Announcements

- Final Exam Dec 15, 8 am (not my idea).
- Practice final handout Thursday.
- Review session: think about good times Wed afternoon.
- Office hours 12/13 at 1:00.
- Please do online class evaluation.
- Questions on problem set?

Recognizing Objects: Feature Matching

- Problem: Match when viewing conditions change a lot.
  - Lighting changes: brightness constancy false.
  - Viewpoint changes: appearance changes, many viewpoints.
- One Solution: Match using edge-based features.
  - Edges less variable to lighting, viewpoint.
  - More compact representation can lead to efficiency.
- Match image or *object* to image
  - If object, matching may be asymmetric
  - Object may be 3D.

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

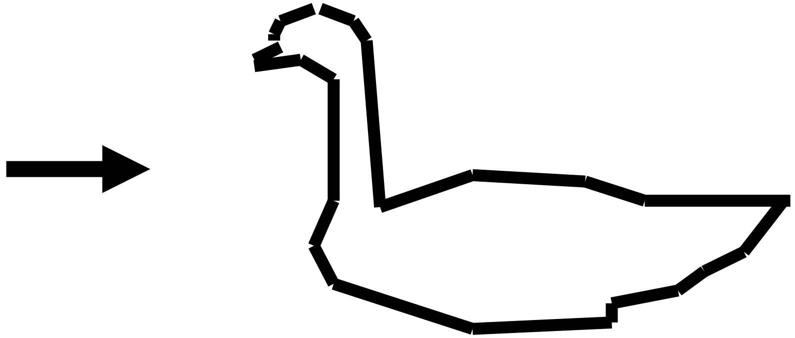
Strategy

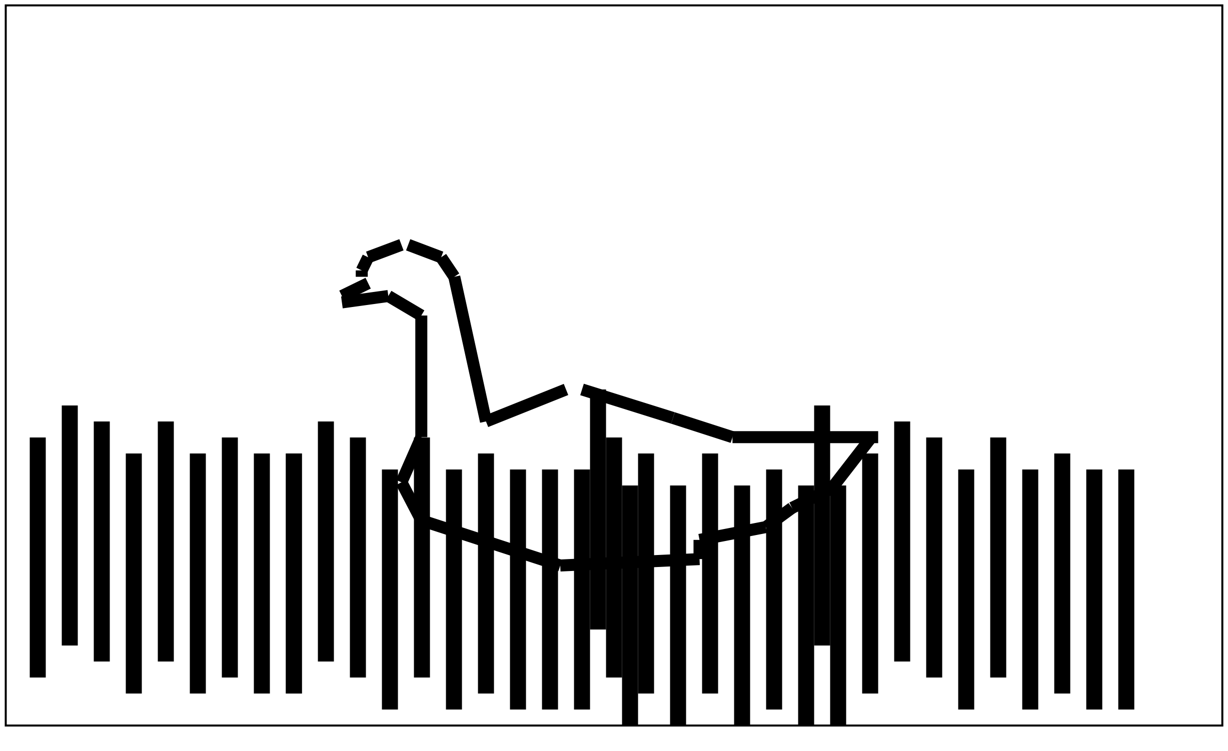
- Build feature descriptions
- Search possible poses.
  - Can search space of poses
  - Or search feature matches, which produce pose
- Transform model by pose.
- Compare transformed model and image.
- Pick best pose.

Presentation Strategy

- Already discussed finding features.
- First discuss picking best pose since this defines the problem.
- Second discuss search methods appropriate for 2D.
- Third discuss transforming model in 2D and 3D.
- Fourth discuss search for 3D objects.

Example

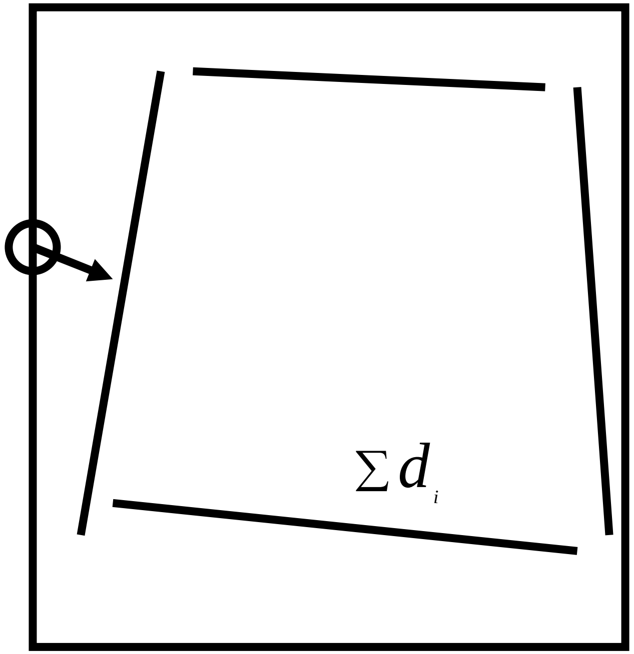




Evaluate Pose

- We look at this first, since it defines the problem.
- Again, no perfect measure;
  - Trade-offs between veracity of measure and computational considerations.

Chamfer Matching

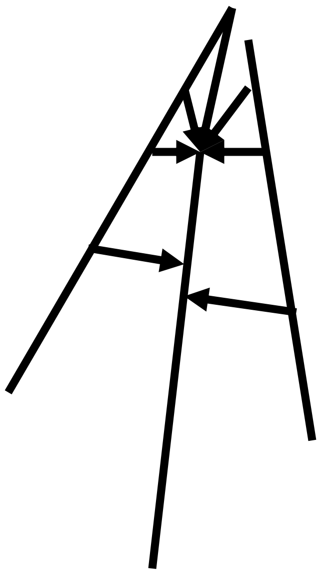


For every edge point in the transformed object, compute the distance to the nearest image edge point. Sum distances.

$$\sum_{i=1}^{n} \min(\| p_i, q_1 \|, \| p_i, q_2 \|, \ldots \| p_i, q_m \|)$$

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.

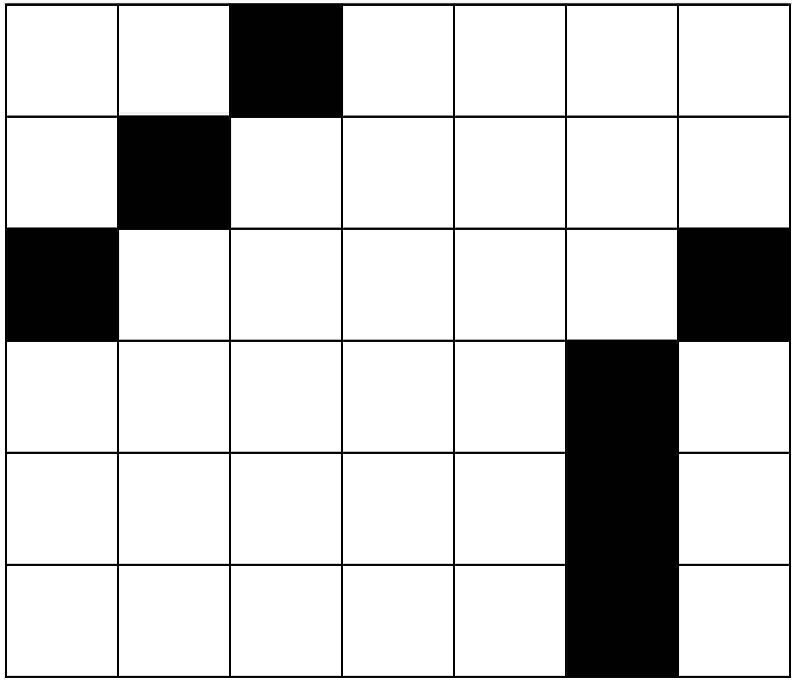
Presentation Strategy

- Already discussed finding features.
- First discuss picking best pose since this defines the problem.
- Second discuss search methods appropriate for 2D.
- Third discuss transforming model in 2D and 3D.
- Fourth discuss search for 3D objects.

Pose Search

- Simplest approach is to try every pose.
- Two problems: many poses, costly to evaluate each.
- We can reduce the second problem with:

Pose: Chamfer Matching with the 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

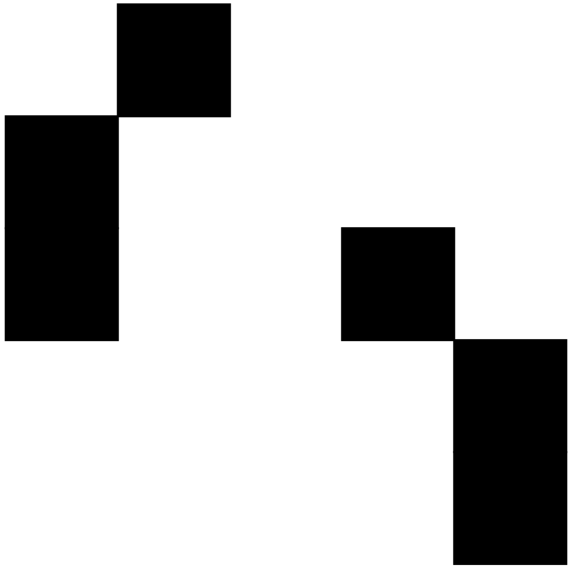
Example: Each pixel has (Manhattan) distance to nearest edge pixel.

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



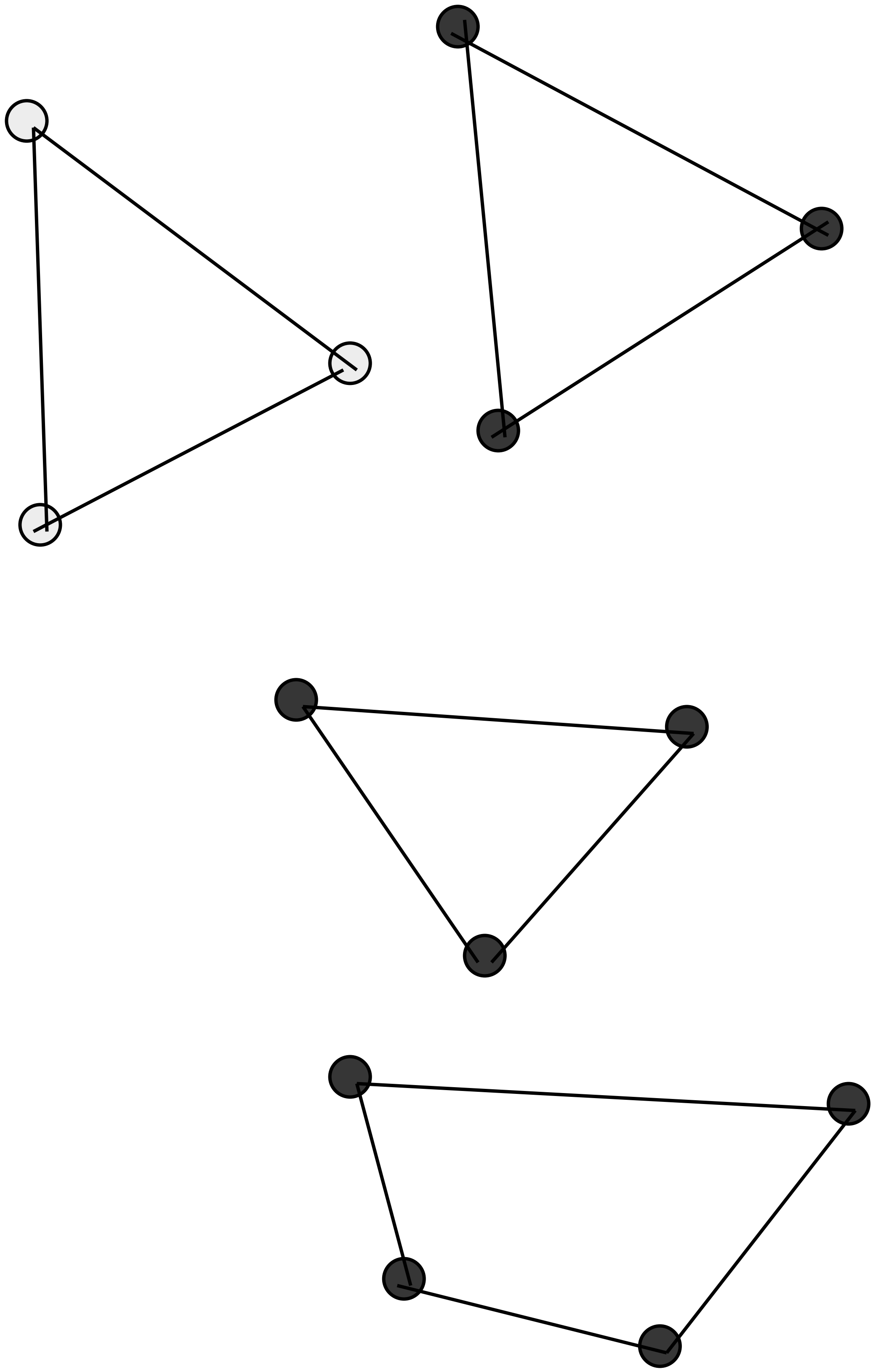
2	1	0	1	2	3	2
1	0	1	2	3	2	1
0	1	2	3	2	1	0
1	2	3	2	1	0	1
2	3	3	2	1	0	1
3	4	3	2	1	0	1

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.

Pose: 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?



Complexity

- Suppose model has *m* points and image has *n* points. There are *nm* 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:

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

Presentation Strategy

- Already discussed finding features.
- First discuss picking best pose since this defines the problem.
- Second discuss search methods appropriate for 2D.
- Third discuss transforming model in 2D and 3D.
- Fourth discuss search for 3D objects.

Computing Pose: Points

- Solve *I = S*P.*
  - In Structure-from-Motion, we knew *I*.
  - In Recognition, we know *I* and *P.*
- This is just set of linear equations
  - Ok, maybe with some non-linear constraints.

Linear Pose: 2D Translation

$$\begin{pmatrix} u_1 & u_2 & . & . & . & u_n \\ v_1 & v_2 & & & & v_n \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

We know *x,y,u,v*, need to find translation.

For one point, $u_1 - x_1 = t_x$; $v_1 - x_1 = t_y$

For more points we can solve a least squares problem.

Linear Pose: 2D rotation, translation and scale

$$\begin{pmatrix} u_1 & u_2 & . & . & . & u_n \\ v_1 & v_2 & & & & v_n \end{pmatrix} = s\begin{pmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \end{pmatrix}\begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

$$\begin{pmatrix} a & b & t_x \\ -b & a & t_y \end{pmatrix}\begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

$$\text{with} \quad a = s\cos\theta, \quad b = s\sin\theta$$

- Notice *a* and *b* can take on any values.
- Equations linear in *a, b*, translation.
- Solve exactly with 2 points, or overconstrained system with more.

$$s = \sqrt{a^2 + b^2} \qquad \cos\theta = {}^{a}\!/_{s}$$

Linear Pose: 3D Affine

$$\begin{pmatrix} u_1 & u_2 & . & . & . & u_n \\ v_1 & v_2 & & & & v_n \end{pmatrix} = \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & t_x \\ s_{2,1} & s_{2,2} & s_{2,3} & t_y \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Pose: Scaled Orthographic Projection of Planar points

$$\begin{pmatrix} u_1 & u_2 & . & . & . & u_n \\ v_1 & v_2 & & & & v_n \end{pmatrix} = \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & t_x \\ s_{2,1} & s_{2,2} & s_{2,3} & t_y \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ 0 & 0 & & & & 0 \\ 1 & 1 & & & & 1 \end{pmatrix}$$

$s_{1,3}$, $s_{2,3}$ disappear. Non-linear constraints disappear with them.

Non-linear pose

- A bit trickier. Some results:
- 2D rotation and translation. Need 2 points.
- 3D scaled orthographic. Need 3 points, give 2 solutions.
- 3D perspective, camera known. Need 3 points. Solve 4^{th} degree polynomial. 4 solutions.

Transforming the Object

We don't really want to know pose, we want to know what the object looks like in that pose.

We start with:

$$\begin{pmatrix} u_1 & u_2 & u_3 & u_4 & ? & ? & ? \\ v_1 & v_2 & v_3 & v_4 & ? & ? & ? \end{pmatrix} = \begin{pmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Solve for pose:

$$\begin{pmatrix} u_1 & u_2 & u_3 & u_4 & ? & ? & ? \\ v_1 & v_2 & v_3 & v_4 & ? & ? & ? \end{pmatrix} = \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & t_x \\ s_{2,1} & s_{2,2} & s_{2,3} & t_y \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Project rest of points:

$$\begin{pmatrix} u_1 & u_2 & . & . & . & u_n \\ v_1 & v_2 & & & & v_n \end{pmatrix} = \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & t_x \\ s_{2,1} & s_{2,2} & s_{2,3} & t_y \end{pmatrix} \begin{pmatrix} x_1 & x_2 & . & . & . & x_n \\ y_1 & y_2 & & & & y_n \\ z_1 & z_2 & & & & z_n \\ 1 & 1 & & & & 1 \end{pmatrix}$$

Recap: Recognition w/ RANSAC

1. Find features in model and image.
   - Such as corners.
2. Match enough to determine pose.
   - Such as 3 points for planar object, scaled orthographic projection.
3. Determine pose.
4. Project rest of object features into image.
5. Look to see how many image features they match.
   - Example: with bounded error, count how many object features project near an image feature.
6. Repeat steps 2-5 a bunch of times.
7. Pick pose that matches most features.

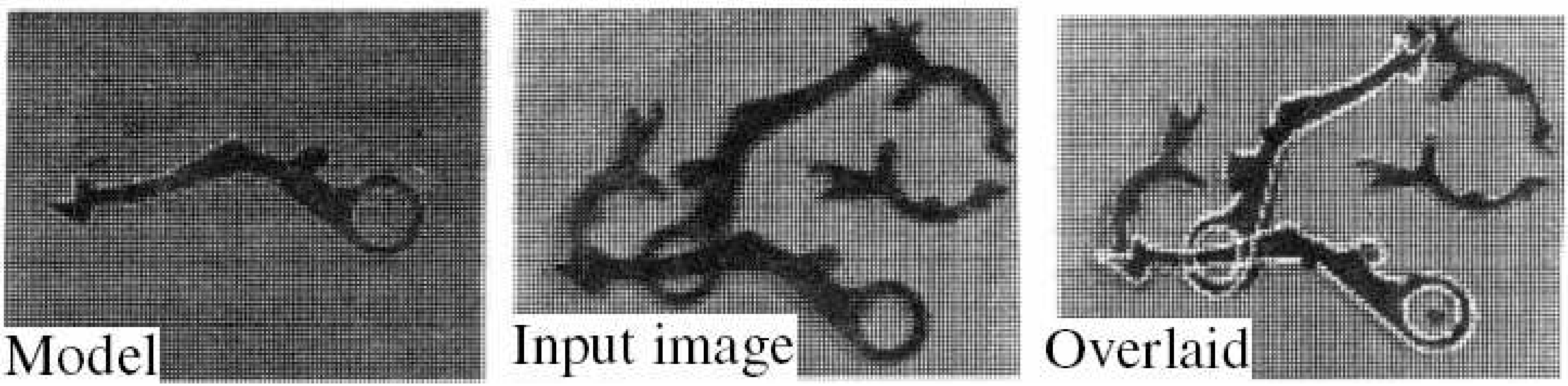
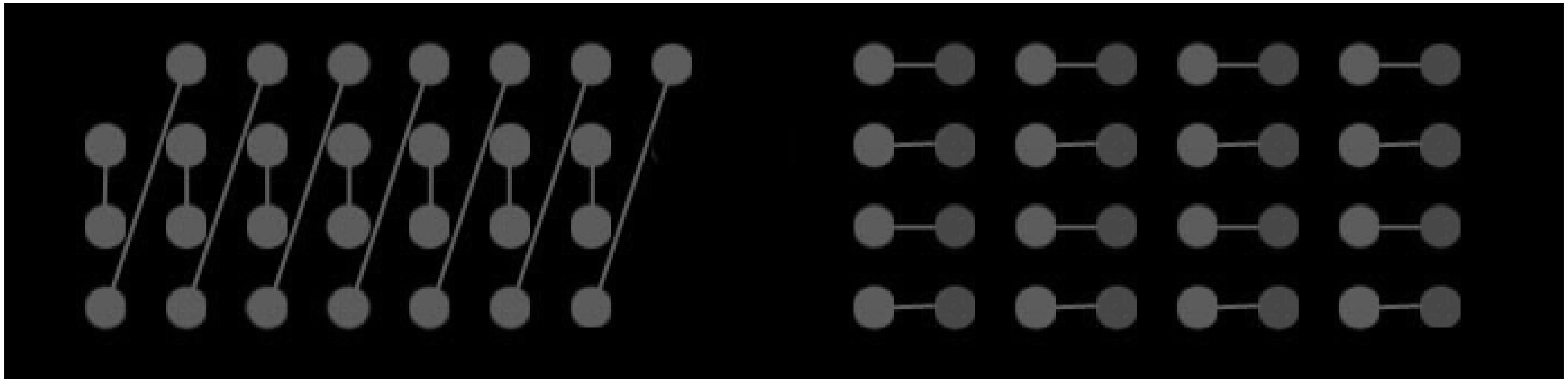


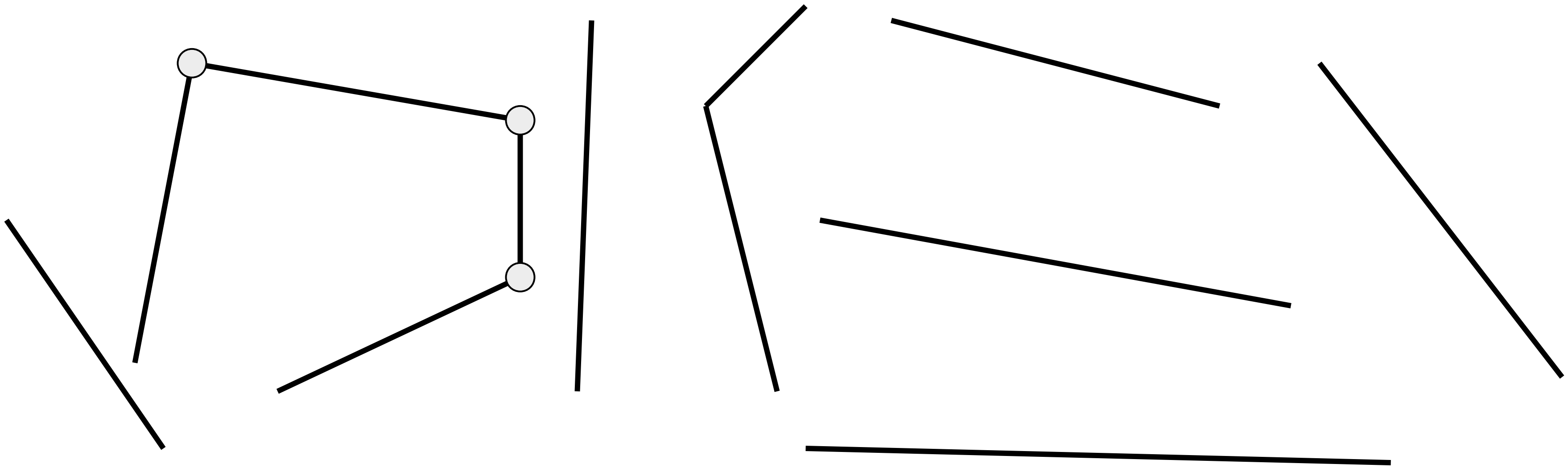
Figure from “Object recognition using alignment,” D.P. Huttenlocher and S. Ullman, Proc. Int. Conf. Computer Vision, 1986, copyright IEEE, 1986

Recognizing 3D Objects

- Previous approach will work.
- But slow. RANSAC considers n^3m^3 possible matches. About m^3 correct.
- Solutions:
  - Grouping. Find features coming from single object.
  - Viewpoint invariance. Match to small set of model features that could produce them.

Grouping: Continuity





Connectivity

- Connected lines likely to come from boundary of same object.
  - Boundary of object often produces connected contours.
  - Different objects more rarely do; only when overlapping.
- Connected image lines match connected model lines.
  - Disconnected model lines generally don’t appear connected.

Other Viewpoint Invariants

- Parallelism
- Convexity
- Common region
- ….

Figures by kind permission of Eric Grimson; further information can be obtained from his web site http://www.ai.mit.edu/people/welg/welg.html.

Figures by kind permission of Eric Grimson; further information can be obtained from his web site http://www.ai.mit.edu/people/welg/welg.html.

Figures by kind permission of Eric Grimson; further information can be obtained from his web site http://www.ai.mit.edu/people/welg/welg.html.

Figures by kind permission of Eric Grimson; further information can be obtained from his web site http://www.ai.mit.edu/people/welg/welg.html.

Figures by kind permission of Eric Grimson; further information can be obtained from his web site http://www.ai.mit.edu/people/welg/welg.html.

What we didn’t talk about

- Smooth 3D objects.
- Can we find the guaranteed optimal solution?
- Indexing with invariants.
- Error propagation.
- Classification/Learning