CS 534: Computer Vision
3D Model-based recognition

Ahmed Elgammal
Dept of Computer Science
Rutgers University

High Level Vision

- Object Recognition: What it means ?
- Two main recognition tasks:
  - Categorization: what is the class of the object: face, car, motorbike, airplane, etc.
  - Identification: "John's face", "my car"
- Which of the two tasks is easier and which comes first?
- The answers from neuroscience and computer vision are different:
  - Computer vision: identification is relatively easy. Categorization very hard.
  - Psychologists and neuroscientists: in biological visual systems, categorization seems to be simpler and immediate stage in the recognition process.
- Two main schools:
  - Object centered approaches
  - View-based (image based) approaches.

Outlines

- Geometric Model-Based Object Recognition
- Choosing features
- Approaches for model-based vision
  – Obtaining Hypotheses by Pose Consistency
  – Obtaining Hypotheses by Pose Clustering
  – Obtaining Hypotheses by Using Invariants
- Geometric Hashing
- Affine invariant geometric hashing

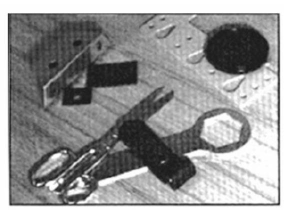
Geometric Model-Based Object Recognition

Object 1 Model

Object 2 Model

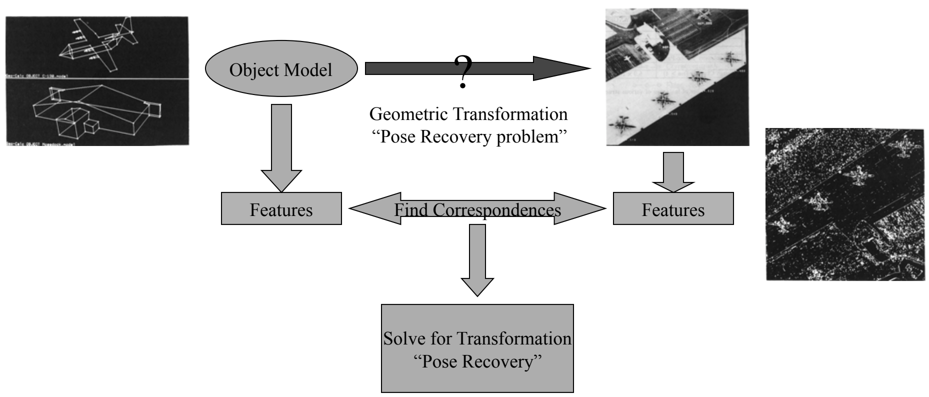
Object N Model

Modelbase
Geometric Models

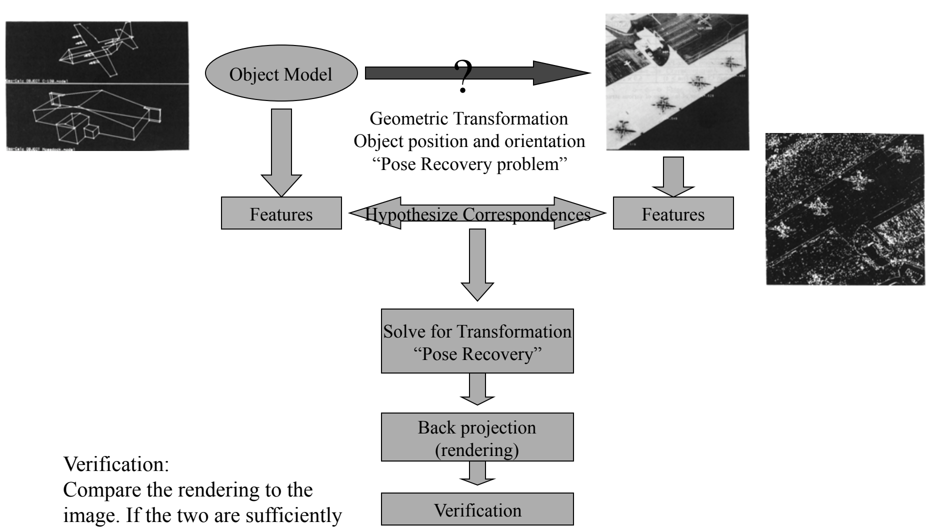


Image

Problem definitions: Given geometric object models (modelbase) and an image, find out which object(s) we see in the image.



Framework



Verification:
Compare the rendering to the image. If the two are sufficiently similar, accept the hypothesis

- General idea
  - Hypothesize object identity and pose
  - Recover camera
  - Render object in camera (widely known as backprojection)
  - Compare to image (verification)
- Issues
  - Where do the hypotheses come from?
  - How do we compare to image (verification)?

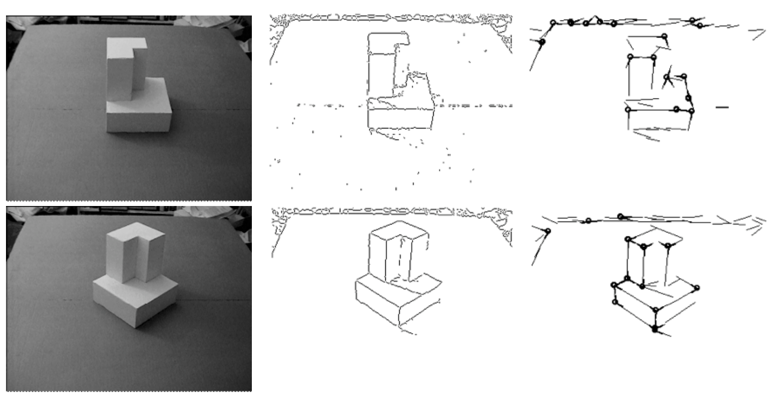
Choosing Features

- Given a 3-D object, how do we decide which points from its surface to choose as its model?
  - choose points that will give rise to detectable features in images
  - for polyhedra, the images of its vertices will be points in the images where two or more long lines meet
    - these can be detected by edge detection methods
  - points on the interiors of regions, or along straight lines are not easily identified in images.

Verification

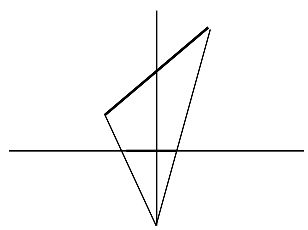
- We do not know anything about scene illumination.
- If we know scene illumination we can render close images…
- So, Verification should be robust to changes in illumination.
- Use object silhouettes rather than texture or intensities.
- Edge score
  - are there image edges near predicted object edges?
  - 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)
- No-one's used texture
  - e.g. does the spanner have the same texture as the wood?
- model selection problem
  - more on these later; no-ones seen verification this way, though

Example images



Choosing Features

- Example: why not choose the midpoints of the edges of a polyhedra as features
  - midpoints of projections of line segments are not the projections of the midpoints of line segments
  - if the entire line segment in the image is not identified, then we introduce error in locating midpoint



Obtaining Hypothesis

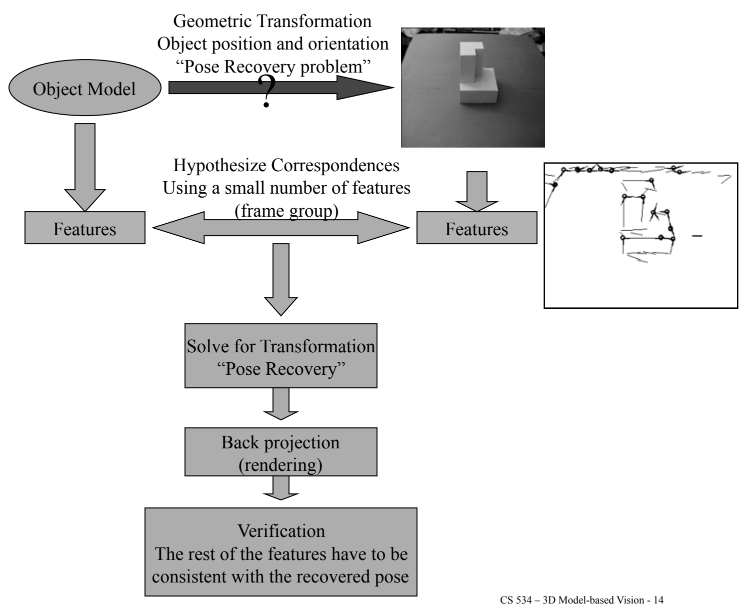
- M geometric features in the image
- N geometric features on each object
- L objects
- Brute Force Algorithm: Enumerate all possible correspondences
- $O(LM^N)$!!
- Instead: utilize geometric constraints
- small number of correspondences are needed to obtain the projection parameters, once these parameters are known the positions of the rest of all other features are known too.

Obtaining Hypotheses

Three basic approaches:

- Obtaining Hypotheses by Pose Consistency
- Obtaining Hypotheses by Pose Clustering
- Obtaining Hypotheses by Using Invariants

Pose Consistency



Pose consistency

- Also called Alignment: object is being aligned to the image
- Correspondences between image features and model features are not independent. – Geometric constraints
- A small number of correspondences yields missing camera parameters --- the others must be consistent with this.
- Typically intrinsic camera parameters are assumed to be known
- Strategy:
  - Generate hypotheses using small numbers of correspondences (e.g. triples of points for a calibrated perspective camera, etc., etc.)
  - Backproject and verify
- "frame groups": a group that can be used to yield a camera hypothesis.
- Example- for Perspective Camera:
  - three points;
  - three directions (trihedral vertex) and a point (necessary to establish scale)
  - Two directions emanating from a shared origin (dihedral vertex) and a point


```
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
```

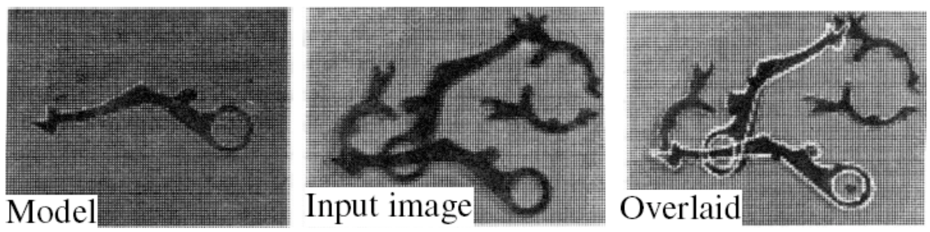


Figure from "Object recognition using alignment," D.P. Huttenlocher and S. Ullman, Proc. Int. Conf. Computer Vision, 1986, copyright IEEE, 1986

RANSAC – RANdom SAmple Consensus

- Searching for a random sample that leads to a fit on which many of the data points agree
- Extremely useful concept
- Can fit models even if up to 50% of the points are outliers.

Repeat

- Choose a subset of points randomly
- Fit the model to this subset
- See how many points agree on this model (how many points fit that model)
- Use only points which agree to re-fit a better model

Finally choose the best fit

RANSAC – RANdom SAmple Consensus

- Four parameters
  n : the smallest # of points required
  k : the # of iterations required
  t : the threshold used to identify a point that fits well
  d : the # of nearby points required

  Until k iterations have occurred
    Pick n sample points uniformly at random
    Fit to that set of n points
    For each data point outside the sample
      Test distance; if the distance < t, it is close
    If there are d or more points close, this is a good fit.
      Refit the line using all these points
  End
  use the best fit

Pose Clustering

- Most objects have many frame groups
- There should be many correspondences between object and image frame groups that verify satisfactorily.
- Each of these correspondences yield approximately the same estimate for the position and orientation of the object w.r.t. the camera
- Wrong correspondences will yield uncorrelated estimates

$\Rightarrow$ Cluster hypothesis before verification.

Vote for the pose: use an accumulator array that represents pose space for each object

```
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
```

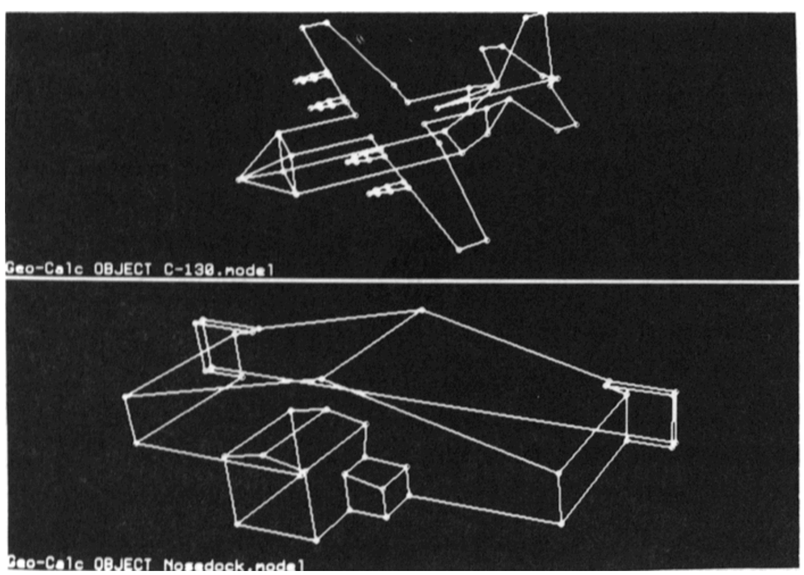



Figure from "The evolution and testing of a model-based object recognition system", J.L. Mundy and A. Heller, Proc. Int. Conf. Computer Vision, 1990 copyright 1990 IEEE

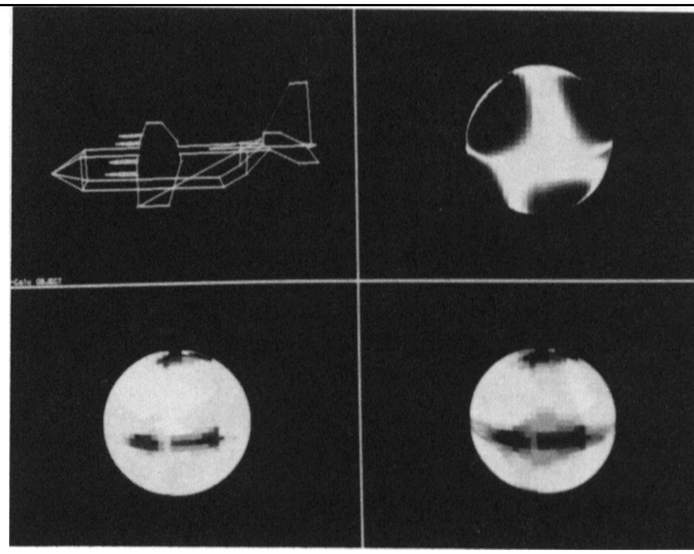


Figure from “The evolution and testing of a model-based object recognition system”, J.L. Mundy and A. Heller, Proc. Int. Conf. Computer Vision, 1990 copyright 1990 IEEE

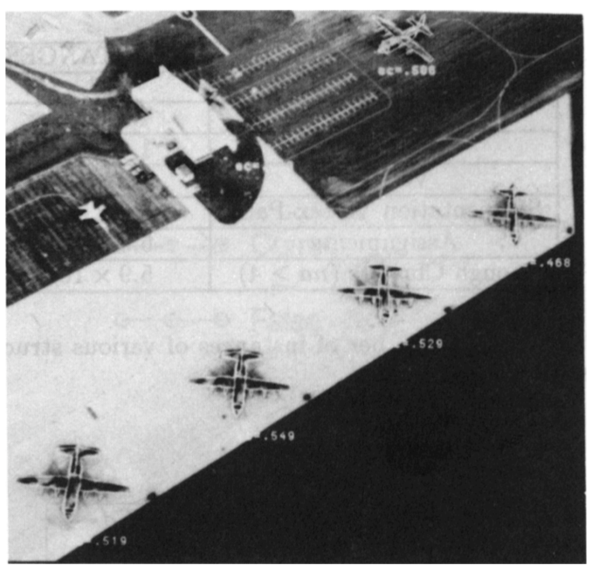


Figure from “The evolution and testing of a model-based object recognition system”, J.L. Mundy and A. Heller, Proc. Int. Conf. Computer Vision, 1990 copyright 1990 IEEE

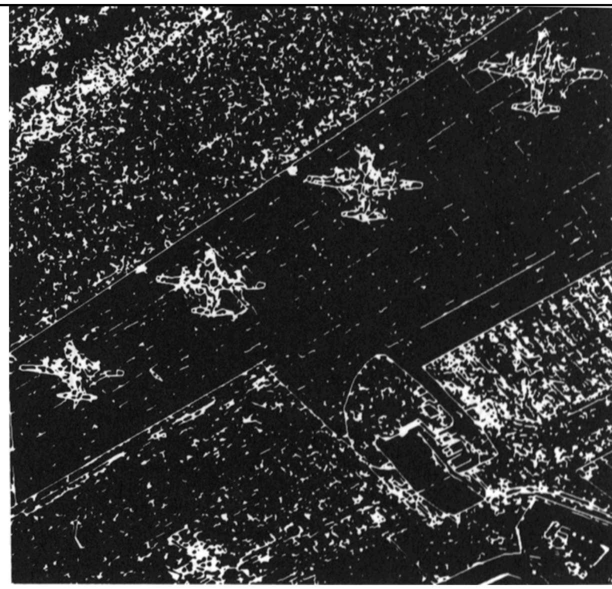


Figure from “The evolution and testing of a model-based object recognition system”, J.L. Mundy and A. Heller, Proc. Int. Conf. Computer Vision, 1990 copyright 1990 IEEE

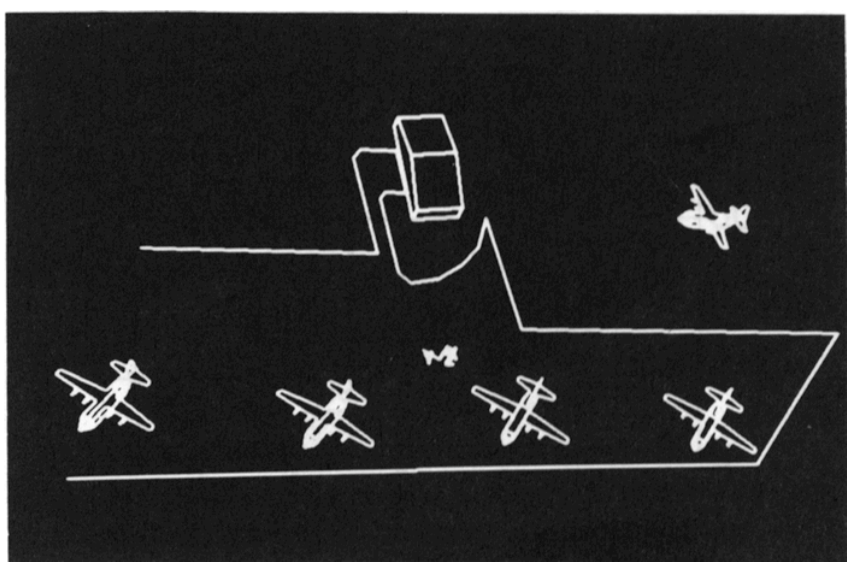


Figure from “The evolution and testing of a model-based object recognition system”, J.L. Mundy and A. Heller, Proc. Int. Conf. Computer Vision, 1990 copyright 1990 IEEE

Problems

- Correspondences should not be allowed to vote for poses from which they would be invisible
- In images with noise or texture – generates many spurious frame groups, the number of votes in the pose array corresponding to real objects may be smaller than the number of spurious votes
- Choosing the size of the bucket in the pose array is difficult:
  - too small mean no much accumulation,
  - too large mean too many buckets will have enough votes

Invariants

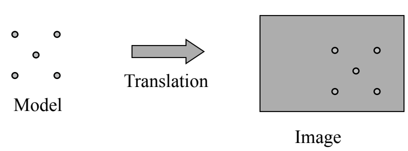
- There are geometric properties that are invariant to camera transformations

Geometric Hashing

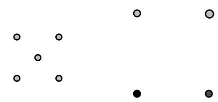
- A technique originally developed in computer vision for matching geometric features against a database of such features
- Widely used for pattern-matching, CAD/CAM, medical imaging
- Also used in other domains: molecular biology and medicinal chemistry !
- Matching is possible under transformations
- Matching is possible from patrial information
- efficient

Geometric hashing – basic idea

- Consider the following simple 2-D recognition problem.
  - We are given a set of object models, M_i
  - each model is represented by a set of points in the plane
    - $M_i = \{P_{i,1}, \ldots, P_{i,ni}\}$
  - We want to recognize instances of these point patterns in images from which point features (junctions, etc.) have been identified
    - So, our input image, B, is a binary image where the 1's are the feature points
    - We only allow the position of the instances of the Mi in B to vary - orientation is fixed. ***Only translation***
    - We want our approach to work even if some points from the model are not detected in B.
    - We will search for all possible models simultaneously

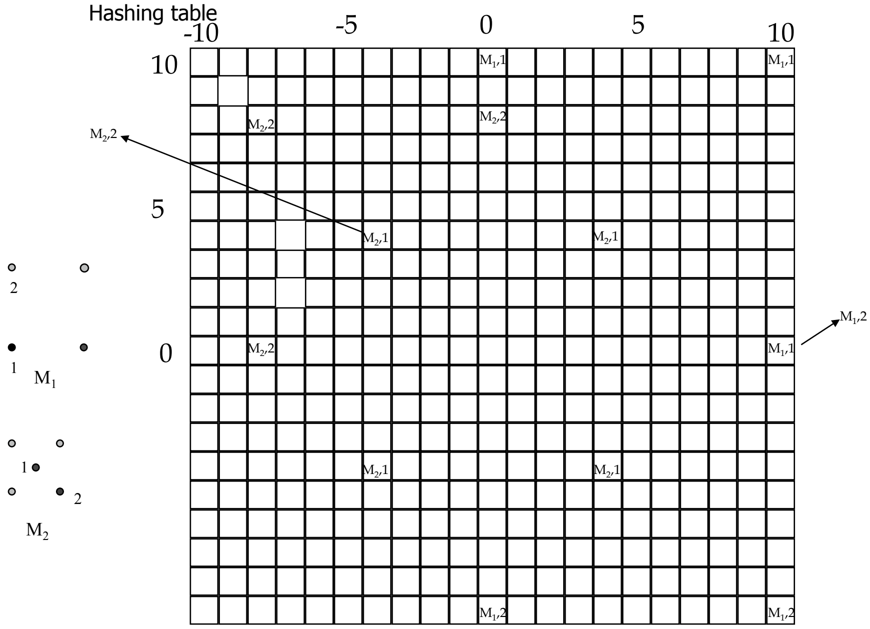


Geometric hashing



- Consider two models
  - M_1 = {(0,0), (10,0), (10,10), (0,10)}
  - M_2 = {(0,0), (4,4), (4,-4), (-4,-4), (-4,4)}
- We will build a table containing, for each model, all of the relative coordinates of these points given that one of the model points is chosen as the origin of its coordinate system
  - this point is called a basis, because choosing it completely determines the coordinates of the remaining points.
  - examples for M_1
    - choose (0,0) as basis obtain (10,0), (10,10), (0,10)
    - choose (10,0) as basis obtain (-10,0), (0,10), -10,10))

Hashing table



Hash table creation

- How many entries do we need to make in the hash table.
  - Mode M_i has n_i point
    - each has to be chosen as the basis point
    - coordinates of remaining n_i -1 points computed with respect to basis point
    - entry (M_i, basis) entered into hash table for each of those coordinates
  - And this has to be done for each of the m models.
  - So complexity is mn^2 to build the table
    - But the table is built only once, and then used many times.

Using the table during recognition

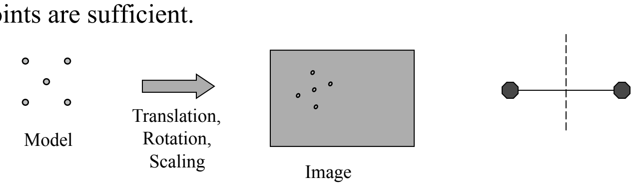
- Pick a feature point from the image as the basis.
  - the algorithm may have to consider all possible points from the image as the basis
- Compute the coordinates of all of the remaining image feature points with respect to that basis.
- Use each of these coordinates as an index into the hash table
  - at that location of the hash table we will find a list of (M_i, p_j) pairs - model basis pairs that result in some point from M_i getting these coordinates when the j'th point from M_i is chosen as the basis
  - keep track of the "score" for each (M_i, p_i) encountered
  - models that obtain high scores for some bases are recorded as possible detections

Some observations

- If the image contains n points from some model, M_i, then we will detect it n times
  - each of the n points can serve as a basis
  - for each choice, the remaining n-1 points will result in table indices that contain (M_i, basis)
- If the image contains s feature points, then what is the complexity of the recognition component of the geometric hashing algorithm?
  - for each of the s points we compute the new coordinates of the remaining s-1 points
  - and we keep track of the (model, basis) pairs retrieved from the table based on those coordinates
  - so, the algorithm has complexity $O(s^2)$, and is independent of the number of models in the database

Geometric Hashing

- Consider a more general case: translation, scaling and rotation
- one point is not a sufficient basis.
- But two points are sufficient.



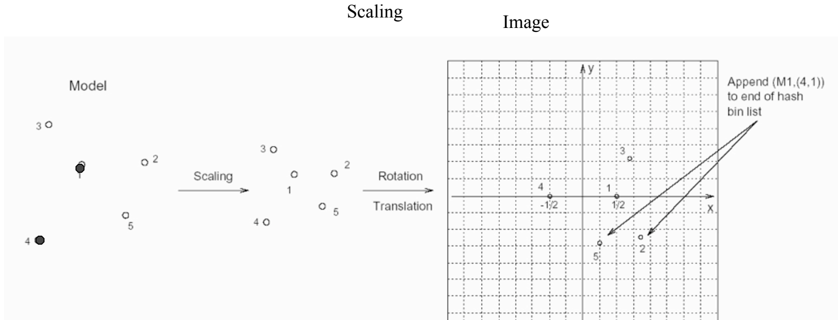
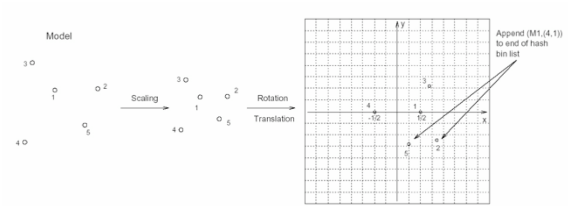


Figure from Haim J. Wolfson "Geometric Hashing an Overview" IEEE computational Science and Engineering, Oct 1996.



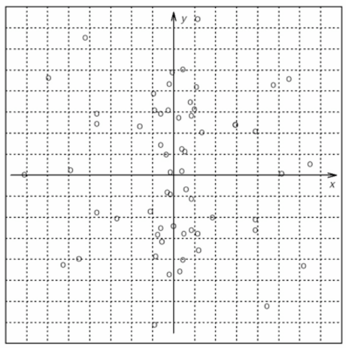
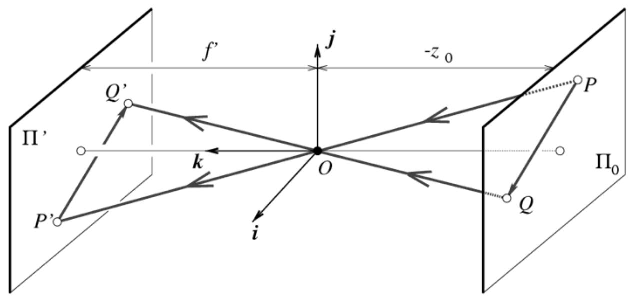


Figure from Haim J. Wolfson "Geometric Hashing an Overview" IEEE computational Science and Engineering, Oct 1996.

Revised geometric hashing

- Table construction
  - need to consider all pairs of points from each model
    - for each pair, construct the coordinates of the remaining n-2 points using those two as a basis
    - add an entry for (model, basis-pair) in the hash table
    - complexity is now mn^3
- Recognition
  - pick a pair of points from the image (cycling through all pairs)
  - compute coordinates of remaining point using this pair as a basis
  - look up (model, basis-pair) in table and tally votes

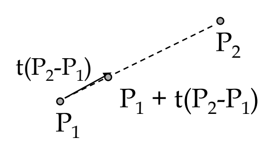
Recall: Affine projection models: Weak perspective projection
Also called Scaled Orthographic Projection SOP



$$\begin{cases} x' = f'\frac{x}{z} \\ y' = f'\frac{y}{z} \end{cases} \Rightarrow \boxed{\begin{cases} x' = -mx \\ y' = -my \end{cases}}$$ where $m = -\frac{f'}{z_0}$ is the magnification (scaling factor).

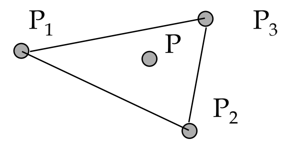
- When the scene depth is small compared its distance from the Camera, we can assume every thing is on one plane,
- m can be taken constant: weak perspective projection
- also called scaled orthography (every thing is a scaled version of the scene)

Affine combinations of points



- Let P_1 and P_2 be points
- Consider the expression $P = P_1 + t(P_2 - P_1)$
  - P represents a point on the line joining P_1 and P_2.
  - if 0 <= t <= 1 then P lies between P_1 and P_2.
  - We can rewrite the expression as $P = (1-t)P_1 + tP_2$
- Define an **affine combination** of two points to be
  - $a_1P_1 + a_2P_2$
  - where $a_1 + a_2 = 1$
  - $P = (1-t)P_1 + tP_2$ is an affine combination with $a_2 = t$.

Affine combinations



- Generally,
  - if P_1, ...P_n is a set of points, and $a_1 + \ldots + a_n = 1$, then
  - $a_1 P_1 + \ldots a_n P_n$ is the point $P_1 + a_2(P_2\text{-}P_1) + \ldots a_n(P_n\text{-}P_1)$
- Let's look at affine combinations of three points. These are points
  - $P = a_1P_1 + a_2P_2 + a_3P_3 = P_1 + a_2(P_2\text{-}P_1) + a_3(P_3\text{-}P_1)$
  - where $a_1 + a_2 + a_3 = 1$
  - if $0 <= a_1, a_2, a_3, <=1$ then P falls in the triangle, otherwise outside
  - (a_2, a_3) are the affine coordinates of P
    - "homogeneous" representation is $(1,a_2,a_3)$
  - P_1, P_2, P_3 is called the affine basis

Affine combinations

- Given any two points, $P = (a_1, a_2)$ and $Q = (a'_1, a'_2)$
  - Q-P is a vector
  - its affine coordinates are $(a'_1\text{-}a_1, a'_2\text{-}a_2)$
  - Note that the affine coordinates of a point sum to 1, while the affine coordinates of a vector sum to 0.

Affine transformations

- Rigid transformations are of the form

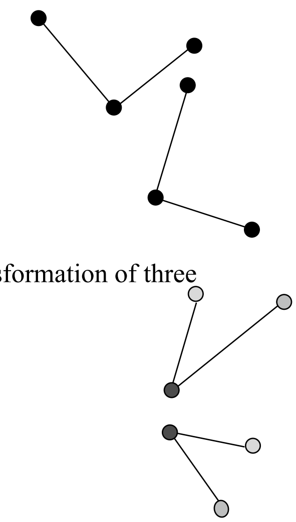
$$[x', y'] = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

where $a^2 + b^2 = 1$

- An affine transformation is of the form

$$[x', y'] = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

for arbitrary a,b,c,d and is determined by the transformation of three points



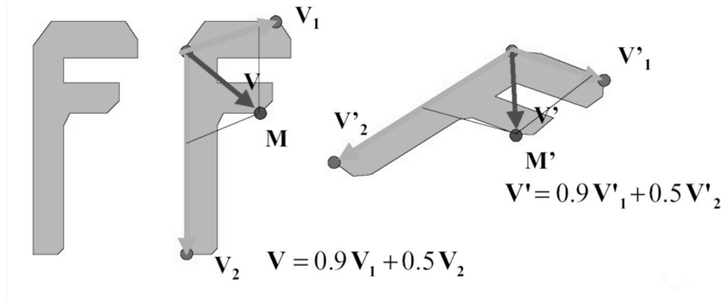
Representation in homogeneous coordinates

- In homogeneous coordinates, this transformation is represented as:

$$[x', y', 1] = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine coordinates and affine transformations

- The affine coordinates of a point are unchanged if the point and the affine basis are subjected to the same affine transformation
- Based on simple properties of affine transformations
- Let T be an affine transformation
  - $T(P_1 - P_2) = TP_1 - TP_2$
  - $T(aP) = aTP$, for any scalar a.



Proof

- Let $P_1 = (x,y,1)$ and $P_2 = (u,v,1)$
  - Note that $P_1 - P_2$ is a vector
- $TP_1 = (ax + by + t_x, cx + dy + t_y, 1)$
- $TP_2 = (au + bv + t_x, cu + dv + t_y, 1)$
  - $TP_1 - TP_2 = (a(x-u) + b(y-v), c(x-u) + d(y-v), 0)$
- $P_1 - P_2 = (x-u, y-v, 0)$
- $T(P_1-P_2) = (a(x-u) + b(y-v), c(x-u)+d(y-v), 0)$

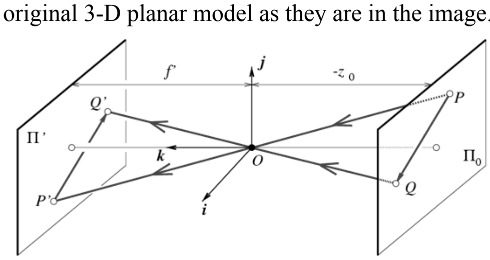
Invariance for Geometric hashing

Affine Coordinates represent an invariant under Affine Transformation. So can be used for Geometric hashing

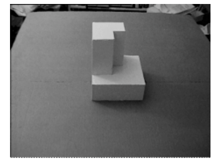
- Let $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$ be an ordered affine basis triplet in the plane.
- Then the affine coordinates (α,β) of a point P are:
  - $P = \alpha(\mathbf{P_2} - \mathbf{P_1}) + \beta\,(\mathbf{P_3} - \mathbf{P_1}) + \mathbf{P_1}$
- Applying any affine transformation T will transform it to
  - $TP = \alpha(T\mathbf{P_2} - T\mathbf{P}_1) + \beta\,(T\mathbf{P_3} - T\,\mathbf{P_1}) + T\mathbf{P}_1$
- So, TP has the same coordinates (α,β) in the basis triplet as it did originally.

What do affine transformations have to do with 3-D recognition

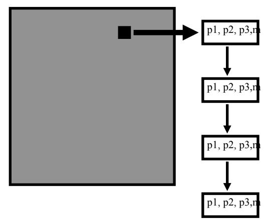
- Suppose our point pattern is a planar pattern - i.e., all of the points lie on the same plane.
  - we construct out hash table using these coordinates, choosing three at a time as a basis
- We position and orient this planar point pattern in space <u>far from the camera,</u> (So SOP is a good model) and take its image.
  - the transformation of the model to the image is an affine transformation
  - so, the affine coordinates of the points in any given basis are the same in the original 3-D planar model as they are in the image.



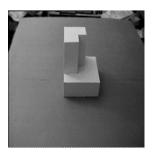
Preprocessing

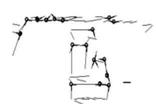


- Suppose we have a model containing m feature points
- For each ordered noncollinear triplet of model points
  - compute the affine coordinates of the remaining m-3 points using that triple as a basis
  - each such coordinate is used as an entry into a hash table where we record the (**base triplet,model)** at which this coordinate was obtained.
- Complexity is m^4 per model.

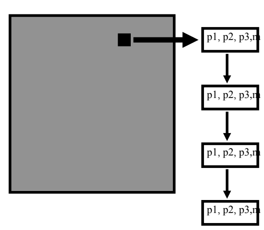


Recognition





- Scene with n interest points
- Choose an ordered triplet from the scene
  - compute the affine coordinates of remaining n-3 points in this basis
  - for each coordinate, check the hash table and for each entry found, tally a vote for the (basis triplet, model)
  - if the triplet scores high enough, we verify the match
- If the image does not contain an instance of any model, then we will only discover this after looking at all n^3 triples.



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

```
For all groups of three image points T(I)
  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
```


Pose Recovery Applications

- In many applications recovering pose is far more important than recognition.
- We know what we are looking at but we need to get accurate measurements of pose
- Examples: Medical applications.

Sources

- Forsyth and Ponce, Computer Vision a Modern approach: chapter 18.
- Slides by D. Forsyth @ UC Berkeley
- Slides by L.S. Davis @ UMD
- Haim J. Wolfson "Geometric Hashing an Overview" IEEE computational Science and Engineering, Oct 1996. (posted on class web page)