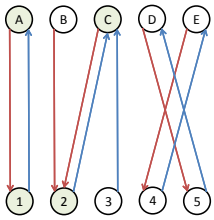
- Many raw matches from SIFT are wrong
- Lowe suggests finding the 2 nearest matches
- A match is only kept if the distance to the best match is less than 80% of the distance to the second-best
- Other heuristics can be used to filter matches
  - The *relative* scale and orientation of matches is often consistent
  - Can estimate the most common rotation/scale between raw matches
  - Then discard any that differ significantly

Relative Orientation of Matches



Forward-Reverse Matching

- Another common filter is to make sure that the match is consistent
- If you match image1's features to image2's, you should get the same result as matching image2 to image 1
- If not, it's not reliable



OpenCV Support - Detection

- A range of feature detectors
  - FAST, MSER, SIFT (DoG), GoodFeaturesToTrack
- A range of feature descriptors
  - SIFT, SURF, Orb, BRIEF
- And feature matchers
  - BruteForceMatcher, FlannBasedMatcher