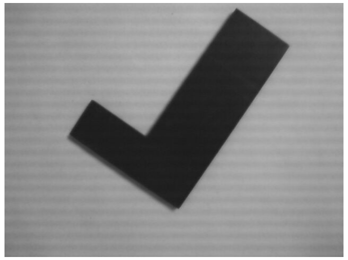
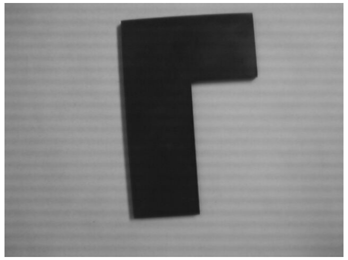
System 1 Overview

How to discriminate between these?
How to estimate object positions?



vs



Geometric model-based recognition

System 1 Overview

Geometric model-based recognition processes

Last Lecture: geometric description

This Lecture: model matching

pose estimation

Verification

Introduction

Given:

Sets of model lines $\{m_i\}$ in a scene coordinate system

Set of image lines $\{d_j\}$ in an image coordinate system

Image to scene scale conversion factor σ (pixels to cm)

Do:

1. Match image and model lines $\{(m_i, d_j)\}$
2. Estimate transformation mapping model onto data: R, $\vec{t}$
3. Verify matching and pose estimate

Output: identity and position (R, $\vec{t}$)

Interpretation Tree matching

Goal: Correspondence between subset of M model features $\{m_i\}$ and D data features $\{d_j\}$

Complete (exhaustive, depth-first) search - if a match exists, it will be found

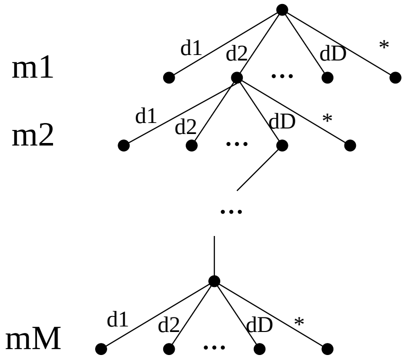
Needs a 'wildcard' ('*') data feature to match model features with no corresponding data feature (occlusion, segmentation failure)

Can find multiple solutions

Result: $\{(m_i, d_{j_i})\}$ set of matched features

Search Tree

Expand by model feature at each new level



Any given node in tree represents a set of matches $\{(m_i, d_{j_i})\}$

Reducing Search Complexity

Do we need to consider all paths in search tree?

No: Suppose current match state has these pairs matched: $\{(m_i, d_{j_i})\}, i = 1..k$

Given a new pair $(m_{k+1}, d_{j_{k+1}})$

1. $unary\_test(m_{k+1}, d_{j_{k+1}})$ - terminates extending search path if new pair has incompatible properties

2. $binary\_test(m_{k+1}, d_{j_{k+1}}, m_x, d_{j_x})$ for all $x = 1..k$ - terminates extending search path

if new pair has incompatible properties with each previous pairing on this tree branch (as all parts of the same object are compatible).

3. Early success limit L - can stop search when have $\{(m_i, d_{j_i})\}, i = 1..L$ compatible pairs

4. Early failure limit L - can stop search when can never get L pairs on this path. If have t non-wildcard matches on this path out of k pairings, then fail if $t + (M - k) < L$

Midlecture Problem

What are good unary/binary properties to test if matching parts with sets of circular holes? Eg:

Computational Complexity

M model feature tree levels. D data features on each level plus 1 wildcard
Worst case: $(D+1)^M$ nodes in tree to visit

p_u - probability that any random model feature and any random data feature pass unary_test

p_b - probability that any 2 random model features and any 2 random data features pass binary_test

Then, if $p_b MD < 2$, then the average case complexity of ITREE search is $O(LD^2)$

Much smaller, but can still be substantial

IT algorithm matlab code

```
% interpretation tree - match model and data lines until
% Limit are successfully paired or can never get Limit
% model - current model
% numM - number of lines in the model
% mlevel - last matched model feature
% Limit - early termination threshold
% pairs(:,2) - paired model-data features
% numpairs - number of paired features

function ok=itree(model,numM,mlevel,Limit,pairs,numpairs)

global Models numlines datalines

% check for termination conditions
if numpairs >= Limit     % enough pairs to verify
```



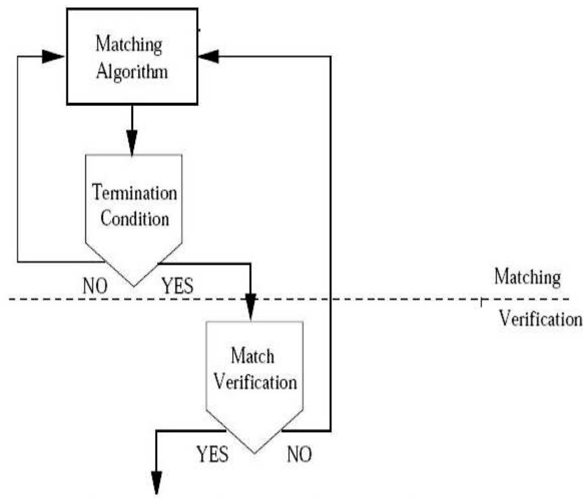
```
  [theta,trans] = estimatepose(model,numpairs,pairs)
  for p = 1 : 4
    ok = verifymatch(theta(p),trans(p,:)',model,
                numpairs,pairs);
    if ok
      return    % successful verification
    end
  end
  return        % failure to verify - continue search
end

% never enough pairs
if numpairs + numM - mlevel < Limit
  ok=0;
  return
end
```



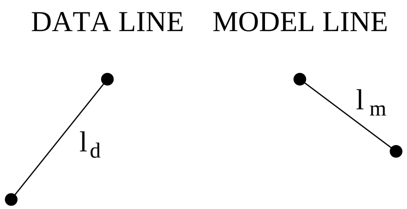
```
% normal case - see if we can extend pair list
mlevel = mlevel+1;
for d = 1 : numlines    % try all data lines

  % do unary test
  if unarytest(model,mlevel,d)

    % do all binary tests
    passed=1;
    for p = 1 : numpairs
      if ~binarytest(model,mlevel,d,pairs(p,1),pairs(p,2))
         passed=0;
         break
     end
    end
```

```
    if passed
      % passed all tests: add to matched pairs and recurse
      pairs(numpairs+1,1)=mlevel;
      pairs(numpairs+1,2)=d;
      ok=itree(model,numM,mlevel,Limit,pairs,numpairs+1);
      if ok
        return  % successful verification
      end
    end
  end
end

% wildcard case - go to next model feature
ok = itree(model,numM,mlevel,Limit,pairs,numpairs);
```


Algorithm Block Diagram



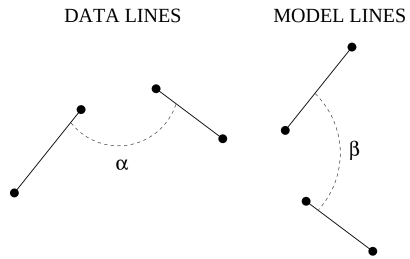
Line matching unary test



Pass test if $\sigma l_m(1-\delta_u) \leq l_d \leq \sigma l_m(1+\delta_u)$

Allows for calibration and segmentation errors
Position independent property
($\delta_u = 0.3$ typical)

Line matching binary tests



Pass test if $\mid \alpha - \beta \mid \leq \delta_b$

Allows for calibration and segmentation errors
Position independent property
($\delta_b = 0.2$ radians typical)

Also: don't allow duplicate use of model or data lines

Matching performance

Limit L = number of model lines - 1
Tries all models
Stops at first verified model instance for each model

Different Matched Models & Instances

Image	True Model	Tee	Thin L	Thick L
1	Tee	4	0	12
2	Tee	4	0	12
3	Tee	21	0	12
4	Tee	21	0	12
5	Thin L	0	15	2
6	Thin L	0	15	2
7	Thin L	0	15	2
8	Thin L	0	24	2
9	Thick L	0	2	3
10	Thick L	0	2	3
11	Thick L	0	2	3
12	Thick L	0	2	3

Pose Estimation

Goal: eliminate invalid matches & find object pose

Given a set $\{(m_i, d_{j_i})\}, i = 1..L$ of compatible pairs

Find the rotation $\mathbf{R}$ and translation $\vec{t}$ that transforms the model onto the data features.

This is the 'pose' or 'position'

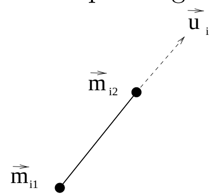
Let $\mathbf{R} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$ be the rotation matrix

If $\vec{p}$ is a model point, then $\mathbf{R}\vec{p} + \vec{t}$ is the transformed model point

Usually estimate rotation $\mathbf{R}$ first and then translation $\vec{t}$

Estimating Rotation

Given model line i endpoints $\{(\vec{m}_{i1}, \vec{m}_{i2})\}$
Corresponding data line endpoints $\{(\vec{d}_{i1}, \vec{d}_{i2})\}$



Model line direction unit vector:

$$\vec{u}_i = \frac{\vec{m}_{i2} - \vec{m}_{i1}}{\|\, \vec{m}_{i2} - \vec{m}_{i1} \,\|}$$

Data line direction unit vector:

$$\vec{v}_i = \frac{\vec{d}_{i2} - \vec{d}_{i1}}{\|\, \vec{d}_{i2} - \vec{d}_{i1} \,\|}$$

If no data errors, want **R** such that

$$\vec{v}_i = \pm \mathrm{R}\vec{u}_i$$

($\pm$ as don't know if endpoints are in same order)

But, as we have errors $\rightarrow$ least squares solution

Step 1: compute vectors perpendicular to $\vec{v}_i$
If $\vec{v}_i = (v_{x1}, v_{y1})$, then perpendicular is $(-v_{yi}, v_{xi})$

Step 2: compute error between $\vec{v}_i$ and $\mathrm{R}\vec{u}_i$
Use dot product of $\mathrm{R}\vec{u}_i$ and perpendicular, which equals sin() of angular error, which is small, so sin(error) $\doteq$ error

$$\epsilon_i = (-v_{yi}, v_{xi})\mathbf{R}(u_{xi}, u_{yi})'$$

Step 3: Reformulate error

Let $\mathbf{R} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$

Multiplying out and grouping terms:

$$\epsilon_i = (v_{xi}u_{yi} - v_{yi}u_{xi}, -v_{yi}u_{yi} - v_{xi}u_{xi})(cos(\theta), sin(\theta))'$$

Make a matrix equation

$$\vec{\epsilon} = \mathrm{D}(cos(\theta), sin(\theta))'$$

Each row of L vector $\vec{\epsilon}$ is ϵ_i and each row of $L \times 2$ matrix $\mathbf{D}$ is $(v_{xi}u_{yi} - v_{yi}u_{xi}, -v_{yi}u_{yi} - v_{xi}u_{xi})$

The least square error is $\vec{\epsilon}'\vec{\epsilon} = (cos(\theta), sin(\theta))\mathbf{D}'\mathbf{D}(cos(\theta), sin(\theta))'$

Step 4: Finding rotation that minimizes least square error

Let $\mathbf{D}'\mathbf{D} = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$

Then, we minimize $(cos(\theta), sin(\theta)) \begin{bmatrix} e & f \\ g & h \end{bmatrix} (cos(\theta), sin(\theta))' =$
$ecos(\theta)^2 + (f+g)cos(\theta)sin(\theta) + hsin(\theta)^2$

Differentiate wrt θ and set equal to 0 gives:

$$(f+g)cos(\theta)^2 + 2(h-e)cos(\theta)sin(\theta) - (f+g)sin(\theta)^2 = 0$$

Divide by $-cos(\theta)^2$ (if $cos(\theta) = 0$ then use special case) gives:

$$(f+g)tan(\theta)^2 + 2(e-h)tan(\theta) - (f+g) = 0$$

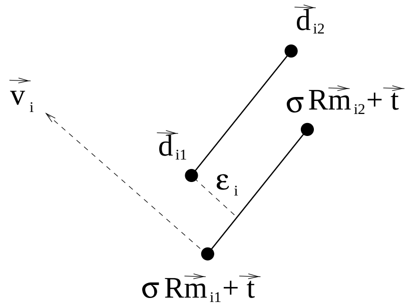
Solving gives:

$$tan(\theta) = \frac{(h-e) \pm \sqrt{(e-h)^2 + (f+g)^2}}{(f+g)}$$

Four θ solutions (2 for $\pm$, 2 for $tan(\theta) = tan(\pi + \theta)$).

Try to verify all 4.

Estimating Translation By Least Squares



$\vec{v}_i$ is perpendicular to rotated model line i

Offset error $\epsilon_i = (\vec{d}_{i1} - \sigma \mathbf{R} \vec{m}_{i1} - \vec{t})' \vec{v}_i$

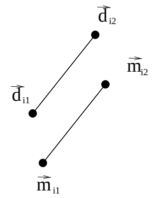
Differentiate $\sum_i \epsilon_i^2$ wrt $\vec{t}$, set equal to $\vec{0}$ and solve for $\vec{t}$ gives:

$$\vec{t} = (\sum \vec{v}_i \vec{v}_i')^{-1} \sum \vec{v}_i \vec{v}_i' (d_{i1} - \sigma \mathbf{R} \vec{m}_{i1})$$

Verification

Transform model lines into place: for each $\vec{m}_i$ compute $\sigma \mathrm{R} \vec{m}_i + \vec{t}$

For each model-data line pair, do 3 tests:
Test 1: Are model and data lines parallel?
(For simplicity, use $\vec{m}_i$ in notation instead of $\sigma \mathrm{R} \vec{m}_i + \vec{t}$)

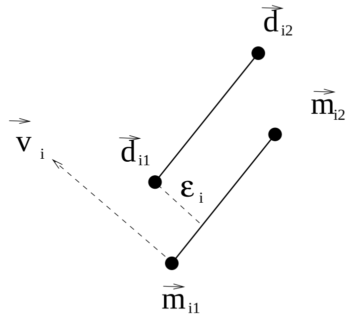


If

$$\left| \frac{\vec{m}_{i1} - \vec{m}_{i2}}{\| \vec{m}_{i1} - \vec{m}_{i2} \|} \cdot \frac{\vec{d}_{i1} - \vec{d}_{i2}}{\| \vec{d}_{i1} - \vec{d}_{i2} \|} \right| > threshold$$

then OK (threshold = 0.9?)

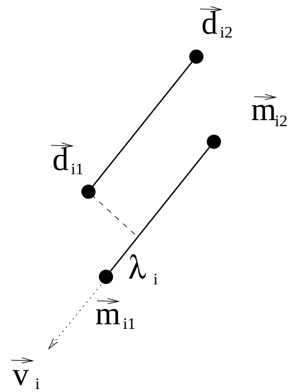
Test 2: Are model and data lines close?



Let $(r, s) = \frac{\vec{m}_{i1} - \vec{m}_{i2}}{\|\vec{m}_{i1} - \vec{m}_{i2}\|}$ and $\vec{v}_i = (-s, r)$

For $k = i1, i2$, compute $\epsilon_i = (\vec{d}_k - \vec{m}_{i1})' \vec{v}_i$
If $| \epsilon_i | < threshold$ then OK (threshold = 15 pixels?)

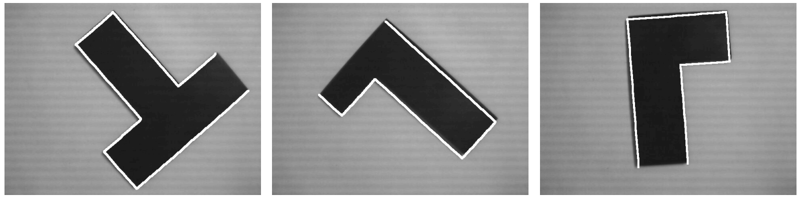
Test 3: Do model and data lines overlap?



For $k = i1, i2$, compute $\lambda_k = (\vec{d}_k - \vec{m}_{i1})' \vec{v}_i$

If $-tolerance \| \vec{m}_{i1} - \vec{m}_{i2} \| \leq \lambda_k \leq (1 + tolerance) \| \vec{m}_{i1} - \vec{m}_{i2} \|$,
then OK (tolerance = 0.3?)

Verified Position Result Examples



Limit = number of model lines - 1

School of Informatics, University of Edinburgh

Confusion Matrix

	Est Tee	Est Thin L	Est Thick L	No Est
True Tee	4	0	0	0
True Thin L	0	3	0	1
True Thick L	0	0	4	0

Image 8 had Thin L model flipped over.
Matching process can be extended to allow this.

School of Informatics, University of Edinburgh

Discussion

- Efficient if good unary/binary tests
- Suitable for 50% (estimated) flat parts
- Similar techniques for shapes other than straight lines: circular arcs, corners, holes, ...
- Extendable to 3D (future lectures)
- Extensions for perspective projection

School of Informatics, University of Edinburgh

What Have We Learned?

Introduction to

- Geometric Model-based Object Recognition
- General Feature Matching Algorithm
- 2D Least Squares rotation and translation estimation algorithms
- 2D Geometric Verification Algorithm