Computer Vision Week 4

COMP9517
Image Alignment
Approaches

• Can we use brute force?
• Direct alignment (optical flow)
• Feature based matching
Brute Force

• The simplest approach is a brute force search
  – Need to define image matching function
    • SSD, normalized correlation, etc.
  – Search over all parameters within a reasonable range:

• e.g. for translation:
  for tx=x0:step:x1,
    for ty=y0:step:y1,
      compare image1(x,y) to image2(x+tx,y+ty)
    end;
  end;

• Need to pick correct \( x0, x1 \) and \( step \)
  – What happens if \( step \) is too large?
Brute Force

• What if we want to search for more complicated transformation, e.g. projective?

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

for a=a0:astep:a1,
  for b=b0:bstep:b1,
    for c=c0:cstep:c1,
      for d=d0:dstep:d1,
        for e=e0:estep:e1,
          for f=f0:fstep:f1,
            for g=g0:gstep:g1,
              for h=h0:hstep:h1,
                compare imagem1 to H(image2)
              end
          end
        end
      end
    end
  end
end
Problems with brute force

• Not realistic
  – Search in $O(N^8)$ is problematic
  – Not clear how to set starting/stopping value and step

• What can we do?
  – Use pyramid search to limit starting/stopping/step values
  – For special cases (rotational panoramas), can reduce search slightly to $O(N^4)$:
    • $H = K_1R_1R_2^{-1}K_2^{-1}$ (4 DOF: $f$ and rotation)

• Alternative: gradient decent on the error function
  – i.e. how do I tweak my current estimate to make the SSD error go down?
  – Can do sub-pixel accuracy
  – BIG assumption?
    • Images are already almost aligned (<2 pixels difference!)
    • Can improve with pyramid
  – Same tool as in **motion estimation**
Direct alignment: optical flow

Will start by estimating motion of each pixel separately
Then will consider motion of entire image
Why estimate optical flow?

• Many uses
  – Track object behavior
  – Correct for camera jitter (stabilization)
  – Align images (mosaics)
  – 3D shape reconstruction
  – Special effects
Problem definition: optical flow

• How to estimate pixel motion from image H to image I?
  • Solve pixel correspondence problem
    – given a pixel in H, look for nearby pixels of the same color in I

Key assumptions
  • color constancy: a point in H looks the same in I
    – For grayscale images, this is intensity constancy
  • small motion: points do not move very far
Feature-based alignment

- It turns out that a rather implausible approach works remarkably well:

  1. Detect ‘feature points’ in both images
Feature-based alignment

• It turns out that a rather implausible approach works remarkably well:

1. Detect ‘feature points’ in both images
2. Find correspondences
Feature-based alignment

• It turns out that a rather implausible approach works remarkably well:

1. Detect ‘feature points’ in both images
2. Find correspondences
3. Find a parametric transformation
Finding correspondences

• Generating potential matches: for each patch in one image, find a short list of patches in the other image that could match it based solely on appearance
  – Exhaustive search
    • For each feature in one image, compute the distance to all features in the other image and find the “closest” ones (threshold or fixed number of top matches)
  – Fast approximate nearest neighbor search
    • Hierarchical spatial data structures (kd-trees, vocabulary trees)

Assumption: use SSD distance between descriptors
Nearest-neighbor matching

• Solve following problem for all feature vectors, \( \mathbf{x} \):

\[
\forall j \; \text{NN}(j) = \arg \min_i ||\mathbf{x}_i - \mathbf{x}_j||, \; i \neq j
\]

• Nearest-neighbor matching is the major computational bottleneck
  – Linear search performs \( dn^2 \) operations for \( n \) features and \( d \) dimensions
  – No exact methods are faster than linear search for \( d>10 \)
  – Approximate methods can be much faster, but at the cost of missing some correct matches. Failure rate gets worse for large datasets.
Feature space outlier rejection

• How can we tell which potential matches are more reliable?
• Heuristic: compare distance of **nearest** neighbor to that of **second** nearest neighbor
  – Ratio will be high for features that are not distinctive
  – Threshold of 0.8 provides good separation

K-d tree construction

Simple 2D example

Slide credit: Anna Atramentov
K-d tree query
Kd-Tree failure example
Approximate k-d tree matching

Key idea:
- Search k-d tree bins in order of distance from query
- Requires use of a priority queue
- Copes better with high dimensionality
- Many different varieties
  - Ball tree, Spill tree etc.