6.869

Advances in Computer Vision

Prof. Bill Freeman

Model-based vision

- Hypothesize and test
- Interpretation Trees
- Alignment
- Pose Clustering
- Geometric Hashing

Readings: F&P Ch 18.1-18.5

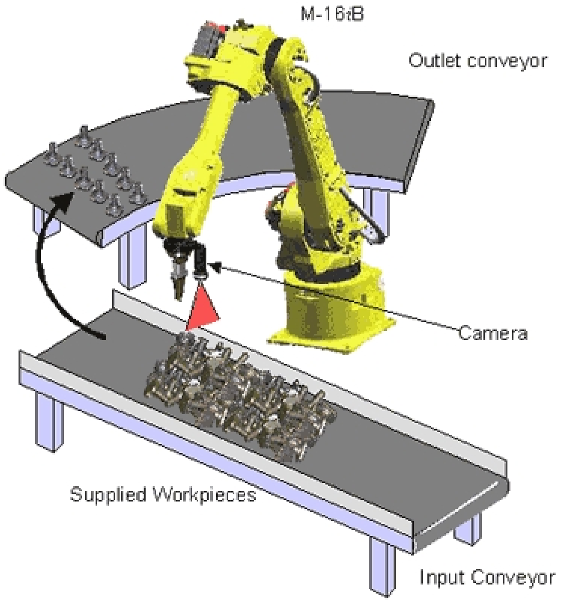
Model-based Vision

Topics:

- Hypothesize and test
  - Interpretation Trees
  - Alignment
- Interpretation trees
- Hypothesis generation methods
  - Pose clustering
  - Invariances
  - Geometric hashing
- Verification methods

Object recognition as a function of time in computer vision research

Picking identical parts from a pile



~1985

http://www.fanuc.co.jp/en/product/robot/robotshow2003/image/m-16ib20_3dv_e.gif

Recognizing instances of textured objects

~1995

dollarfifty.tripod.com/ pho/004lg.jpg

Recognizing object classes, material properties

~2005

http://images.google.com/imgres?imgurl=http://www.displayit-info.com/food/images/desserts/2131.JPG&imgrefurl=http://www.displayit-info.com/food/dessert6.html&h=504&w=501&sz=181&tbnid=FXJATGzVyA4J:&tbnh=128&tbnw=127&start=13&prev=/images%3Fq%3Dice%2Bcream%2Bsundae%26hl%3Den%26lr%3D%26sa%3DG

Paths to computer vision research

Computer science

Electrical engineering,
physics

Computer vision

Tools:
Binary numbers,
Counting,
Threshold tests,

Graph cuts.

Tools:
Real numbers,
Probabilities,
Soft decisions,

Belief propagation.

Approach

- Given
  - CAD Models (with features)
  - Detected features in an image
- Hypothesize and test recognition…
  - Guess
  - Render
  - Compare

Hypothesize and Test Recognition

- Hypothesize object identity and correspondence
  - Recover pose
  - Render object in camera
  - Compare to image
- Issues
  - where do the hypotheses come from?
  - How do we compare to image (verification)?

Features?

- Points

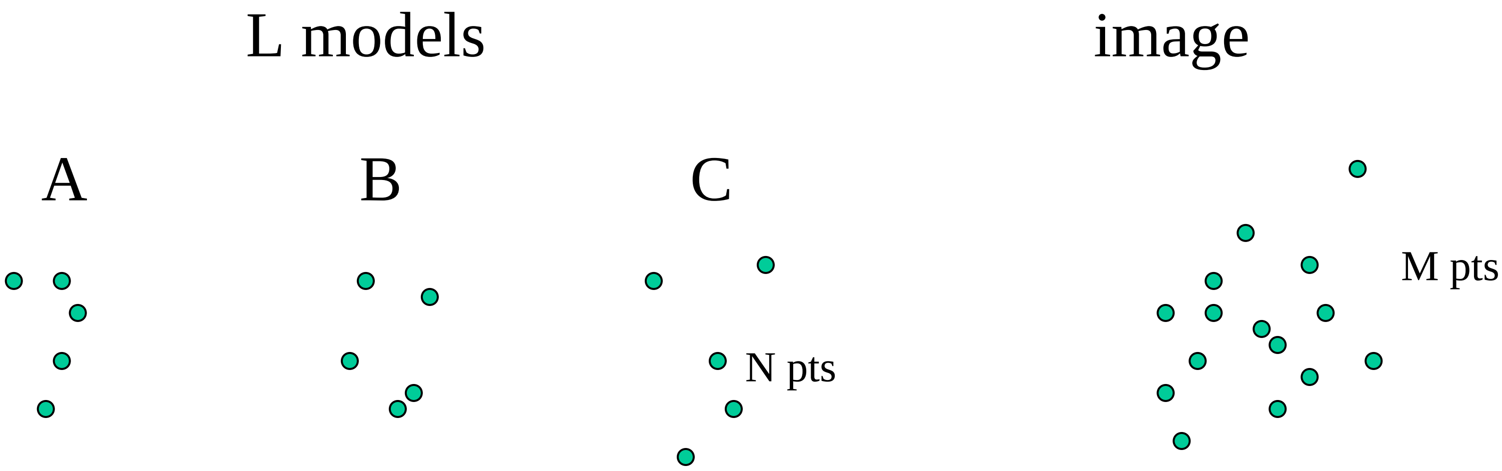
but also,

- Lines
- Conics
- Other fitted curves
- Regions (particularly the center of a region, etc.)
- More descriptive local features (eg work by Schmid and Lowe). "…of intermediate complexity, which means that they are distinctive enough to determine likely matches in a large database of features, but are sufficiently local to be insensitive to clutter and occlusion". (Lowe, CVPR01)

How to generate hypotheses?

- Brute force
  - Construct a correspondence for all object features to every correctly sized subset of image points
  - Expensive search, which is also redundant.
  - L objects with N features
  - M features in image
  - $O(LM^N)$!

Brute force method



Try all M image feature points for a model point,
Then try all M-1 remaining image feature points for another model point, then all M-2 for the next, etc.

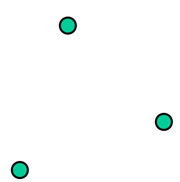
$M * (M-1) * (M-2) \ldots * (M-N+1)$ for each of L models$= O(LM^N)$

Ways around that combinatorial explosion

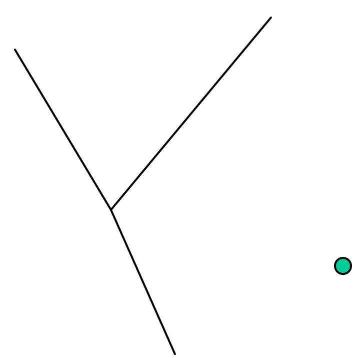
- Add geometric constraints to prune search, leading to *interpretation tree search*
- Try subsets of features (frame groups)…

Frame groups

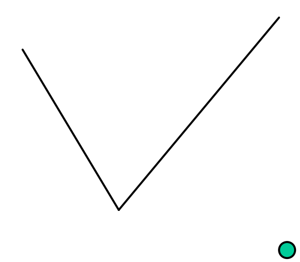
- A group of features that can yield a camera hypothesis.
- If you know the intrinsic parameters of your camera, then these are the set of features needed to specify the object's pose relative to the camera.
- With a perspective camera model, known intrinsic camera parameters, some frame groups are:



3 points



Trihedral vertex, and a point (for scale)



Dihedral vertex, and a point

Adding constraints

- Correspondences between image features and model features are not independent.
- A small number of good correspondences yields a reliable pose estimation --- the others must be consistent with this.
- Generate hypotheses using small numbers of correspondences (e.g. triples of points for a calibrated perspective camera, etc., etc.)

Pose consistency / Alignment

- Given known camera type in some unknown configuration (pose)
  - Hypothesize configuration from set of initial features
  - Backproject
  - Test

Rendering an object into the image

Perspective camera

pixel coordinates

world coordinates

$$\vec{p} = \frac{1}{z} M\vec{P}$$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z}\begin{pmatrix} . & m_1^T & . & . \\ . & m_2^T & . & . \\ . & m_3^T & . & . \end{pmatrix}\begin{pmatrix} W_x \\ W_y \\ W_z \\ 1 \end{pmatrix}$$

$$\begin{cases} u = \dfrac{m_1 \cdot \vec{P}}{m_3 \cdot \vec{P}} \\[2ex] v = \dfrac{m_2 \cdot \vec{P}}{m_3 \cdot \vec{P}} \end{cases}$$

z is in the *camera* coordinate system, but we can solve for that, since $1 = \frac{m_3 \cdot \vec{P}}{z}$, leading to:

Rendering an object into the image

Affine camera

Rendering ith 3d pt to 2d image position

$$p_i = \Pi A P_i$$

General affine transformation

Orthographic camera

$$\Pi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A frame group for an affine camera model

Affine camera

Rendering ith 3d pt to 2d image position

$$p_i = \Pi A P_i$$

Orthographic camera

$$\Pi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

General affine transformation

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Relating observed 2-d positions to 3-d model positions

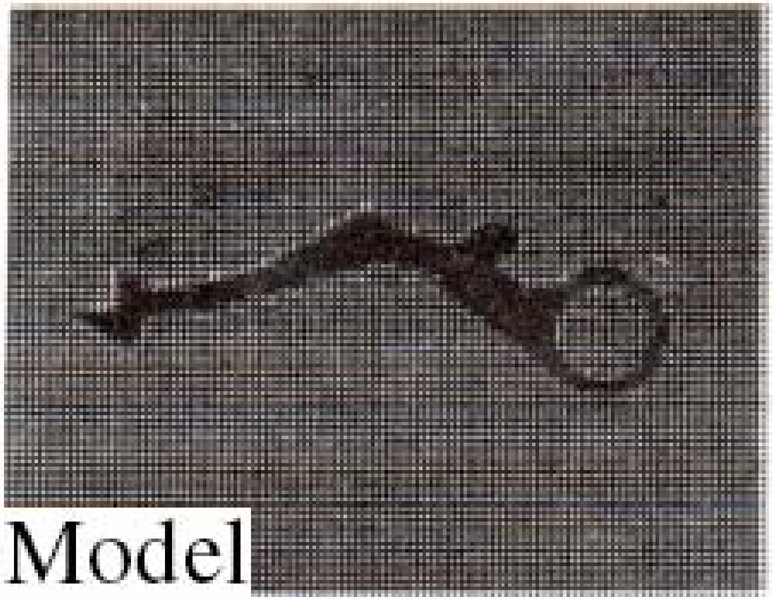
$$\begin{pmatrix} p_{i0} \\ p_{i1} \end{pmatrix} = \begin{pmatrix} a_{00}P_{i0} + a_{01}P_{i1} + a_{02}P_{i2} + a_{03}P_{i3} \\ a_{10}P_{i0} + a_{11}P_{i1} + a_{12}P_{i2} + a_{13}P_{i3} \end{pmatrix}$$

Need at least 4 points in general position to determine the affine camera parameters.

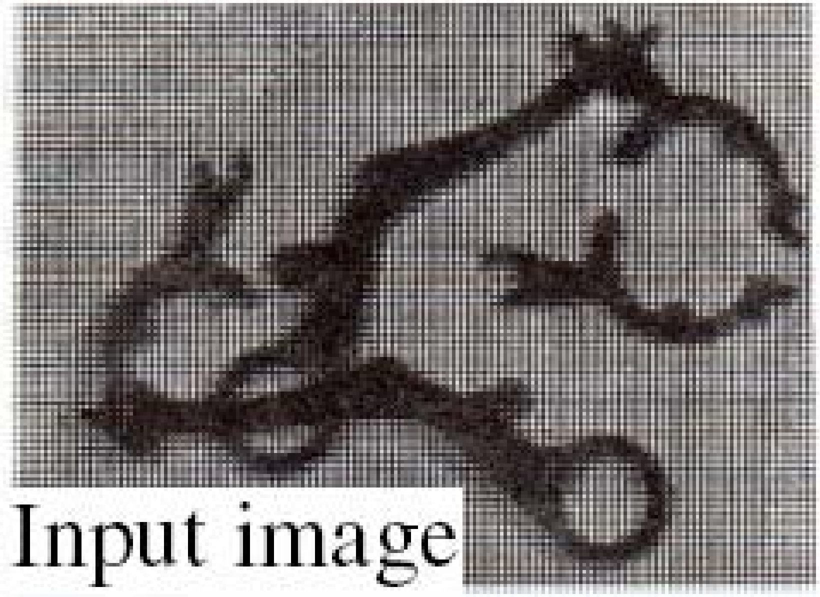
(Note: only the 1st 2 rows of A contribute to the projection, so we only need to estimate them.)

Alignment algorithm

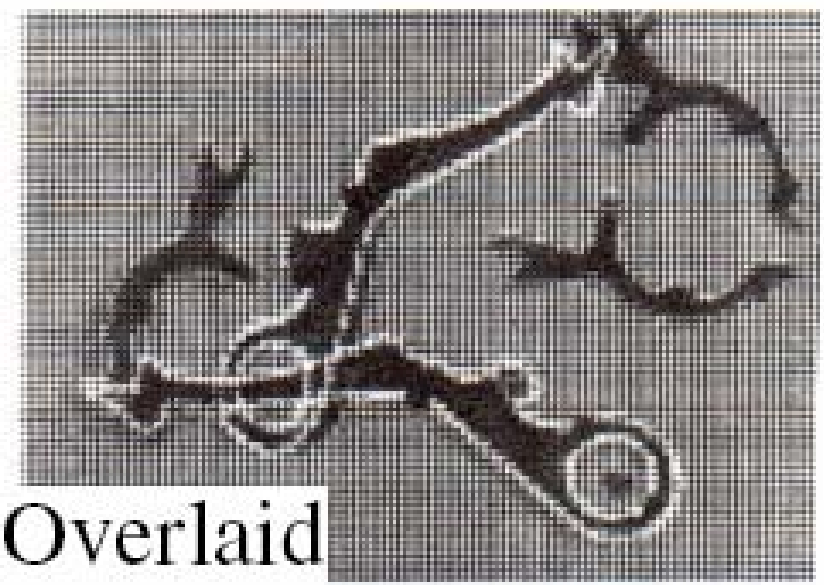
```
For all object frame groups O
  For all image frame groups F
    For all correspondences C between
      elements of F and elements
      of O

      Use F, C and O to infer the missing parameters
      in a camera model

      Use the camera model estimate to render the object

      If the rendering conforms to the image,
        the object is present
    end
  end
end
```



Model



Input image



Overlaid

More than 1 object in image

- Require same intrinsic camera parameters for each object.

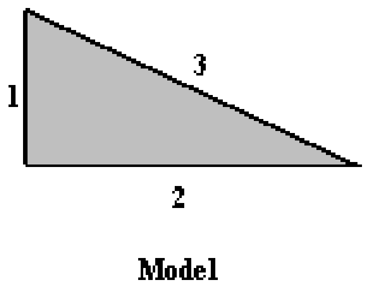
Model-based Vision

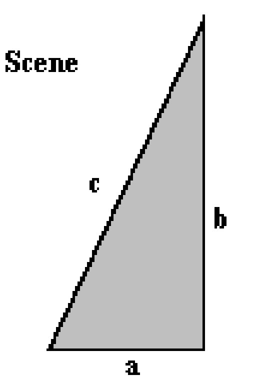
Topics:

- Hypothesize and test
  - Interpretation Trees
  - Alignment
- Interpretation trees
- Hypothesis generation methods
  - Pose clustering
  - Invariances
  - Geometric hashing
- Verification methods

Interpretation Trees

- Tree of possible model-image feature assignments
- Depth-first search
- Prune when unary (binary, …) constraint violated
  - length
  - area
  - orientation



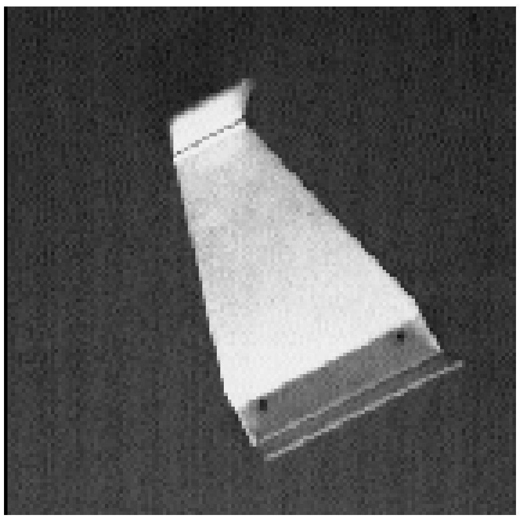


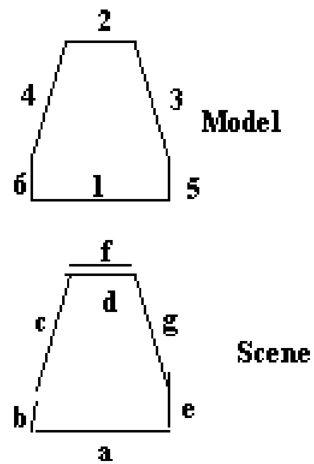
(a,1)

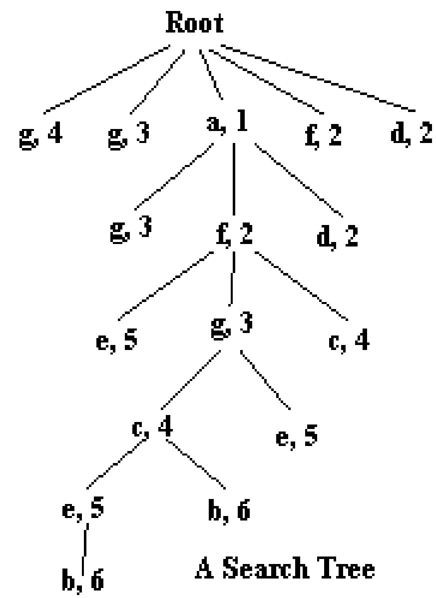
(b,2) …

…

Interpretation Trees







"Wild cards" handle spurious image features

[A.M. Wallace. 1988.]

http://faculty.washington.edu/cfolson/papers/pdf/icpr04.pdf

Model-based Vision

Topics:

- Hypothesize and test
  - Interpretation Trees
  - Alignment
- Interpretation trees
- Hypothesis generation methods
  - Pose clustering
  - Invariances
  - Geometric hashing
- Verification methods

- How does the hypothesize and test method fail?
  - False matches
  - Too many hypotheses to consider
- To add robustness and efficiency, use other heuristics to select candidate object poses

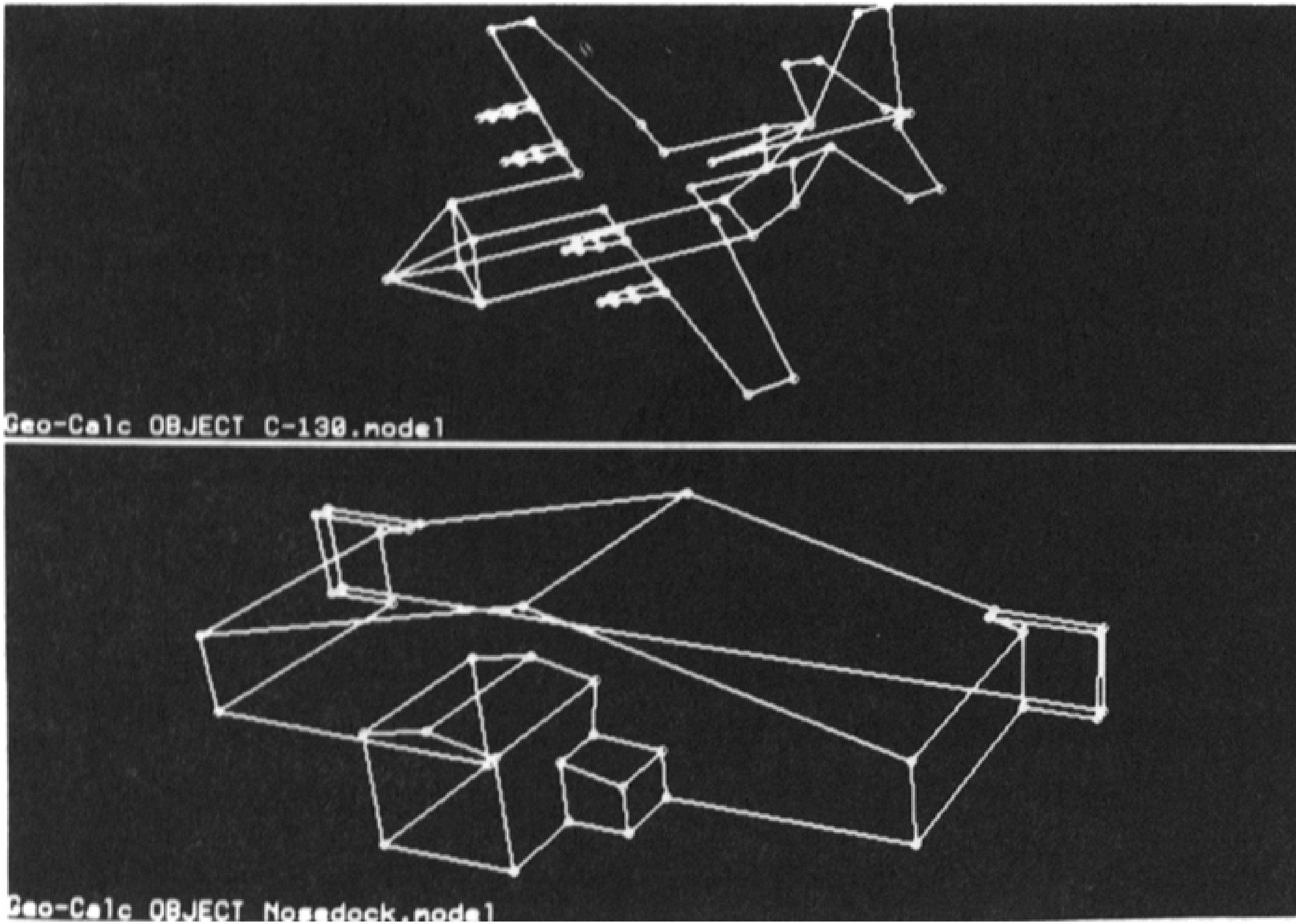
Pose clustering

- Each model leads to many correct sets of correspondences, each of which has the same pose
- Vote on object pose, in an accumulator array (per object)
- This is a computer science approach to doing a more probabilistic thing: treating each set of feature observations as statistically independent and multiplying together their probabilities of occurrence to obtain a likelihood function.

Pose Clustering

```
For all objects O
  For all object frame groups F(O)
    For all image frame groups F(I)
      For all correspondences C between
        elements of F(I) and elements
        of F(O)

        Use F(I), F(O) and C to infer object pose P(O)

        Add a vote to O's pose space at the bucket
        corresponding to P(O).
      end
    end
  end
end
For all objects O
  For all elements P(O) of O's pose space that have
    enough votes

    Use the P(O) and the
    camera model estimate to render the object

    If the rendering conforms to the image,
    the object is present
  end
end
```

Two models used in an early pose clustering system



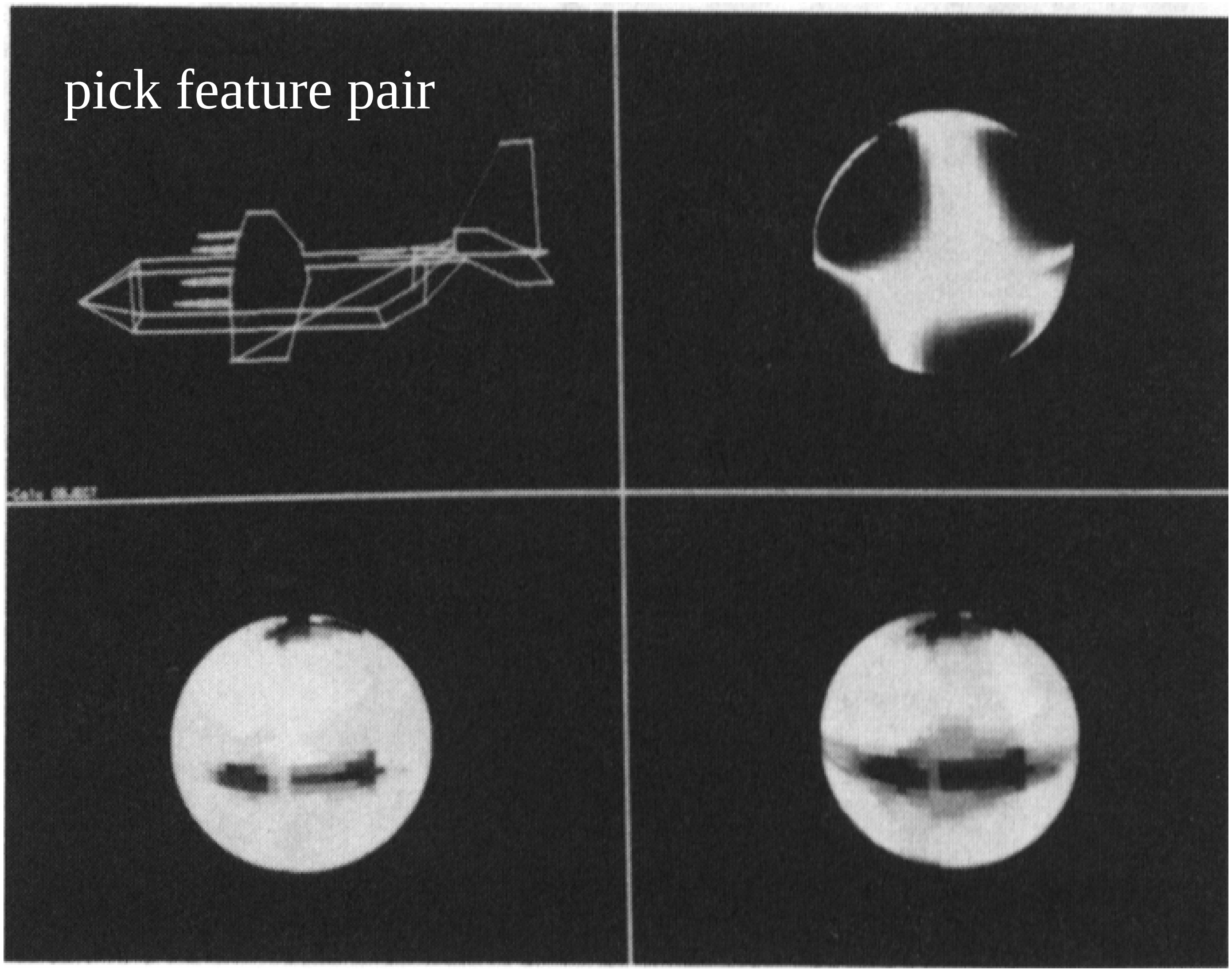
Pose clustering

Problems

- Clutter may lead to more votes than the target!
- Difficult to pick the right bin size

Confidence-weighted clustering

- See where model frame group is reliable (visible!)
- Downweight / discount votes from frame groups at poses where that frame group is unreliable…
- Again, we can make this more precise in a probabilistic framework later.



dark regions show reliable-pose-estimate views of those features over the viewing sphere

Test image, with edge points marked

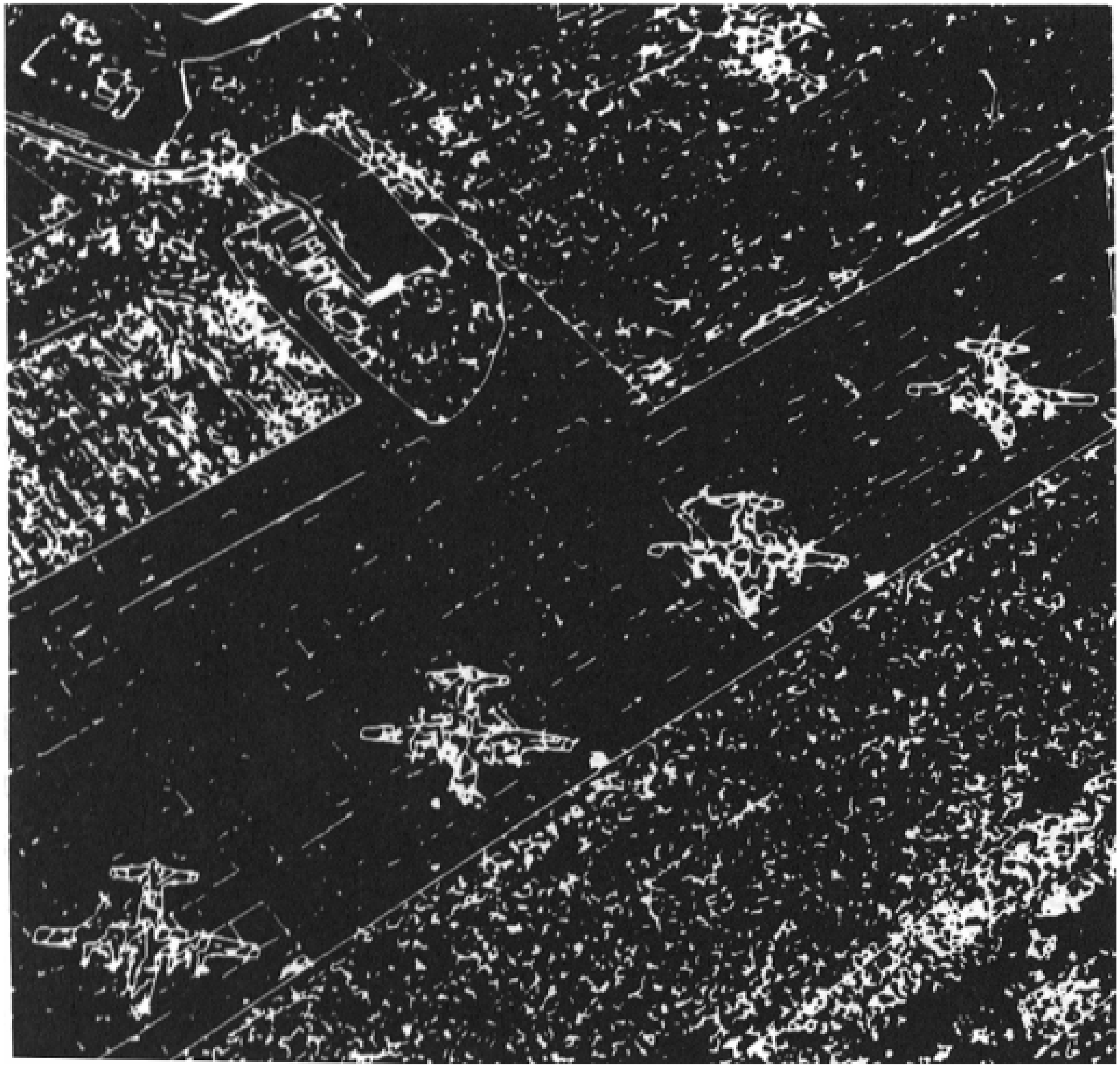
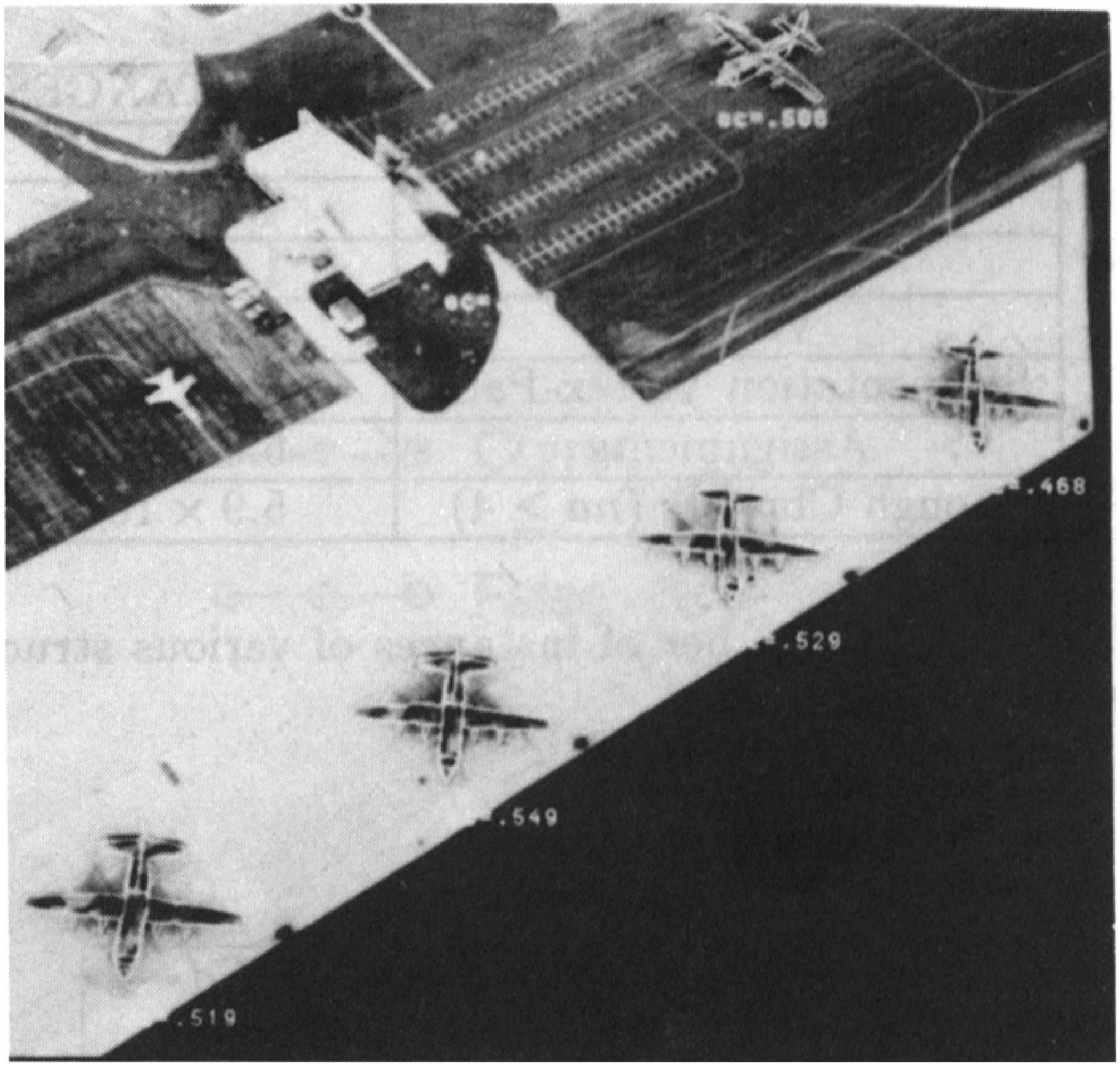
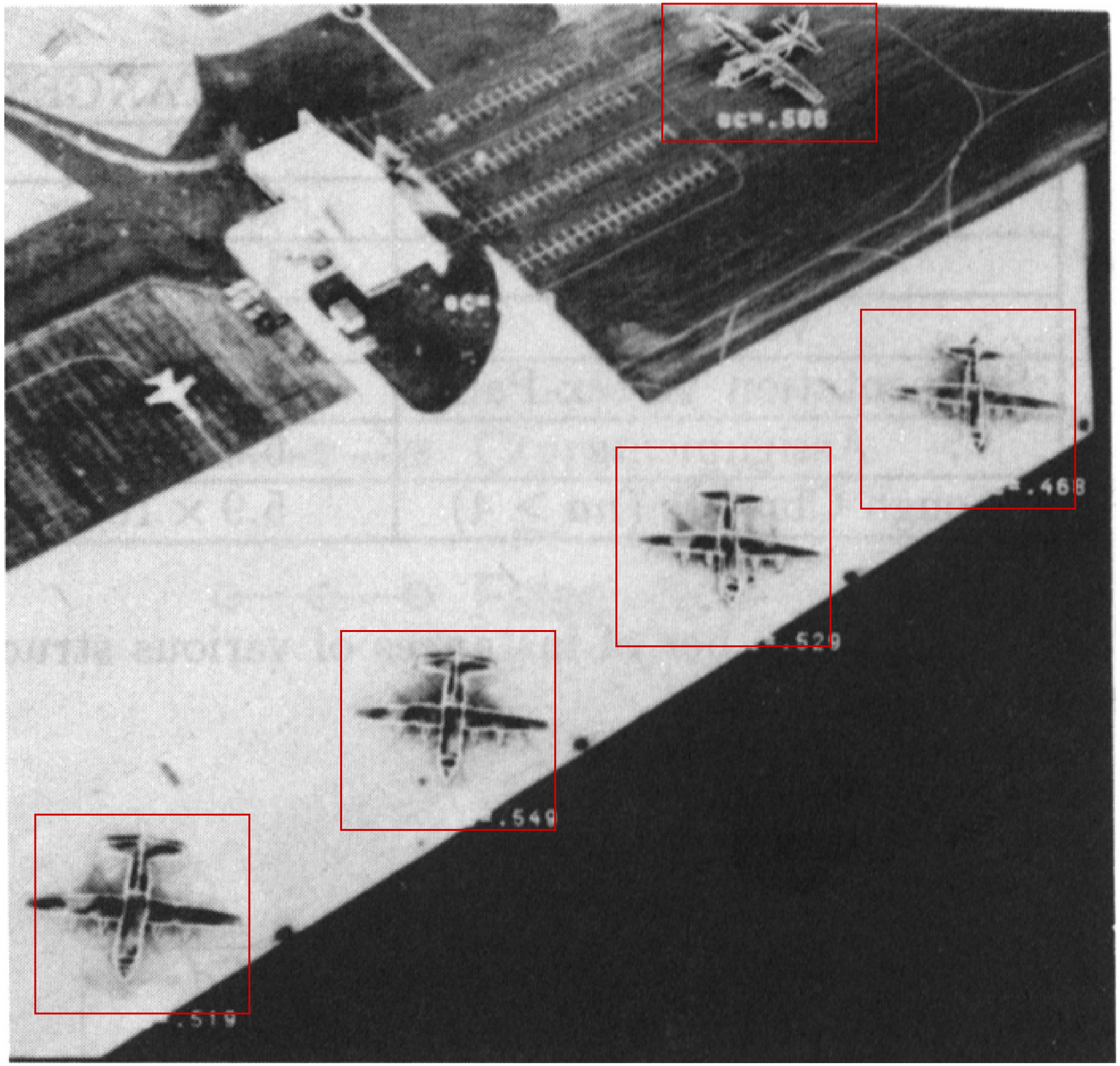
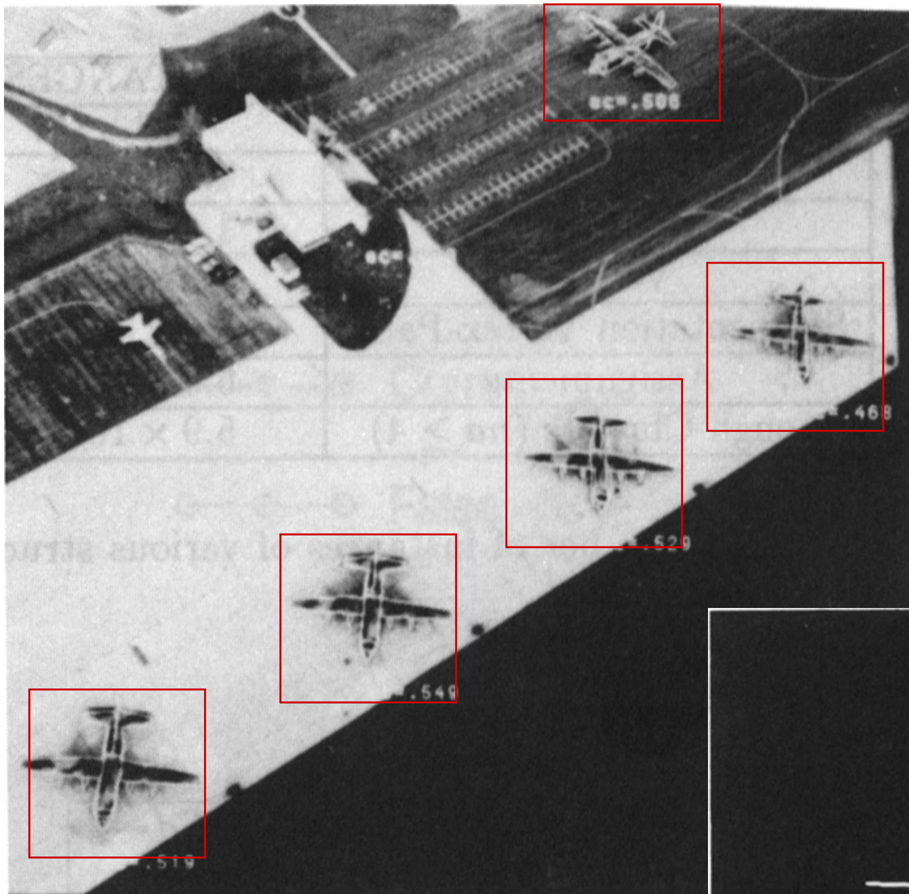


Image with edges of found models overlaid

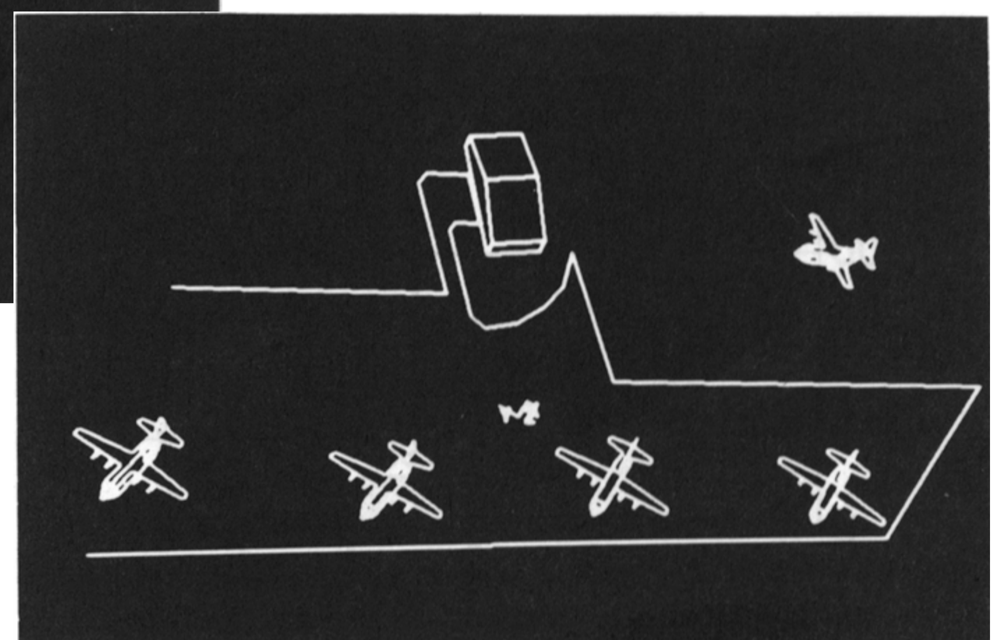




.468
.529
.549
.519



Detected airplanes, rerendered at their detected poses. (Note mis-estimated pose of plane on runway.)



A more recent pose/view clustering example

- "Local feature view clustering for 3D object recognition", by David Lowe (see his web page for copy).

- Schmid, Lowe incorporate "super-features", point features with robust local image descriptors

Detecting 0.1% inliers among 99.9% outliers?

- Example: David Lowe's SIFT-based Recognition system
- Goal: recognize clusters of just 3 consistent features among 3000 feature match hypotheses
- Approach
  - Vote for each potential match according to model ID and pose
  - Insert into multiple bins to allow for error in similarity approximation
  - Using a hash table instead of an array avoids need to form empty bins or predict array size

[Lowe]

Lowe’s Model verification step

- Examine all clusters with at least 3 features
- Perform least-squares affine fit to model.
- Discard outliers and perform top-down check for additional features.
- Evaluate probability that match is correct
  - Use Bayesian model, with probability that features would arise by chance if object was *not* present
  - Takes account of object size in image, textured regions, model feature count in database, accuracy of fit (Lowe, CVPR 01)

[Lowe]

Solution for affine parameters

- Affine transform of [x,y] to [u,v]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Rewrite to solve for transform parameters:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

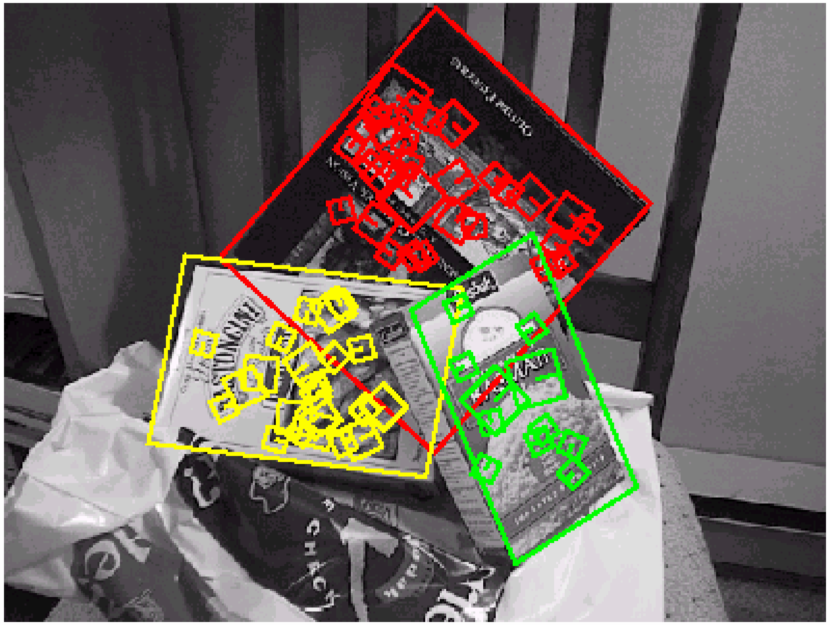
[Lowe]

Models for planar surfaces with SIFT keys:

[Lowe]

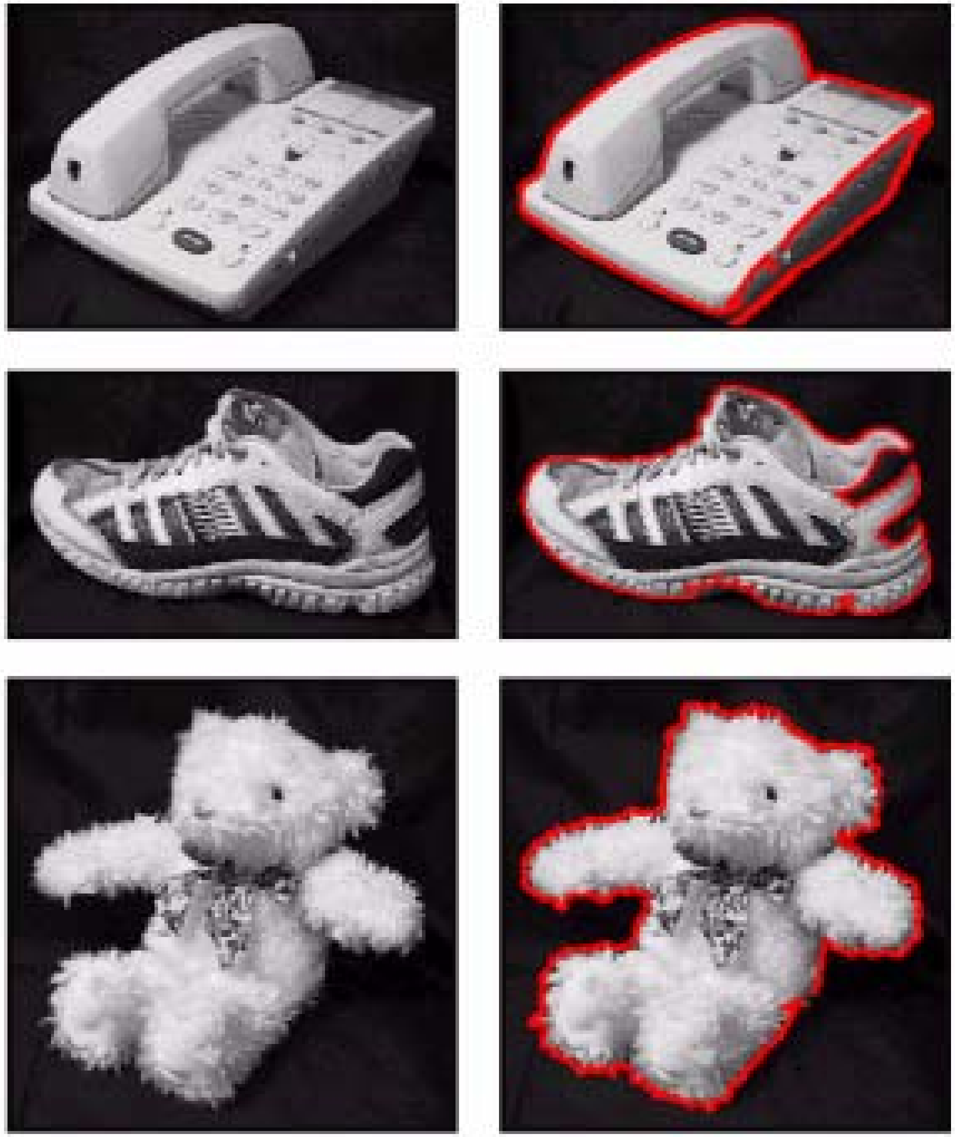
Planar recognition

- Planar surfaces can be reliably recognized at a rotation of 60° away from the camera
- Affine fit approximates perspective projection
- Only 3 points are needed for recognition



[Lowe]

3D Object Recognition



- Extract outlines with background subtraction

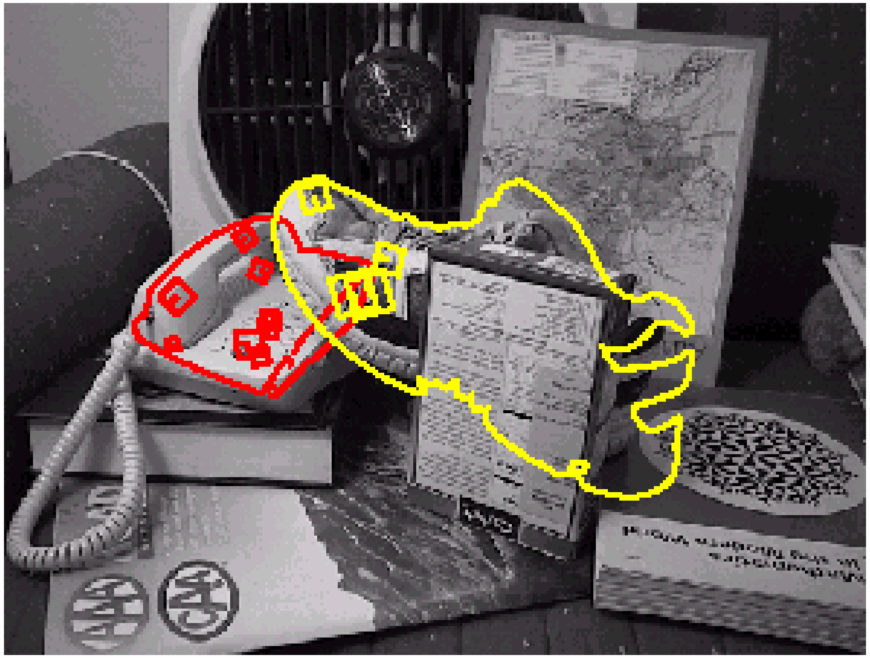
[Lowe]

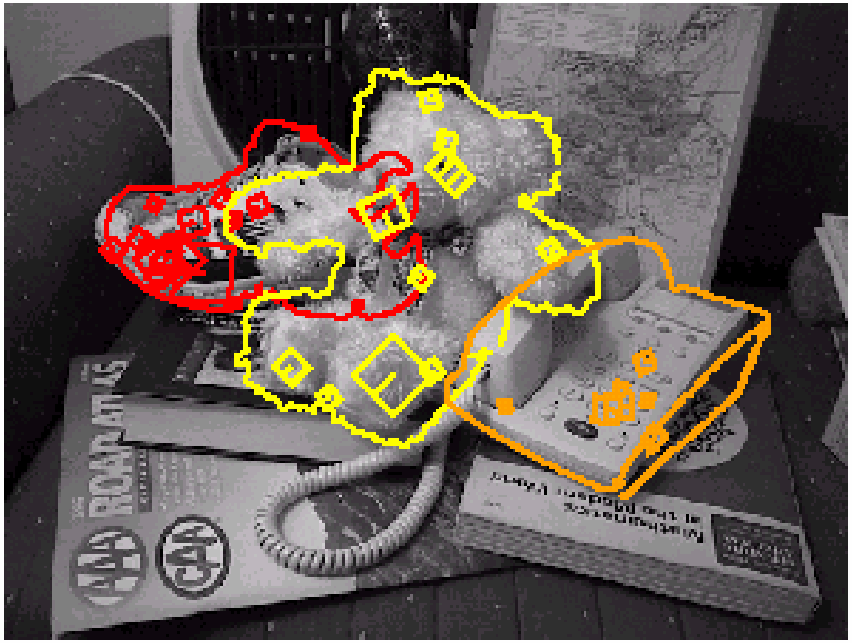
3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate

[Lowe]

Recognition under occlusion





[Lowe]

Model-based Vision

Topics:

- Hypothesize and test
  - Interpretation Trees
  - Alignment
- Interpretation trees
- Hypothesis generation methods
  - Pose clustering
  - Invariances
  - Geometric hashing
- Verification methods

Geometric Invariant recognition

- It’s a pain to compute some many pose or correspondences for verification. So insert a pruning step that is invariant to camera/object pose parameters.

- Affine invariants
  - Planar invariants
  - Geometric hashing
- Projective invariants
  - Determinant ratio
- Curve invariants

Invariance

- There are geometric properties that are invariant to camera transformations
- Easiest case: view a plane object in scaled orthography.
- Assume we have three base points P_i on the object
  - then any other point on the object can be written as

$$P_k = P_1 + \mu_{ka}\left(P_2 - P_1\right) + \mu_{kb}\left(P_3 - P_1\right)$$

Invariance

- Now image points are obtained by multiplying by a plane affine transformation, so

$$
\begin{aligned}
p_k &= AP_k \\
&= A\big(P_1 + \mu_{ka}(P_2 - P_1) + \mu_{kb}(P_3 - P_1)\big) \\
&= p_1 + \mu_{ka}(p_2 - p_1) + \mu_{kb}(p_3 - p_1)
\end{aligned}
$$

Invariance

$$P_k = P_1 + \mu_{ka}\left(P_2 - P_1\right) + \mu_{kb}\left(P_3 - P_1\right)$$

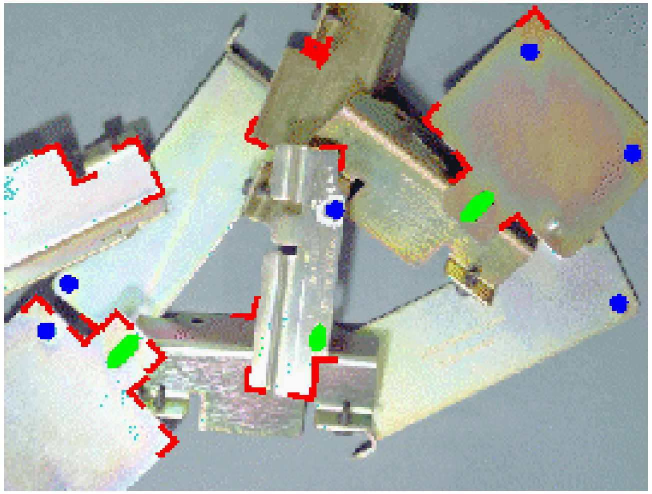
$$\begin{aligned} p_k &= AP_k \\ &= A\left(P_1 + \mu_{ka}\left(P_2 - P_1\right) + \mu_{kb}\left(P_3 - P_1\right)\right) \\ &= p_1 + \mu_{ka}\left(p_2 - p_1\right) + \mu_{kb}\left(p_3 - p_1\right) \end{aligned}$$

Given the base points in the image, read off the μ values for the object

- they're the same in object and in image **--- invariant**
- search correspondences, form μ's and vote

Indexing

- Operation that lets you select the model from a menu of possible ones, before you need to find the pose and verify.



http://www.tnt.uni-hannover.de/project/imgint/industrial/3dpos/3DLageerkennung.gif

Indexing with invariants

- Generalize to heterogeneous geometric features
- Groups of features with identity information invariant to pose – *invariant bearing groups*

Projective invariants

- Projective invariant for coplanar points
- Perspective projection of coplanar points is a plane perspective transform:

  p=MP $\longrightarrow$ p=AP, with 3x3 A

- determinant ratio of 5 point tuples is invariant

$$\frac{\det([p_i p_j p_k])\det([p_i p_l p_m])}{\det([p_i p_j p_l])\det([p_i p_k p_m])}$$

$$\frac{\det([p_i\, p_j\, p_k])\det([p_i\, p_l\, p_m])}{\det([p_i\, p_j\, p_l])\det([p_i\, p_k\, p_m])} = \frac{\det([AP_i\, AP_j\, AP_k])\det([AP_i\, AP_l\, AP_m])}{\det([AP_i\, AP_j\, AP_l])\det([AP_i\, AP_k\, AP_m])}$$

$$= \frac{\det(A\,[P_i P_j P_k])\det(A\,[P_i P_l P_m])}{\det(A\,[P_i P_j P_l])\det(A\,[P_i P_k P_m])}$$

$$= \frac{(\det(A)^2)}{(\det(A)^2)}\frac{\det([P_i P_j P_k])\det([P_i P_l P_m])}{\det([P_i P_j P_l])\det([P_i P_k P_m])}$$

$$= \frac{\det([P_i P_j P_k])\det([P_i P_l P_m])}{\det([P_i P_j P_l])\det([P_i P_k P_m])}$$

Geometric Hashing

- Objects are represented as sets of “features”
- Preprocessing:
  - For each tuple *b* of features, compute location (μ) of all other features in basis defined by *b*
  - Create a table indexed by (μ)
  - Each entry contains *b* and object ID

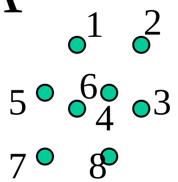
S. Rusinkiewicz

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

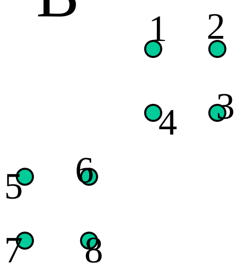
Geometric hashing

Models

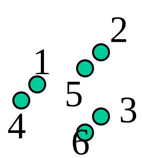
A



B



C



Hash table

$$\begin{array}{cc} & \mu_1 \longrightarrow \\ \begin{array}{c}\mu_2\\ \downarrow\end{array} & \begin{pmatrix} D;1,5,6 & & \\ & A;4,5,7 & A;4,5,7 \\ A;1,3,4\,/\,D;1,5,6 & D;1,3,4 & \\ & & B;2,3,6 \end{pmatrix} \end{array}$$

GH: Identification

- Find features in target image
- Choose an arbitrary basis *b'*
- For each feature:
  - Compute (μ') in basis *b'*
  - Look up in table and vote for (Object, *b*)
- For each (Object, *b*) with many votes:
  - Compute transformation that maps *b* to *b'*
  - Confirm presence of object, using all available features

S. Rusinkiewicz

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

Geometric Hashing

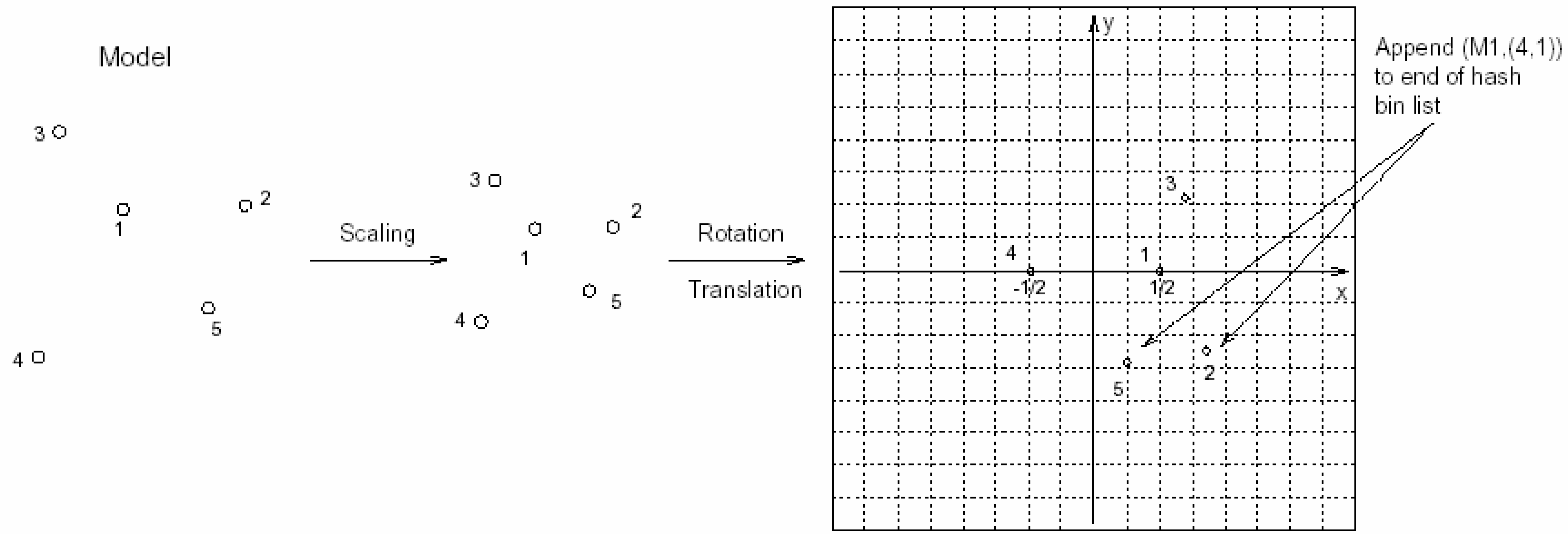
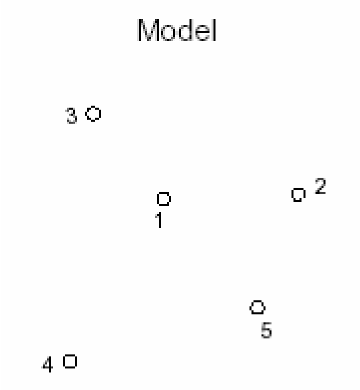


Figure 1. Determining the hash table entries when points 4 and 1 are used to define a basis. The models are allowed to undergo rotation, translation, and scaling. On the left of the figure, model *M*₁ comprises five points.

Wolfson and Rigoutsos, *Geometric Hashing, an Overview*, 1997

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

Basis Geometric Hashing



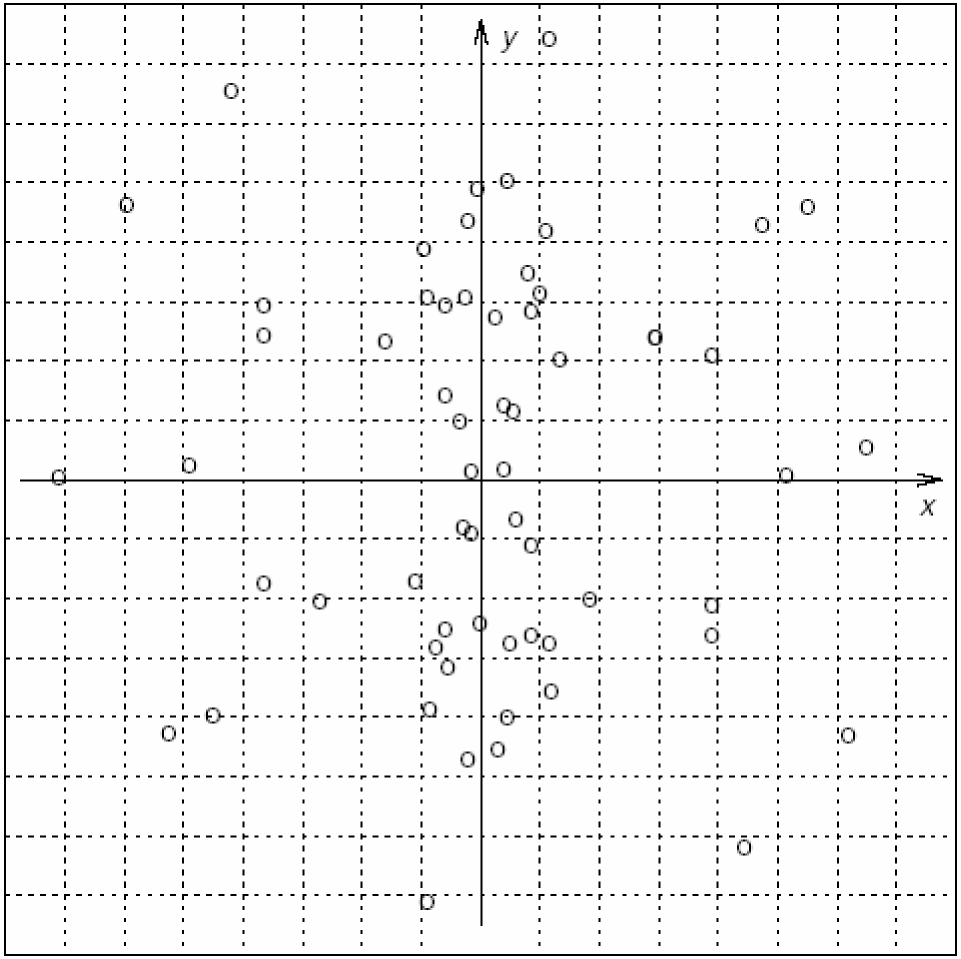


Figure 2. The locations of the hash table entries for model M_1. Each entry is labeled with the information "model M_1" and the basis pair (i, j) used to generate the entry. The models are allowed to undergo rotation, translation, and scaling.

Wolfson and Rigoutsos, *Geometric Hashing, an Overview*, 1997

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

Geometric Hashing

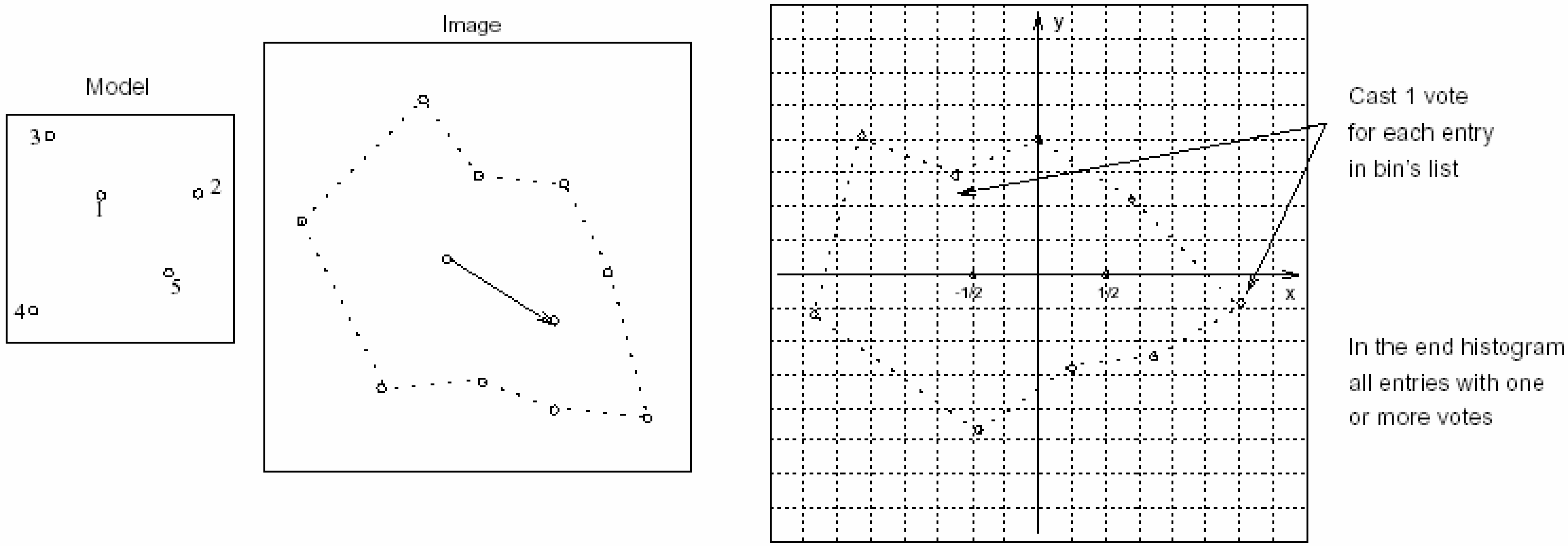


Figure 3. Determining the hash table bins that are to be notified when two arbitrary image points are selected as a basis. Similarity transformation is allowed.

Wolfson and Rigoutsos, *Geometric Hashing, an Overview*, 1997

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

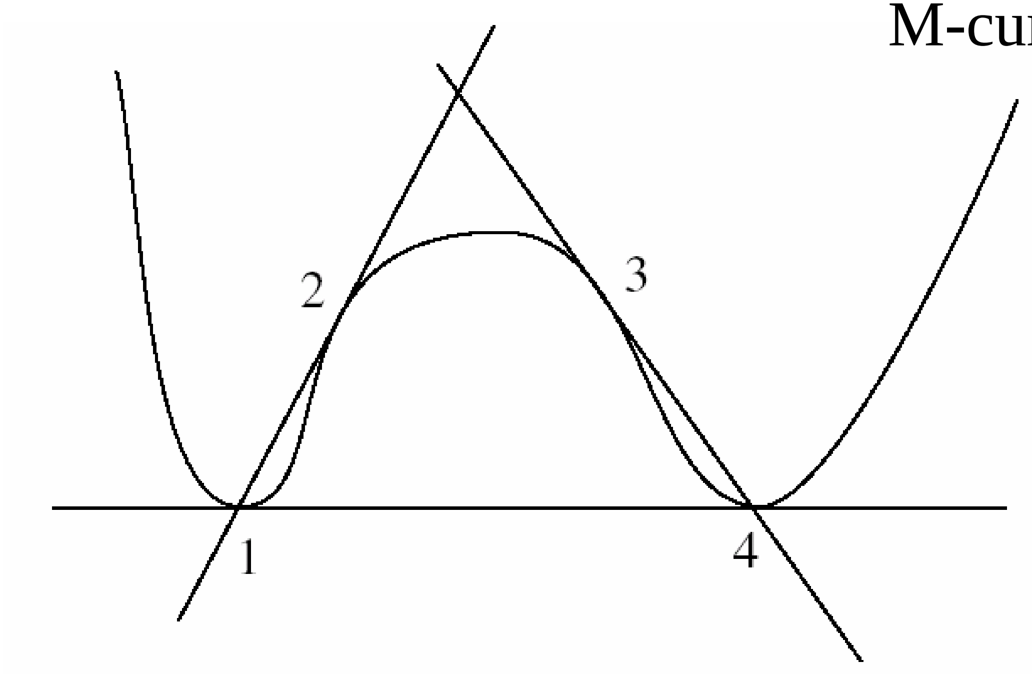
Algorithm 18.3: Geometric hashing: voting on identity and point labels

For all groups of three image points $T(I) == b$
  For every other image point p
    Compute the μ's from p and $T(I)$
    Obtain the table entry at these values
      if there is one, it will label the three points in $T(I)$
      with the name of the object
      and the names of these particular points.
    Cluster these labels;
      if there are enough labels, backproject and verify
    end
  end
end

Tangent invariance

- Incidence is preserved despite transformation

M-curve construction



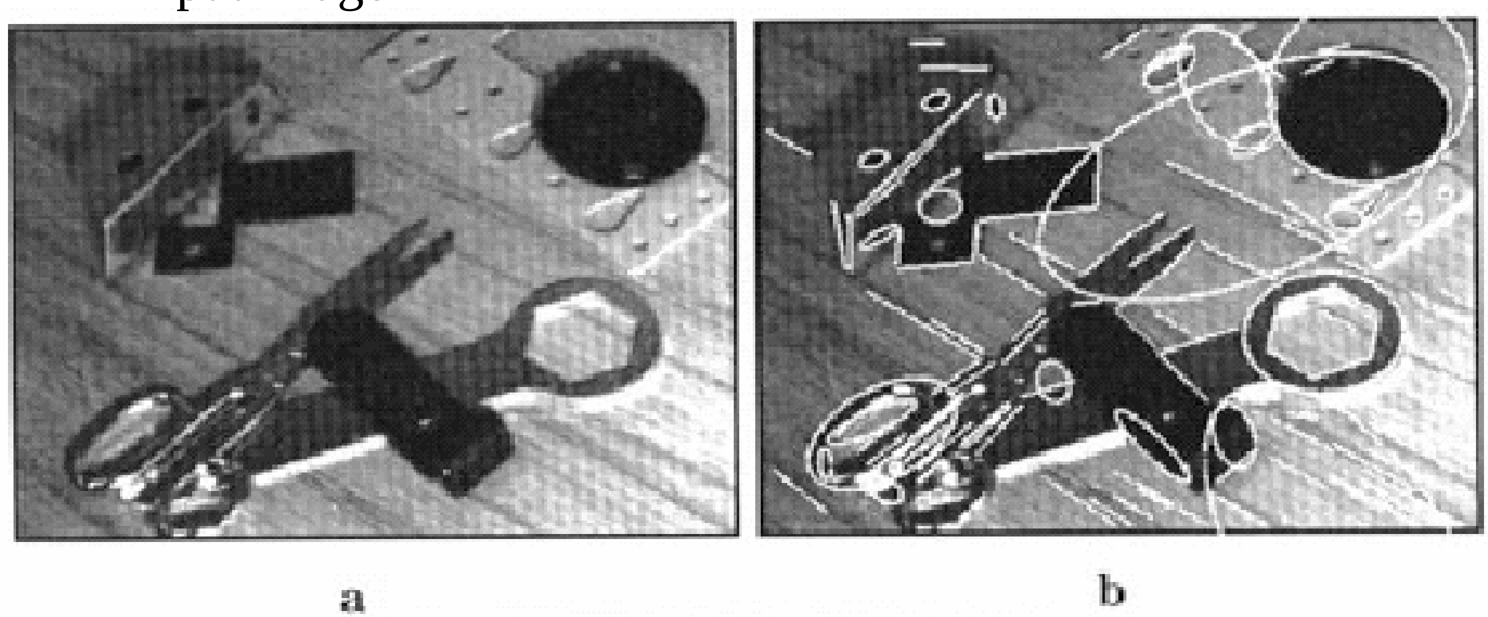
- Transform four points above to unit square: measurements in this canonical frame will be invariant to pose.

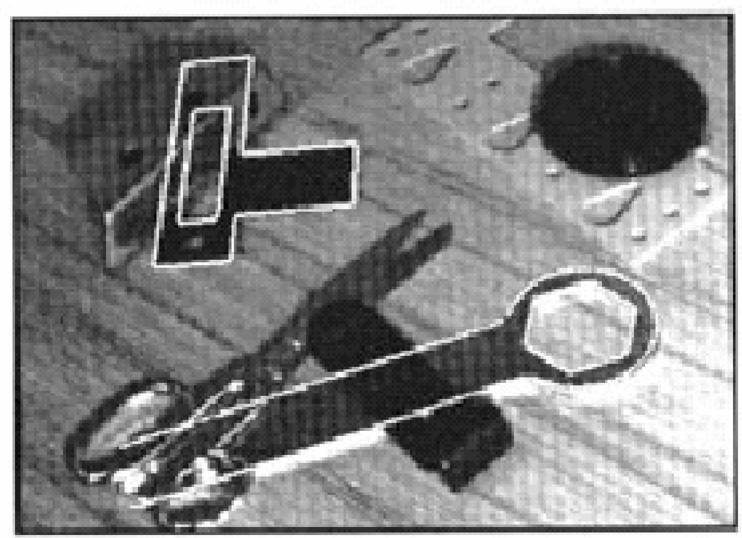
```
For each type T of invariant-bearing group
  For each image group G of type T

  Determine the values V of the invariants of G

    For each model feature group M of type T whose invariants
    have the values V

      Determine the transformation that takes M to G

      Render the model using this transformation

      Compare the result with the image, and accept if
      similar
    end
  end
end
```

Recognizing planar objects using invariants.

Input image

Edge points fitted with lines or conics





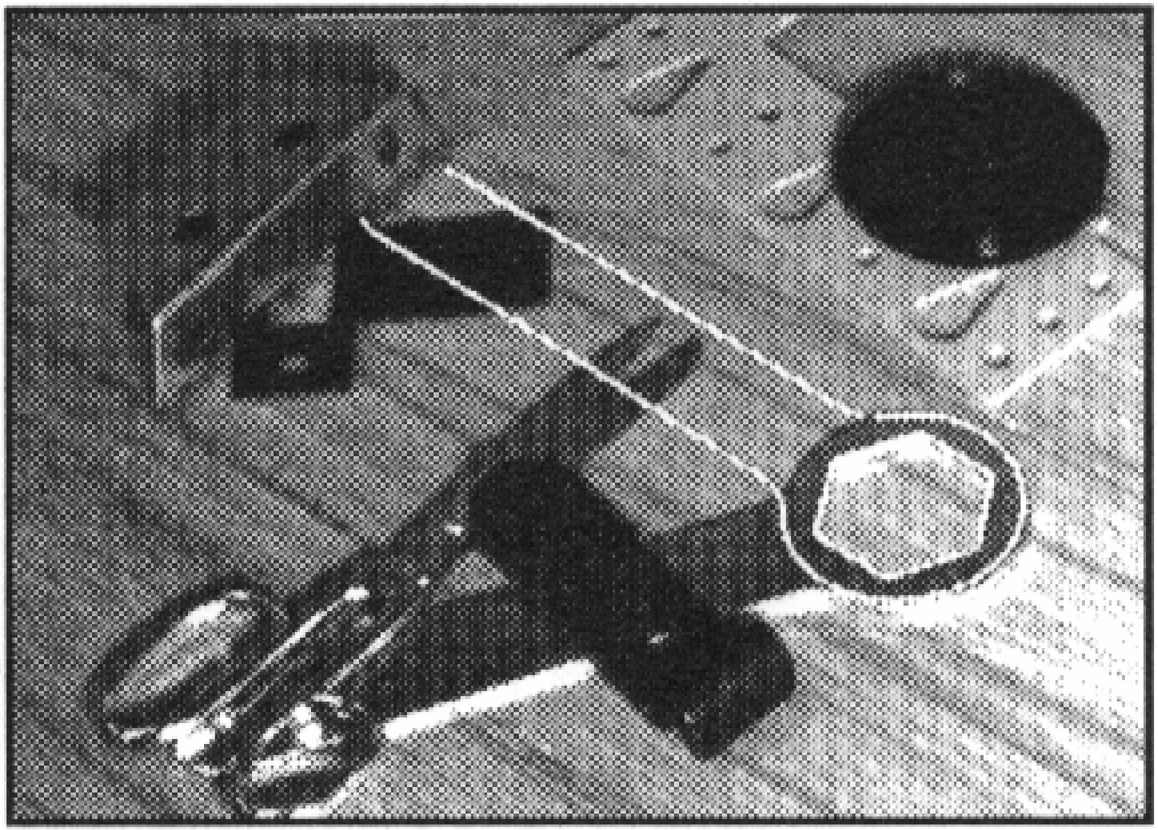
Objects that have been recognized and verified.

From Rothwell et al, CVPR 92.

Verification

- Edge score
  - are there image edges near predicted object edges?
  - very unreliable; in texture, answer is usually yes
- Oriented edge score
  - are there image edges near predicted object edges with the right orientation?
  - better, but still hard to do well (see next slide)
- Texture largely ignored [Forsythe]
  - e.g. does the spanner have the same texture as the wood?

52% of the edge points for this candidate object were verified in the wood texture underneath.



Rothwell et al, CVPR 92.

Algorithm Sensitivity

- Geometric Hashing
  - A relatively sparse hash table is critical for good performance
  - Method is not robust for cluttered scenes (full hash table) or noisy data (uncertainty in hash values)
- Generalized Hough Transform
  - Does not scale well to multi-object complex scenes
  - Also suffers from matching uncertainty with noisy data

Grimson and Huttenlocher, 1990

[http://www.cs.princeton.edu/courses/archive/fall03/cs597D/lectures/rigid_registration.pdf]

Comparison to template matching

- Costs of template matching
  - 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations
  - Does not easily handle partial occlusion and other variation without large increase in template numbers
  - Viola & Jones cascade must start again for each qualitatively different template
- Costs of local feature approach
  - 3000 evaluations (reduction by factor of 10,000)
  - Features are more invariant to illumination, 3D rotation, and object variation
  - Use of many small subtemplates increases robustness to partial occlusion and other variations

[Lowe]

Model-based Vision

Topics:

- Hypothesize and test
  - Interpretation Trees
  - Alignment
- Hypothesis generation methods
  - Pose clustering
  - Invariances
  - Geometric hashing
- Verification methods