Template Matching – Rigid Motion

• Find transformation to align two images.
• Focus on geometric features
  – (not so much interesting with intensity images)
  – Emphasis on tricks to make this efficient.

Problem Definition

• An Image is a set of 2D geometric features, along with positions.
• An Object is a set of 2D/3D geometric features, along with positions.
• A pose positions the object relative to the image.
  – 2D Translation; 2D translation + rotation; 2D translation, rotation and scale; planar or 3D object positioned in 3D with perspective or scaled orth.
• The best pose places the object features nearest the image features
Two parts to the problem

- Definition of cost function.
- Search method for finding best pose.
  1. Can phrase this as search among poses.
  2. Or as search among correspondences
  3. There are connections between two.

Example
Cost Function

• We look at this first, since it defines the problem.
• Again, no perfect measure;
  – Trade-offs between veracity of measure and computational considerations.
• One-to-one vs. many-to-one
• Bounded error vs. metric

Example: Chamfer Matching
Many-to-one, distance

For every edge point in the transformed object, compute the distance to the nearest image edge point. Sum distances.

\[ \sum_{i=1}^{n} \min(\| p_i, q_1 \|, \| p_i, q_2 \|, \ldots, \| p_i, q_m \|) \]
Main Feature:

- Every model point matches an image point.
- An image point can match 0, 1, or more model points.

Variations

- Sum a different distance
  - \( f(d) = d^2 \)
  - or Manhattan distance.
  - \( f(d) = 1 \) if \( d < \) threshold, 0 otherwise.
    - This is called bounded error.
- Use maximum distance instead of sum.
  - This is called: directed Hausdorff distance.
- Use other features
  - Corners.
  - Lines. Then position and angles of lines must be similar.
    - Model line may be subset of image line.
Other comparisons

• Enforce each image feature can match only one model feature.
• Enforce continuity, ordering along curves.
• These are more complex to optimize.

Pose Search: Standard Optimization Heuristics

• Brute force search with dense sampling.
• Random starting point + gradient descent.
  – Multiple random starting points
  – Stochastic gradient descent
• Any other optimization method you can think of.
Clever Idea 1: Chamfer Matching with the Distance Transform

Example: Each pixel has (Manhattan) distance to nearest edge pixel.

D.T. Adds Efficiency

- Compute once.
- Fast algorithms to compute it.
- Makes Chamfer Matching simple.
Then, try all translations of model edges. Add distances under each edge pixel.

That is, correlate edges with Distance Transform

```
 2 1 0 1 2 3 2
1 0 1 2 3 2 1
0 1 2 3 2 1 0
1 2 3 2 1 0 1
2 3 3 2 1 0 1
3 4 3 2 1 0 1
```

**Computing Distance Transform**

- It's only done once, per problem, not once per pose.
- Basically a shortest path problem.
- Simple solution passing through image once for each distance.
  - First pass mark edges 0.
  - Second, mark 1 anything next to 0, unless it's already marked. Etc….
- Actually, a more clever method requires 2 passes.
Chamfer Matching Complexity

• Brute force approach: for each pose, compare each model point to every image point. $O(pnm)$. $p =$ number poses, $n =$ number of image points, $m =$ number of model points.

• With distance transform: compute D.T., then for every pose, sum value under each model edge. $O(s + pm)$. $s =$ number of pixels, which is about same as $p$.

Clever Idea 2: Ransac

• Match enough features in model to features in image to determine pose.

• Examples:
  – match a point and determine translation.
  – match a corner and determine translation and rotation.
  – Points and translation, rotation, scaling?
  – Lines and rotation and translation?
Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
- $n$ — the smallest number of points required
- $k$ — the number of iterations required
- $t$ — the threshold used to identify a point that fits well
- $d$ — the number of nearby points required to assert a model fits well

Until $k$ iterations have occurred,
- Draw a sample of $n$ points from the data
- Uniformly and at random
- Fit to that set of $n$ points
- For each data point outside the sample
- Test the distance from the point to the line against $t$; if the distance from the point to the line is less than $t$, the point is close
- End

If there are $d$ or more points close to the line
then there is a good fit. Refit the line using all these points.
End

Use the best fit from this collection, using the fitting error as a criterion

(Forsyth & Ponce)
Complexity

- Suppose model has \( m \) points and image has \( n \) points. There are \( mn \) matches.
  
  When we match a model point, there is a \( 1/n \) probability this match is right.

  If we match \( k \) model points, probability all are right is approximately \( (1/n)^k \).

  If we repeat this \( L \) times, probability that at least one pose is right is:

\[
1 - \left( 1 - \left( \frac{1}{n} \right)^k \right)^L
\]

The Hough Transform for Lines

• A line is the set of points \((x, y)\) such that:
  \[ y = mx + b \]
  
• For any \((x, y)\) there is a line in \((m, b)\) space describing the lines through this point. Just let \((x, y)\) be constants and \(m, b\) be unknowns.

• Each point gets to vote for each line in the family; if there is a line that has lots of votes, that should be the line passing through the points.

Mechanics of the Hough transform

• Construct an array representing \(m, b\)
  
• For each point, render the line \(y=mx+b\) into this array, adding one at each cell
  
• Questions
  – how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)

• How many lines?
  – count the peaks in the Hough array

• Who belongs to which line?
  – tag the votes

• Can modify voting, peak finding to reflect noise.

• Big problem if noise in Hough space different from noise in image space.
Generalized Hough Transform

Some pros and cons

• Run-time for line finding
  – Complexity of RANSAC $n^3$
  – Complexity of Hough $n^d$
Error behavior

- Hough handles error with buckets. This gives a larger set of lines consistent with point, but ad-hoc.
- Ransac handles error with threshold. Well-motivated for error in other points, but not for error in first 2 points.
  - But works if we find some 2 points w/ low error.
- Error handling sloppy -> clutter bigger problem.
- Many variations to handle these issues.

Pose: Generalized Hough Transform

- Like Hough Transform, but for general shapes.
- Example: match one point to one point, and for every rotation of the object its translation is determined.
- Example: match enough features to determine pose, then vote for best pose.
Correspondence: Interpretation
Tree Search

- Represent all possible sets of matches as
  exponential sized tree.
- Each node involves another match
- Wildcard allowed for no matches.
- Prune tree when set of matches incompatible
  (this seems to imply bounded error).
- Trick: some fast way of evaluating
  compatibility.
- Trick: different tree search algorithms. Best
  first. A*....

2D Euclidean Transformation

- Check pairwise compatibility
  - Fast
  - Conservative test
Cass: Correspondence pose duality

• Suppose we match two features with bounded error.
  – There is a set of transformations that fit.
  – For nm matches, nm sets.
  – As these intersect, they carve \textit{transformation space} into regions.
    • Within a region, feasible matches are the same.
    • If sets are convex, \#regions is limited.
    • If everything is linear, this becomes easier.

Example: points, 2D translation
L-infinity norm
• Every cell is bounded by axial lines.
• Must contain point where two lines intersect.
  • No more than \((nm)^2\) cells.
• If we sample points where all pairs of lines intersect, we sample all cells.

General case

• Can extend to any linear transformation and convex, polygonal error bound.
• Every model point and every error line lead to hyperplane in transformation space.
• These divide transformation space into convex cells. Each has vertices at intersection of \(d\) hyperplanes.
• Complexity \((mn)^d\)
Can also mix and match

- Alignment, then tree search.
  - continue to add feasible additional matches.
- Greedy heuristics
- RANSAC + distance transform
- ...

Summary

- All these methods exponential in dimension of transformation.
- Clever & effective for translation, 2D euclidean.
- Too slow for 3D to 2D recognition
  - Grouping heuristics
    - Roberts – group quadrilaterals, then alignment.
    - Lowe – Perceptually salient grouping based on parallelism, proximity, ....