Global Search Algorithms

• Most model matching techniques are ‘local’
  – We assume we know roughly where the target is
  – Optimise some model parameters to match more accurately
  – Usually require good starting points
• This lecture will cover more ‘global’ approaches
  – Assume little prior knowledge about position
  – Good start points for ‘local’ search

Simple Feature Search

• We can rapidly search entire images for simple features, finding all examples of them
  – Edge points (e.g., use Sobel or Canny)
  – Peaks (point brighter than all neighbours)
  – Troughs (darker than all neighbours)
  – Circles (Hough transform)
  – Corners (see later lectures for details)

Composite Objects

• Most objects of interest are more complex
• Represent them as combinations of simple features
  – Need to model relationships between features
• Search becomes a ‘combinatoric’ problem
  – Which subset of all found features corresponds to the target?

Hypothesis and Test

• Consider looking for an object with the following shape

---

Hypothesis and Test

• Suppose output of circle finder is a set of \( n \) circles with centres and radii \( (x_i, r_i) \)
• Every pair \((i, j)\) gives one hypothesis
• Test radius:
  \[ |r_i - r_j| \leq \epsilon, \]
• Test separation:
  \[ \|x_i - x_j\| \leq 0.5(r_i + r_j) \leq \epsilon, \]

---
Hypothesis and Test

• Final verification can be obtained by projecting predicted shape into image and checking it lies on image edges
• Represent object outline with set of points and expected edge direction \([p, u_i]\)

Testing Hypotheses

• If outline represented as set of points and directions \([p, u_i]\)
• Quality of match to image given by
  \[ Q = \sum \mu_i G_i(p) + \alpha_i G_i(p) \]
  – Large values when image edges line up with model edges
• Apply a suitable threshold to decide if object actually present

Example

Correct pair
Wrong separation
No support for model outline
Circles different sizes

Representation

• Object is modelled as
  – Two circles with a given separation
  – A fixed shaped outline

Edge fragments (‘edgels’)

• Applying basic edge processing (e.g., Canny) gives us edge points
• Need to link together to form line segments (‘edgels’)
  – Some implementations of Canny do part of this when ‘edge following’
• These are useful features
  – Object outlines constructed from set of such edgels

Finding edge chains

• Given an image containing edge pixels
• Trace through to find chains of connected pixels
  – Scan through from corner until find an edge pixel
Finding edge chains

- Test each neighbour in turn to find next edge pixel
- Remove each found pixel from image (to avoid duplication)
- Use current direction as start direction when searching for next pixel
  - Or use direction associated with edge
- Re-start scan of whole image once reach the end of a chain (pixel has no neighbours)

Finding edge chains

- Re-start scan of whole image once reach the end of a chain (pixel has no neighbours)

Segmenting edge chains

- Break down each chain into a sequence of approximately straight sections: `edgels`

Representation with edgels

- Can work well for straight edged objects
  - Some manufactured things
  - Boxes etc
- Not so good for curved edges
  - Ambiguity as to how to split a curve into straight sections

Example: Finding a rectangle

- Suppose we attempt to find a rectangle in a cluttered image
  - Clutter in background leads to lots of spurious edgels
  - Noise can cause edges to break up

Rectangle finding

- We could test each edgel in turn as if it were one side of the rectangle
- If rectangle is of known size,
  - Each correctly matched edgel suggests
    1. an orientation
    2. A range of positions for the centre

Noise can cause edges to break up
Constraints imposed by an edgel

Centre constrained to lie on this line

Match this edgel to long side of rectangle

Constraints imposed by two edgels

Two (non-parallel) edgels uniquely define position

Match second edgel to short side of rectangle

Edge constraints

- Note that two edgels are insufficient to fully determine the rectangle if the scale can change

Require a point on a third edge to fully define position

General Approach

- Represent rigid object as
  1. A collection of features
     - Lines, Circles, Corners
  2. A set of relationships between the features
     - Relative positions, orientations, scales
  3. A complete description for verification
     - (eg a set of points and directions around the object outline)

General search

- Search for all edgels, circles, corners etc
- Associating each model features with an image feature applies some constraint to the position of the object
- Find the minimum number required to fully constrain the object
  - (eg 2 edgels for a rectangle of known size, 3 for one of varying size etc)

Combinatoric Search

- Suppose we have \( m \) model features and \( n \) image features of a particular type,
- There are \( m^n \) ways of choosing one image feature for each model feature
  - Eg 10,000 ways of choosing a pair of circles from 100 found in an image
- ‘Combinatorial Explosion’
- Can’t afford to do full test on all possible combinations
Tree search

- Exhaustive search can be expressed as a tree search
- Consider assigning every image feature to each model feature in turn

  Eg If four image features
  Model feature 1
  Model feature 2

Tree search

- Common approach is to use simple tests to discard false matches as soon as possible
  – If model feature 1 does not match image feature 1, don’t explore that branch at all

  Model feature 1
  Model feature 2

Extended features

- Complexity can be reduced if we use more information about each feature to allow us to rapidly discard bad examples
  – Eg build model of expected structures in vicinity – does the pattern of edges around a particular edgel match what we’d expect?

Occlusion: Missing features

- If features hidden, locating object harder
- Require a redundant model
  – Describe object with more than minimal number of features
  – Better chance that enough will be visible to find object

One end of rectangle completely hidden (occluded)

Tree-search allowing missing features

- During tree search, add extra ‘null’ option to allow for missing features

  Eg If four image features
  Model feature 1
  Model feature 2

Deformable Objects

- Techniques OK for rigid objects
- Deformable objects much harder
  – Need more features to pin them down
- Can use similar approach to give an approximate hypothesis to start local search
- Care required to avoid generating too many hypotheses