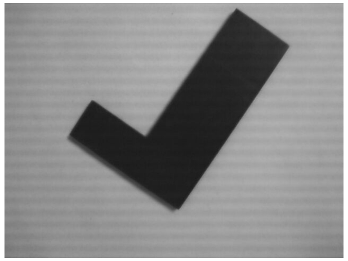
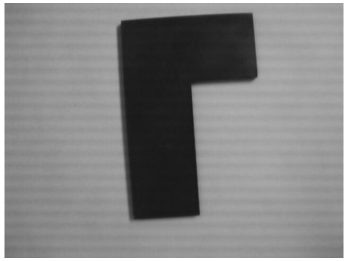
System 1 Overview

How to discriminate between and also estimate image positions?



vs



Geometric Model-based Object Recognition

System 1 Overview

Geometric Model-based Object Recognition

This Lecture: Geometric description

Next Lecture: Model matching

Pose estimation

Verification

Motivation - automated visual inspection

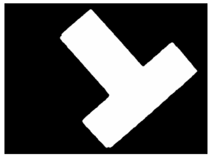
Manufacturing

- High speed product verification
- Largest use of computer vision systems worldwide
- Most western manufacturing has some visual quality control

Introduction

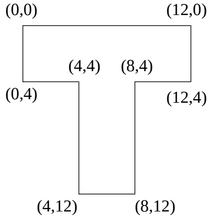
Given:

Isolated binary image object

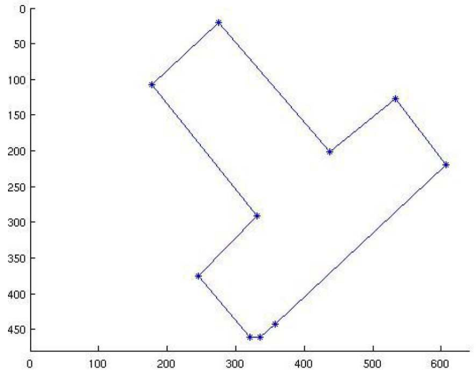


Assume:

1. Geometric shape models for parts to be recognized



2. Image feature positions



System does:

1. Matches image and model features
2. Estimates transformation mapping model onto data

Data Description

Goal: describe parts in same vocabulary of boundary shapes as model

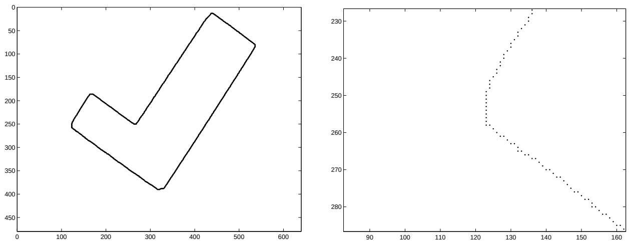
- Get object pixels that lie on the boundary
- Split pixels into straight line sets
- Find corners where the lines meet

(Here we ignore curved boundaries.)

Boundary Finding

1) Get points that lie on boundary:

```
[r,c] = find( bwperim(Image,4) == 1 )
```

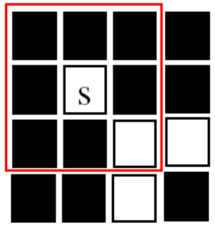


2) Remove any spurs on boundary, track and segment

```
[sr,sc] = removespurs(r,c,H,W);
[tr,tc] = boundarytrack(sr,sc);
[cr,cc] = findcorners(tr,tc);
```

Removing Dangling Spurs

Spur: any boundary pixel with only 1 neighbor inside a 3x3 neighborhood



```
changed=1;
while changed==1
  changed = 0;
  [sr,sc] = find(work==1);    % work: boundary pixels
  for i = 1 : length(sr)      % check each boundary point
    neigh = work(sr(i)-1:sr(i)+1,sc(i)-1:sc(i)+1);
    count=sum(sum(neigh));
    if count < 3              % only point and at most
      work(sr(i),sc(i)) = 0;  % 1 neighbor so remove it
      changed=1;
```

Trailing `ends` omitted.

Removing Unnecessary Boundary Pixels

Find unnecessary corners:

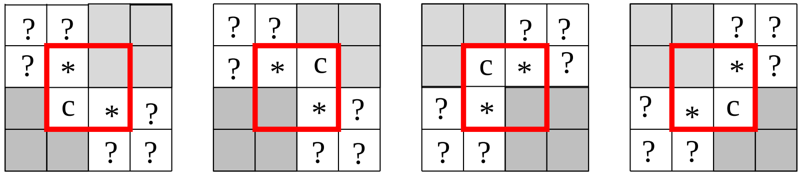
* - boundary point to keep

c - boundary point to remove

? - boundary point thru here somehow

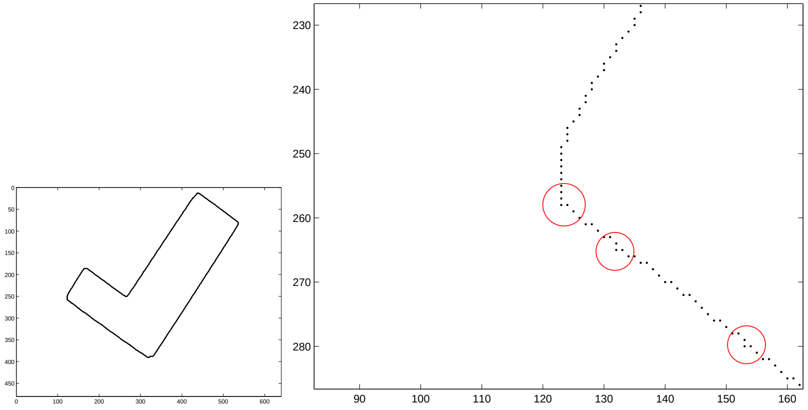
shaded box - interior or exterior pixel

thick red box - pixel neighbourhood inspected

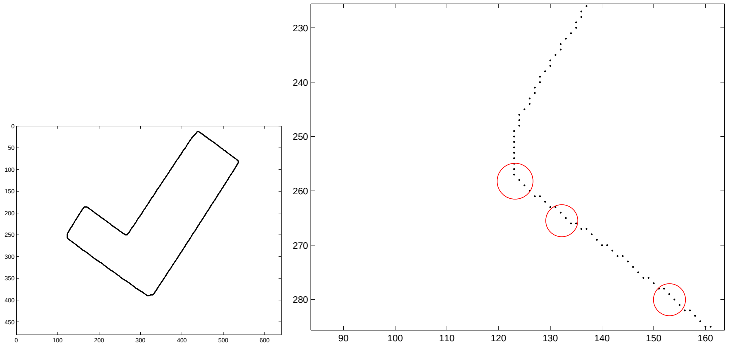


Boundary Cleaning Results

Raw boundary:



Cleaned boundary:



Getting a Consecutive Boundary Track

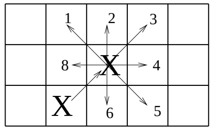
TRACK TO FIRST
UNTRACKED BOUNDARY
PIXEL ENCOUNTERED
AS i GOES 1...7

NEXT DIRECTIONS

1	2	3
8	x	4
7	6	5

NEXT = (LAST + 3 + i) MOD 8 + 1

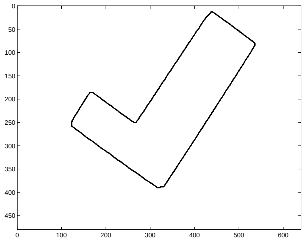
EXAMPLE TRACKING



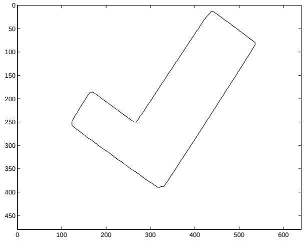
LAST MOVE = 3
NEXT MOVE = 8,1,2,3,4,5,6

Tracking Results

Despurred boundary (unorganized point set):

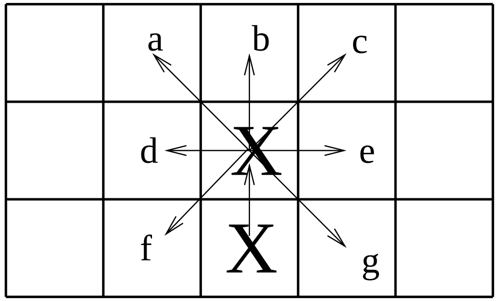


Tracked boundary (consecutive point set):



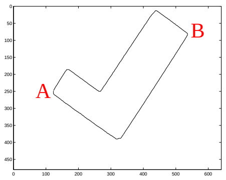
Midlecture Problem

Given the following tracking sequence:



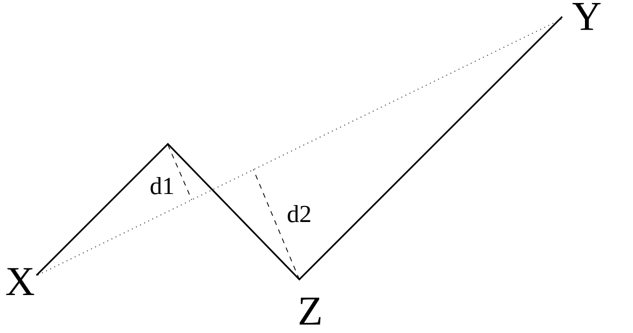
What is the order of pixels to be considered for tracking to the next pixel?

Recursive splitting the boundary into linear segments



1. Find leftmost point A
2. Find rightmost point B
3. Split points in set A– >B and B– >A:
   (a) Find line thru current segment endpoints X & Y
   (b) Find point Z furthest from the line at distance d
   (c) If d is less than a threshold, then this segment finished
   (d) Otherwise, create new sets X– >Z and Z– >Y and recurse

Recursive Splitting Algorithm



Recursive Splitting Code

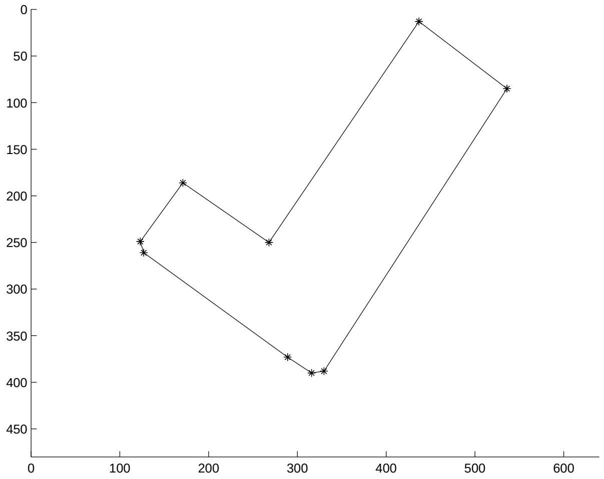
```
function recsplit(r,c,threshold)
  global numlines lines
  n = length(r);        % total number of points
  vec = [c(n)-c(1), r(1)-r(n)];  % unit vector
  vec = vec/norm(vec);           % perpendicular to XY

  % find point furthest from line
  maxdist = 0;
  for i = 1 : n
    dist = abs( [r(i) - r(1), c(i) - c(1)] * vec' );
    if dist > maxdist
      maxdist = dist;
      maxindex = i;           % where furthest
    end
  end
```

```
% check for splitting by testing maximum point distance
if maxdist < threshold
  % then it's a single line - save it
  numlines = numlines + 1;
  lines(numlines,1) = r(1);
  lines(numlines,2) = c(1);
  lines(numlines,3) = r(n);
  lines(numlines,4) = c(n);
else
  % otherwise it needs to be split up
  recsplit(r(1:maxindex),c(1:maxindex),threshold);
  recsplit(r(maxindex:n),c(maxindex:n),threshold);
end
```

Splitting Results

Segmented boundary:

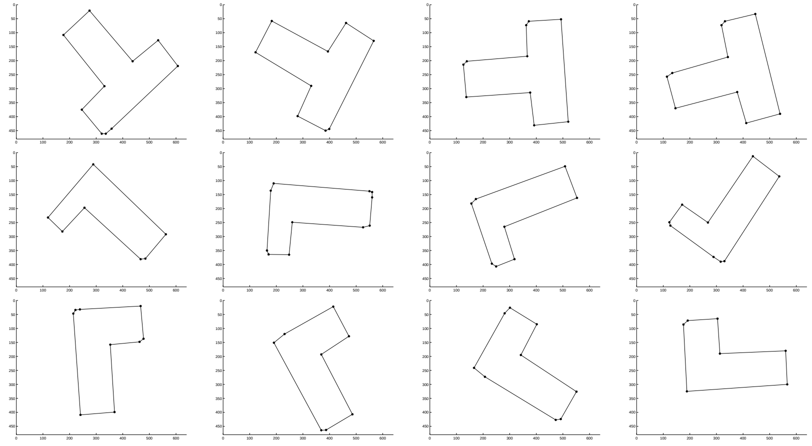


Describing Lines

Endpoints	Length	True Length
(249,123)-(261,127)	13	-
(261,127)-(373,289)	197	247
(373,289)-(390,316)	32	-
(390,316)-(388,330)	14	-
(388,330)-(85,536)	366	371
(85,536)-(13,437)	122	124
(13,437)-(250,268)	291	294
(250,268)-(186,171)	116	124
(186,171)-(249,123)	79	77

Input into matcher: extra lines, short lines, longer lines

Full Result Set



Running Program

```
>> doall
Input model to image scale factor (float)
?30.9
Want to use live test data (0,1)
?0
Test image file stem (filestem)
?TESTDATA1/f

initial_split =


   108   177   219   607


numlines =


   10
```

```
ans =


   108   177   291   331
   ...
    21   275   108   177

Want to process another image 2 (0,1)
?1
```

Discussion

1. Simple boundary track and segment process
2. Gives compact line-based description
3. May have some extra segments
4. Segments may be too long or short
5. Description is input into matcher

What Have We Learned?

Introduction to

- Data cleaning
- Boundary/Curve tracking
- Curve segmentation

From pixels to descriptions

Next: Matching descriptions to models