Interpretation Tree Model Matching

Robert B. Fisher
School of Informatics
University of Edinburgh

Interpretation Tree Model Matching

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 ? 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) ? - if a match exists, it will be found

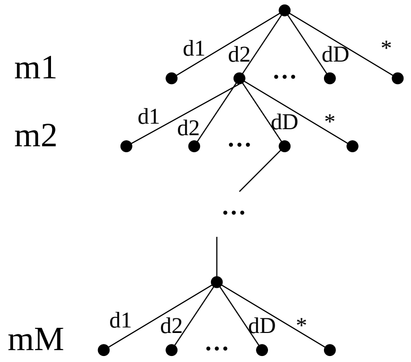
Needs a ‘wildcard’ (‘*’) data feature to match model features with no corresponding data feature (occlusion, segmentation failure)

Can find multiple solutions

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

Search Tree

Expand by ? 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 $\boxed{?\qquad}$ 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 $\boxed{?\qquad}$ 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 $t + (M - k) < L$

Computational Complexity

M model feature tree levels. D data features on each level plus 1 wildcard
$\boxed{?\qquad}$ 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

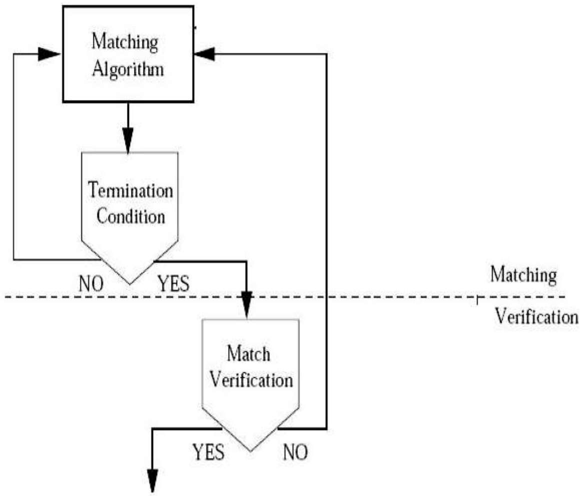
```
% interpretation tree - match model and data lines until
% Limit are successfully paired or can never get Limit
% model - current model
% numM - number of lines in the model
% mlevel - last matched model feature
% Limit - early termination threshold
% pairs(:,2) - paired model-data features
% numpairs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

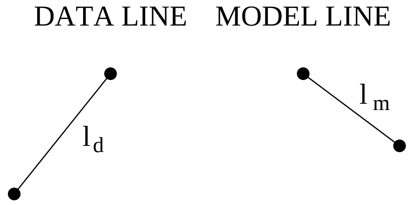
```
% normal case - see if we can extend pair list
mlevel = mlevel+1;
for d = 1 : numlines    % try all data lines

  % 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



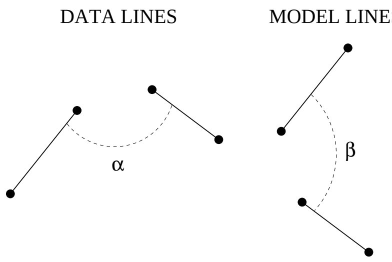
Line matching unary test



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 $\boxed{?}$ property
($\delta_u = 0.3$ typical)

Line matching $\boxed{?}$ 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 $\boxed{?}$ 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

What Have We Learned?

Introduction to

- General $\boxed{?\qquad\qquad}$ Matching Algorithm
- Efficient if good unary/binary tests
- Suitable for 50% (estimated) flat parts