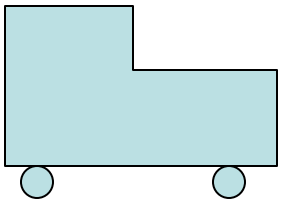
Matching in 2D
Readings: Ch. 11: Sec. 11.1-11.6

- alignment
- point representations and 2D transformations
- solving for 2D transformations from point correspondences
- some methodologies for alignment
    - local-feature-focus method
    - geometric hashing
- relational matching
- example system: RIO with 2D images of 3D objects

Specific Object Recognition: Matching in 2D



engine model

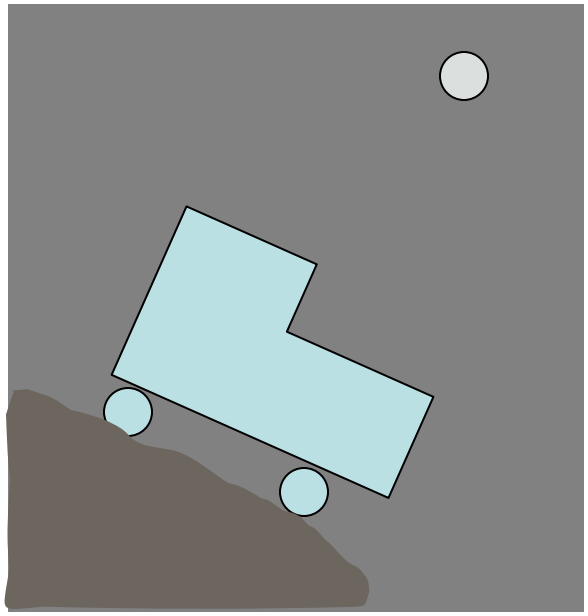


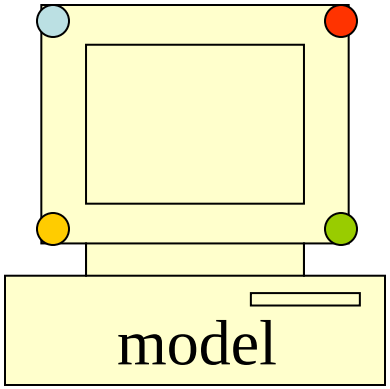
image containing an instance of the model

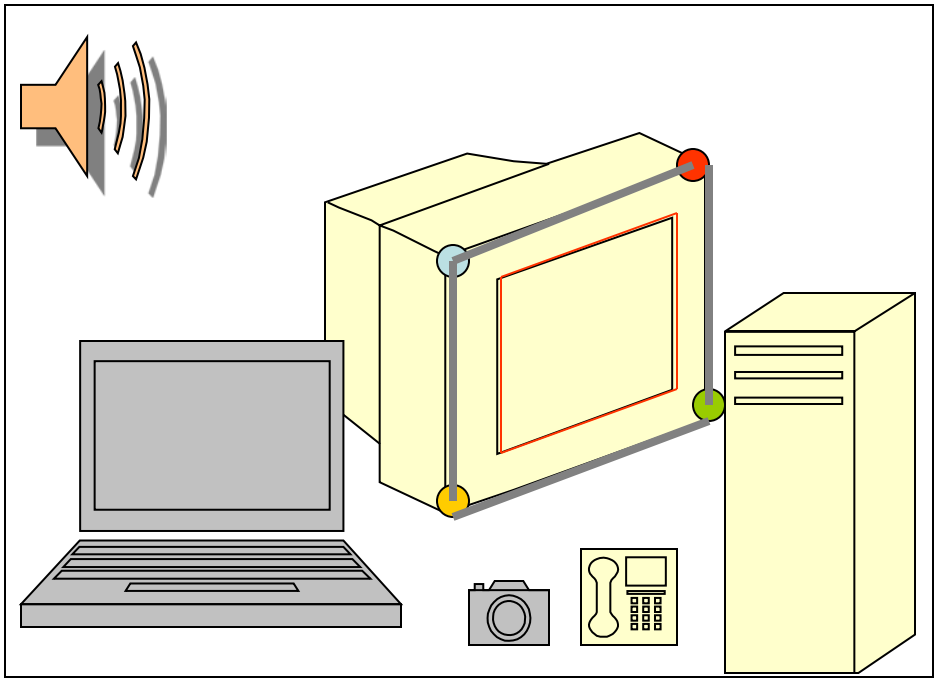
Is there an engine in the image?
If so, where is it located?

Alignment

- Use a geometric feature-based model of the object.
- Match features of the object to features in the image.
- Produce a hypothesis h (matching features)
- Compute an affine transformation T from h
- Apply T to the features of the model to map the model features to the image.
- Use a verification procedure to decide how well the model features line up with actual image features

Alignment



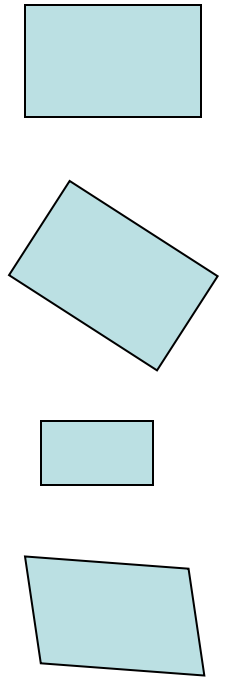


image

How can the object in the image differ from that in the model?

Most often used:
2D Affine Transformations

1. translation
2. rotation
3. scale
4. skew



Point Representation and Transformations

Normal Coordinates for a 2D Point

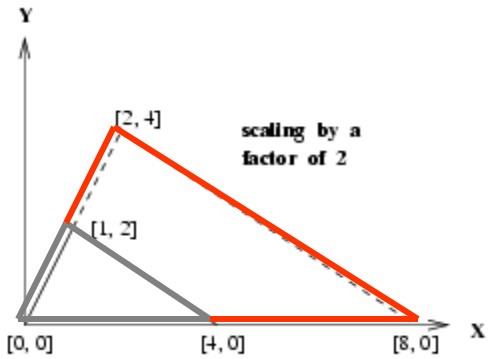
$$P = [x, y]^t = \begin{pmatrix} x \\ y \end{pmatrix}$$

Homogeneous Coordinates

$P = [sx, sy, s]^t$ where s is a scale factor

Scaling

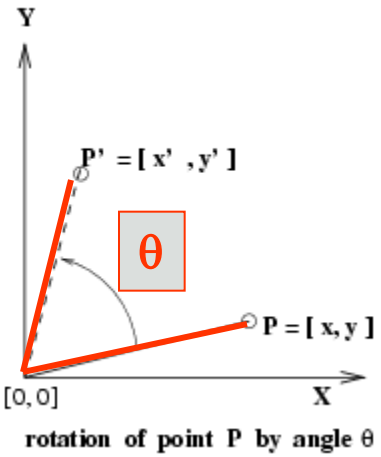
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} c_x & 0 \\ 0 & c_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_x * x \\ c_y * y \end{bmatrix}$$

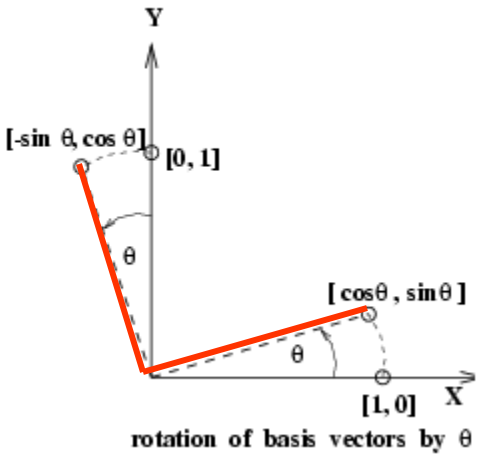


scaling by
a factor of 2
about (0,0)

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$





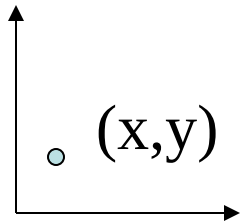
rotate point

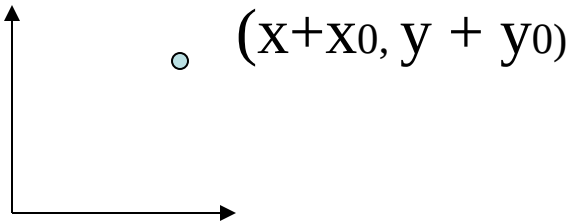
rotate axes

Translation

2 X 2 matrix doesn't work for translation!
Here's where we need homogeneous coordinates.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + x_0 \\ y + y_0 \\ 1 \end{pmatrix}$$



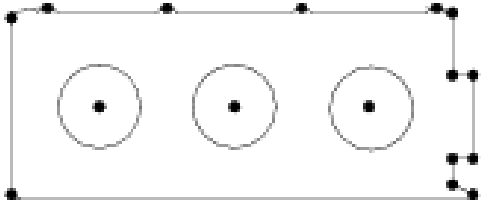


Rotation, Scaling and Translation

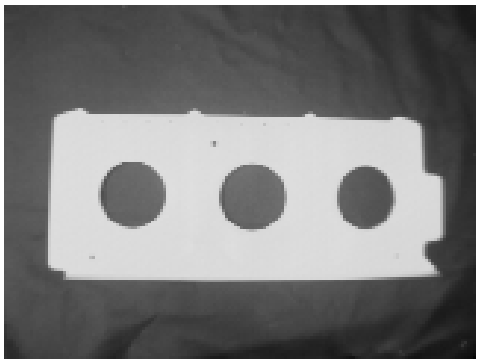
$$\begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \end{pmatrix} \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{T_R} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

T S R

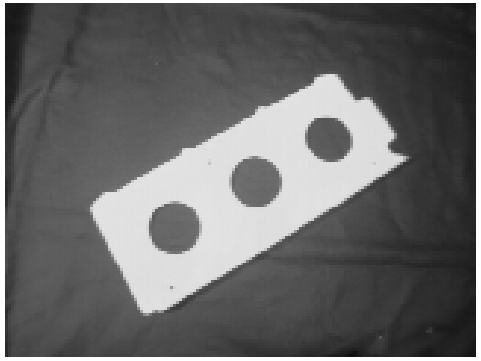
2D Model and 3 Matching Images of a Boeing Airplane Part



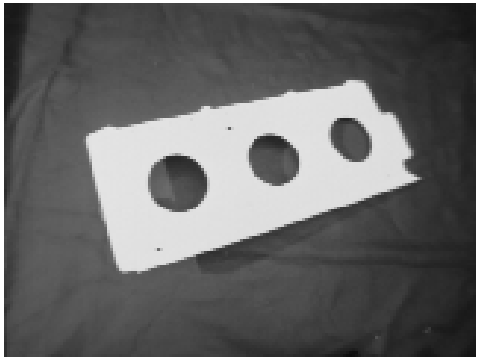
a) Part Model



b) Horizontal Image

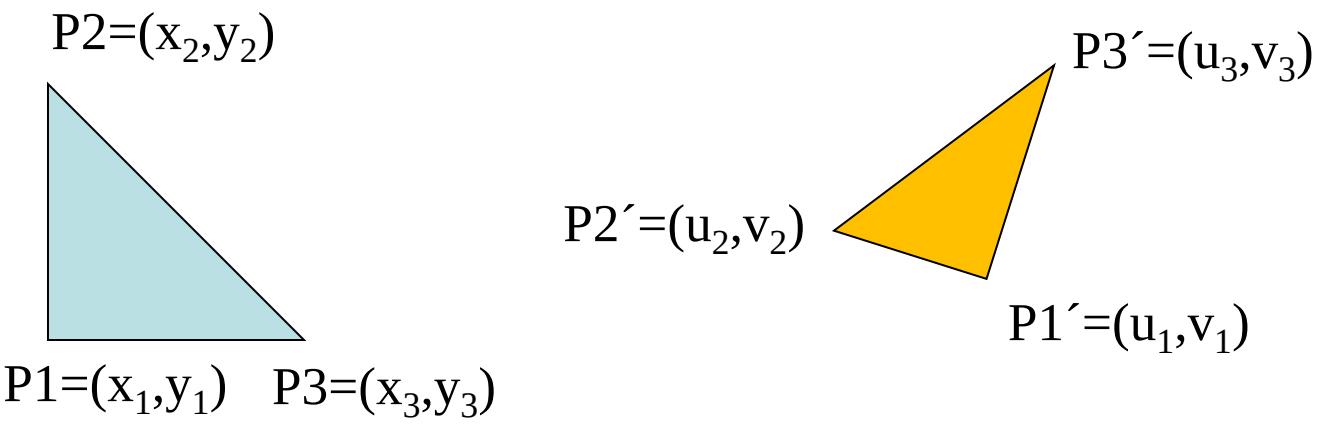


c) Rotated Image



d) Rotated and Skewed Image

Computing Affine Transformations between Sets of Matching Points



Given 3 matching pairs of points, the affine transformation can be computed through solving a simple matrix equation.

$$\begin{pmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}$$

A More Robust Approach

Using only 3 points is dangerous, because if even one is off, the transformation can be far from correct.

Instead, use many (n =10 or more) pairs of matching control points to determine a least squares estimate of the six parameters of the affine transformation.

$$\text{Error}(a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}) = \sum_{j=1,n} ((a_{11}*x_j + a_{12}*y_j + a_{13} - u_j)^2 + (a_{21}*x_j + a_{22}*y_j + a_{23} - v_j)^2)$$

The Equations to Solve

$$\varepsilon(a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}) = \sum_{j=1}^{n}((a_{11}x_j + a_{12}y_j + a_{13} - u_j)^2 + (a_{21}x_j + a_{22}y_j + a_{23} - v_j)^2) \tag{11.16}$$

Taking the six partial derivatives of the error function with respect to each of the six variables and setting this expression to zero gives us the six equations represented in matrix form in Equation 11.17 .

$$\begin{bmatrix} \Sigma {x_j}^2 & \Sigma x_j y_j & \Sigma x_j & 0 & 0 & 0 \\ \Sigma x_j y_j & \Sigma y_j^2 & \Sigma y_j & 0 & 0 & 0 \\ \Sigma x_j & \Sigma y_j & \Sigma 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Sigma {x_j}^2 & \Sigma x_j y_j & \Sigma x_j \\ 0 & 0 & 0 & \Sigma x_j y_j & \Sigma y_j^2 & \Sigma y_j \\ 0 & 0 & 0 & \Sigma x_j & \Sigma y_j & \Sigma 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} \Sigma u_j x_j \\ \Sigma u_j y_j \\ \Sigma u_j \\ \Sigma v_j x_j \\ \Sigma v_j y_j \\ \Sigma v_j \end{bmatrix} \tag{11.17}$$

What is this for?

- Many 2D matching techniques use it.
- Example: Geometric Hashing

Geometric Hashing

- This method was developed for the case where there is a whole database of models to try to find in an image.

- It trades:

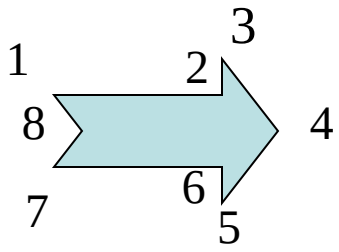
  a large amount of offline preprocessing and
  a large amount of space

- for potentially fast online

  object recognition
  pose detection

Theory Behind Geometric Hashing

- A model M is a an ordered set of feature points.

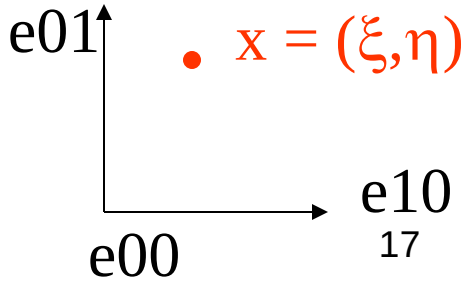


M = <P1,P2,P3,P4,P5,P6,P7,P8>

- An affine basis is any subset E={e00,e01,e10} of noncollinear points of M.

- For basis E, any point x ∈ M can be represented in affine coordinates (ξ,η).

x = ξ(e10 – e00) + η(e01-e00) + e00



Affine Transform

If x is represented in affine coordinates (ξ,η).

$$\mathbf{x = \xi(e10 - e00) + \eta(e01 - e00) + e00}$$

and we apply affine transform T to point x, we get

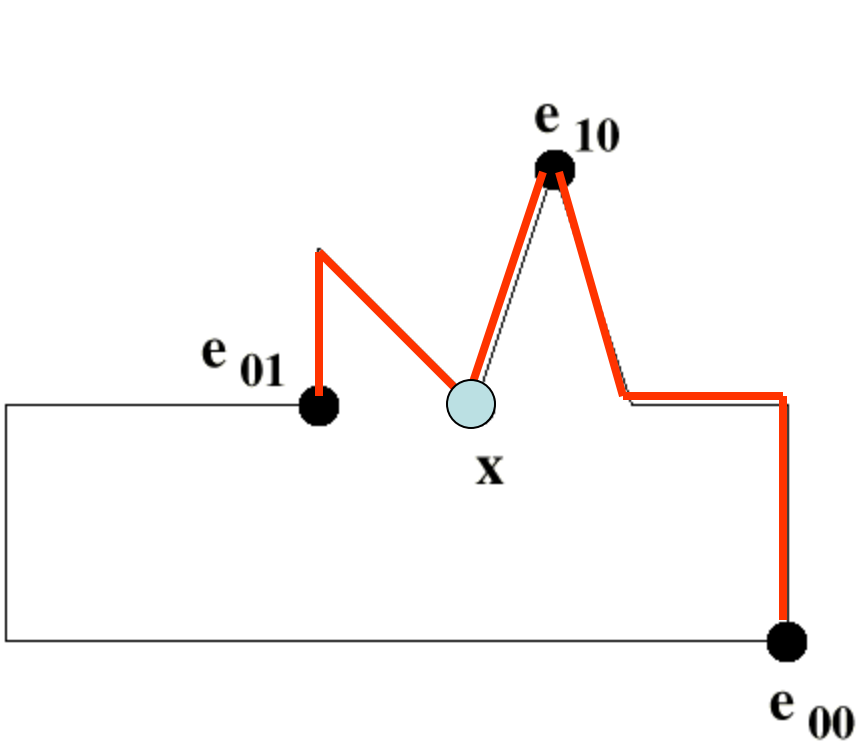
$$\mathbf{Tx = \xi(Te10 - Te00) + \eta(Te01 - Te00) + Te00}$$

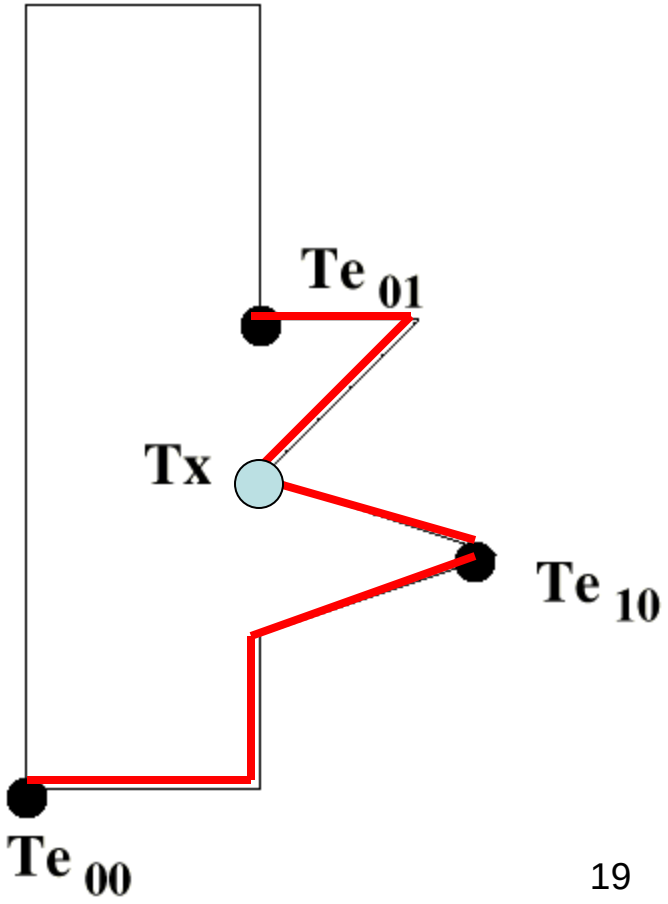
In both cases, x has the same coordinates (ξ,η).

Example

original object

transformed object



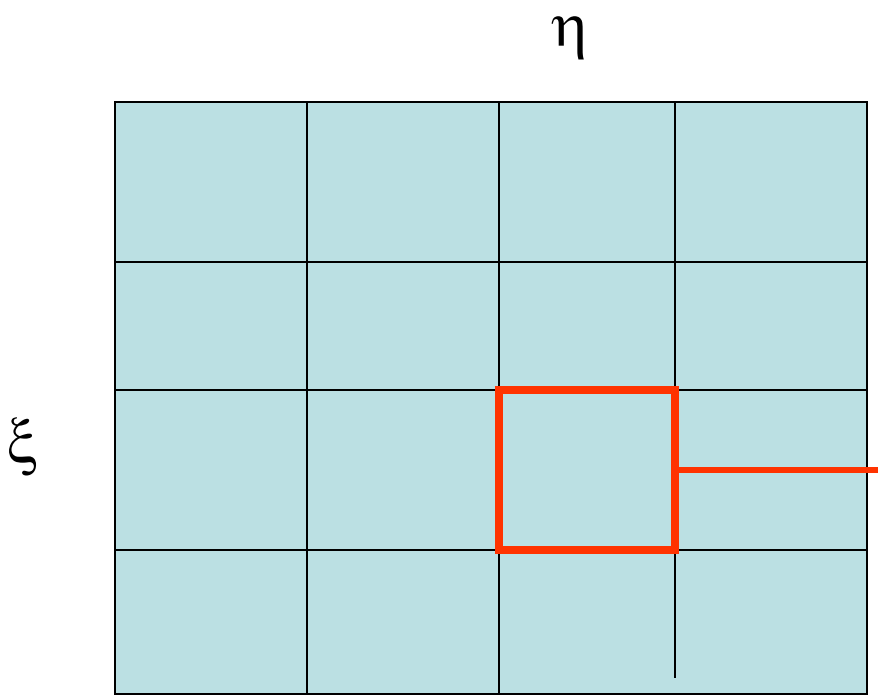


Offline Preprocessing

```
For each model M
  {
   Extract feature point set FM

   for each noncollinear triple E of FM  (basis)
        for each other point x of FM
           {
            calculate (ξ,η) for x with respect to E
            store (M,E) in hash table H at index (ξ,η)
            }
   }
```

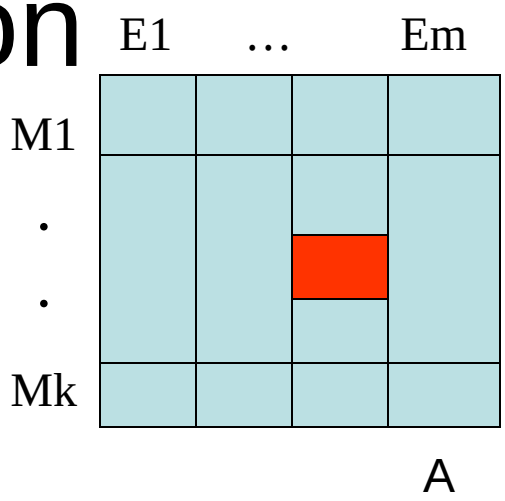
Hash Table



list of model / basis pairs that hash to bin

M1, E1
M2, E2
.
.
Mn, En

Online Recognition



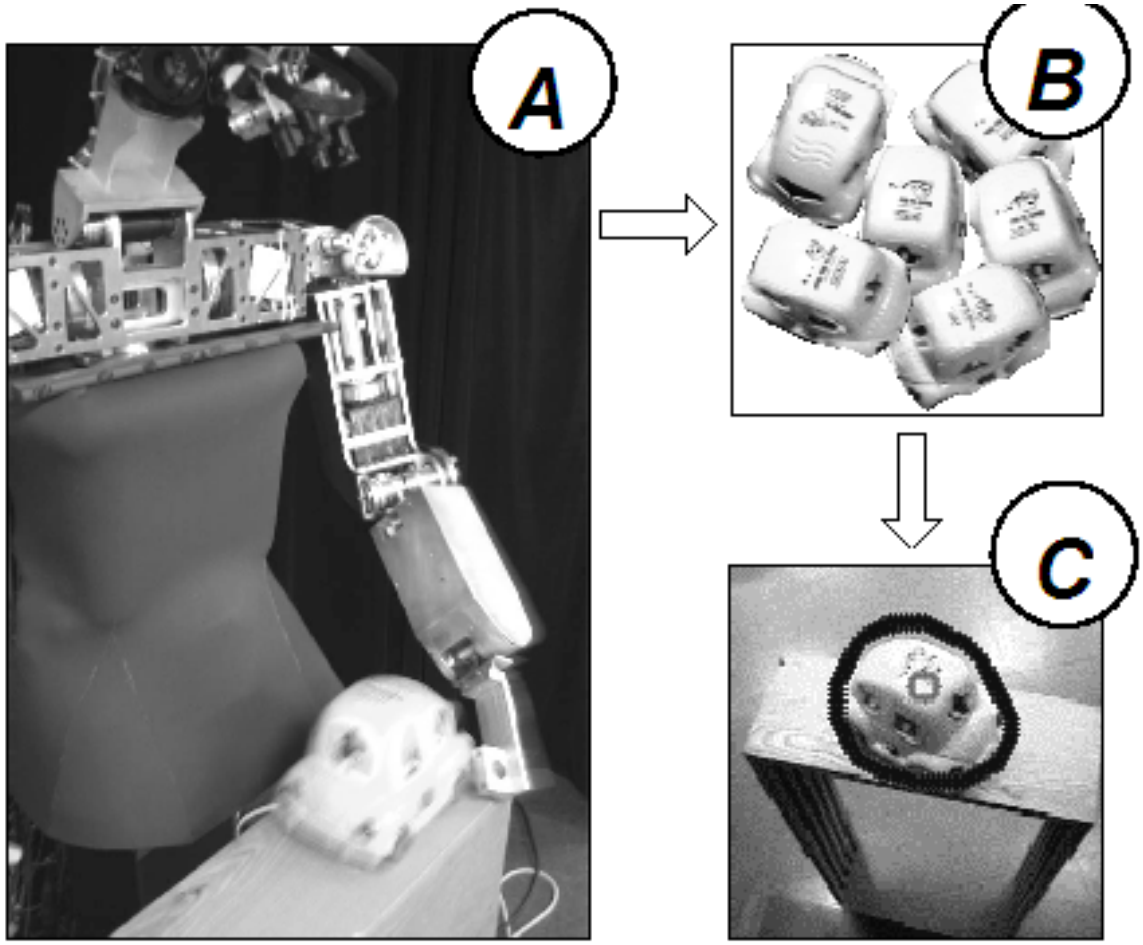
initialize accumulator A to all zero
extract feature points from image
for each basis triple F /* one basis */
  for each other point v /* each image point */
    {
     calculate (ξ,η) for v with respect to F
     retrieve list L from hash table at index (ξ,η)
     for each pair (M,E) of L
        A[M,E] = A[M,E] + 1
     }
    find peaks in accumulator array A
    for each peak (M,E) in A
       calculate and try to verify T э: F = TE

/*(M,E)->T*/

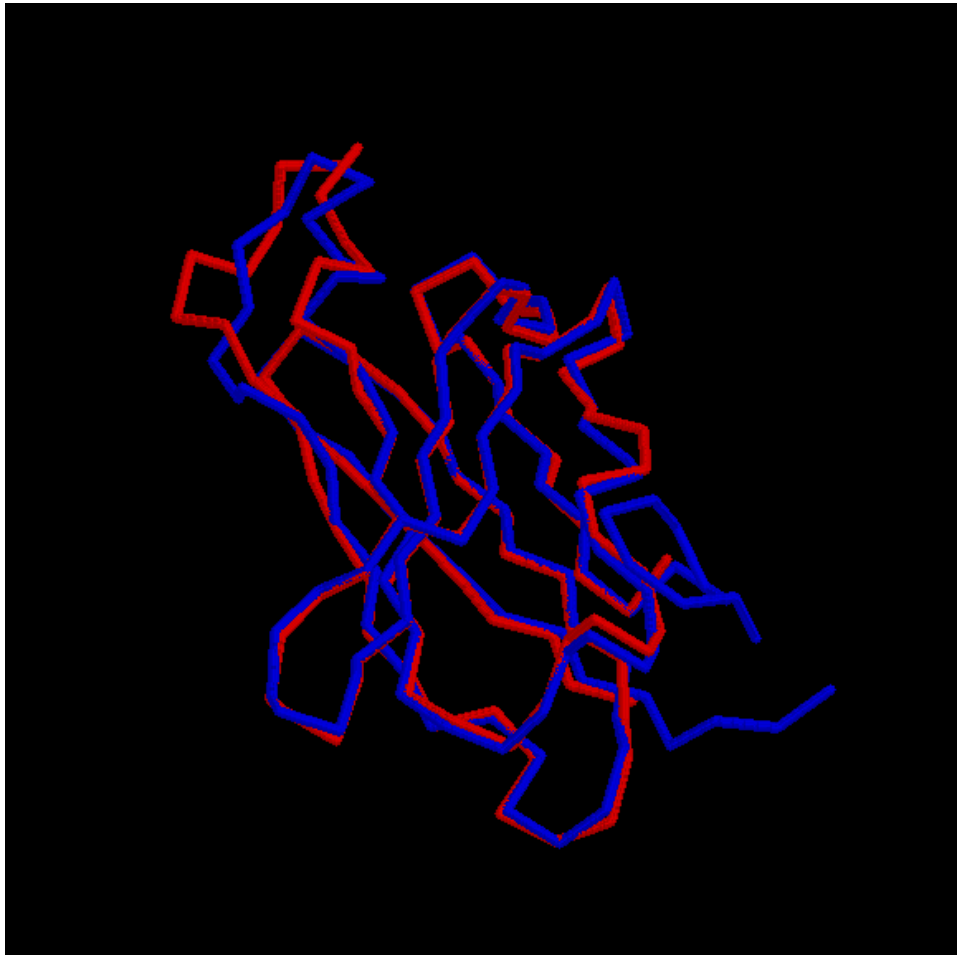
How do you verify?

- Original paper looked for just boundary points.
- We found that was not enough.
- The system could “hallucinate” objects that were not there, but had to pass through some of the same points.
- Best to look for the whole boundary and expect to see a significant portion of it.

Robotics Example (MIT)



Molecular Biology Example
Tel Aviv University
Protein Matching



2D Matching Mechanisms

- We can formalize the recognition problem as finding a mapping from model structures to image structures.

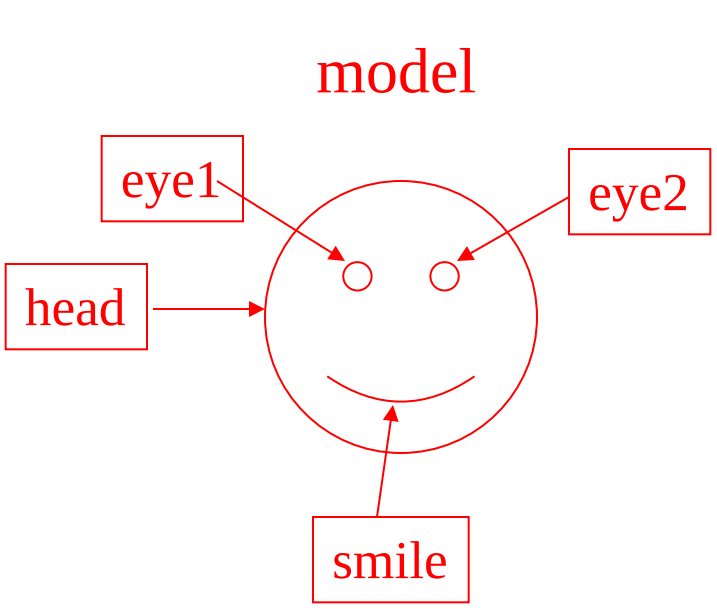
- Then we can look at different paradigms for solving it.

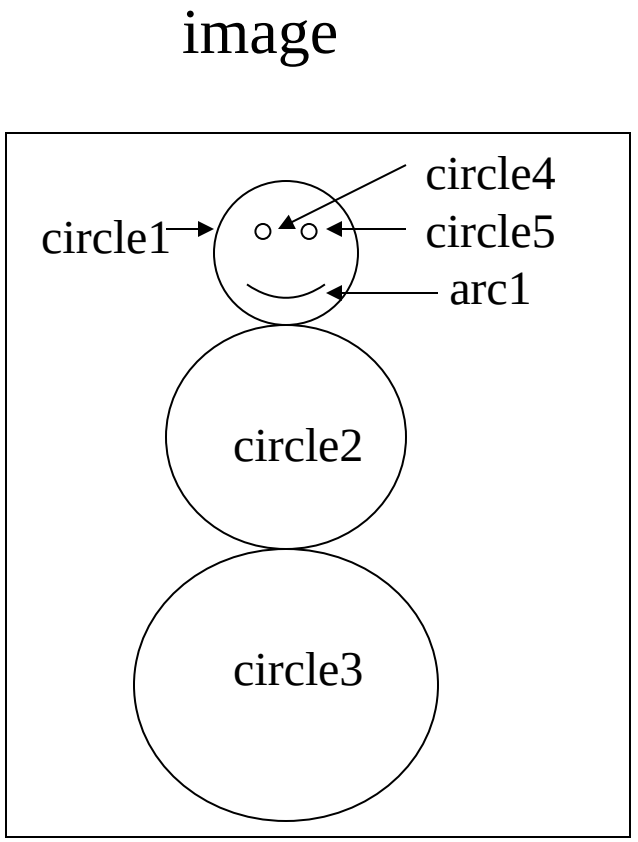
  - interpretation tree search
  - discrete relaxation
  - relational distance

Formalism

- A part (unit) is a structure in the scene, such as a region or segment or corner.
- A label is a symbol assigned to identify the part.
- An N-ary relation is a set of N-tuples defined over a set of parts or a set of labels.
- An assignment is a mapping from parts to labels.

Example





What are the relationships?

What is the best assignment
of model labels to image features?

Consistent Labeling Definition

Given:

1. a set of units P
2. a set of labels for those units L
3. a relation R_P over set P
4. a relation R_L over set L

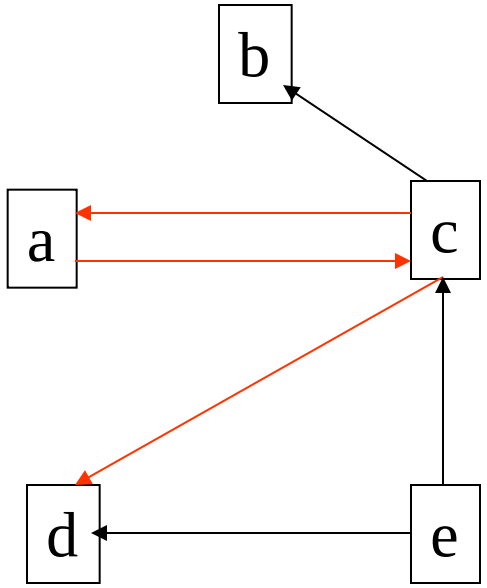
A **consistent labeling** f is a mapping **f: P -> L** satisfying

if $(p_i, p_j) \in R_P$, then $(f(p_i), f(p_j)) \in R_L$

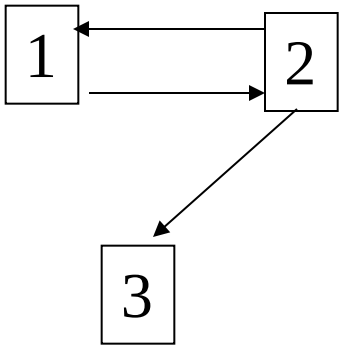
which means that a consistent labeling preserves relationships.

Abstract Example

binary relation R_L



binary relation R_P



$P = \{1,2,3\}$

$R_P = \{(1,2),(2,1),(2,3)\}$

$L = \{a,b,c,d,e\}$

$R_L = \{(a,c),(c,a),(c,b),(c,d),(e,c),(e,d)\}$

One consistent labeling is {(1,a),(2,c),(3,d)

House Example

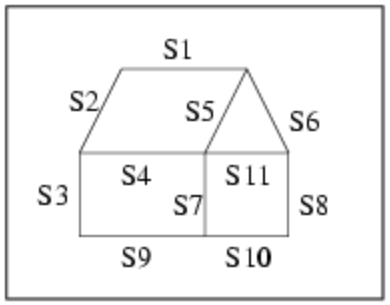


Image 1 **P**

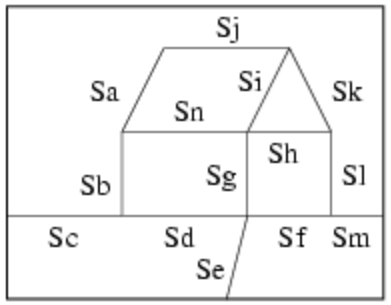


Image 2 **L**

RP and RL are connection relations.

P = {S1,S2,S3,S4,S5,S6,S7,S8,S9,S10,S11}.

L = {Sa,Sb,Sc,Sd,Se,Sf,Sg,Sh,Si,Sj,Sk,Sl,Sm}.

R_P = { (S1,S2), (S1,S5), (S1,S6), (S2,S3), (S2,S4), (S3,S4), (S3,S9), (S4,S5), (S4,S7), (S4,S11), (S5,S6), (S5,S7), (S5,S11), (S6,S8), (S6,S11), (S7,S9), (S7,S10), (S7,S11), (S8,S10), (S8,S11), (S9,S10) }.

R_L = { (Sa,Sb), (Sa,Sj), (Sa,Sn), (Sb,Sc), (Sb,Sd), (Sb,Sn), (Sc,Sd), (Sd,Se), (Sd,Sf), (Sd,Sg), (Se,Sf), (Se,Sg), (Sf,Sg), (Sf,Sl), (Sf,Sm), (Sg,Sh), (Sg,Si), (Sg,Sn), (Sh,Si), (Sh,Sk), (Sh,Sl), (Sh,Sn), (Si,Sj), (Si,Sk), (Si,Sn), (Sj,Sk), (Sk,Sl), (Sl,Sm) }.

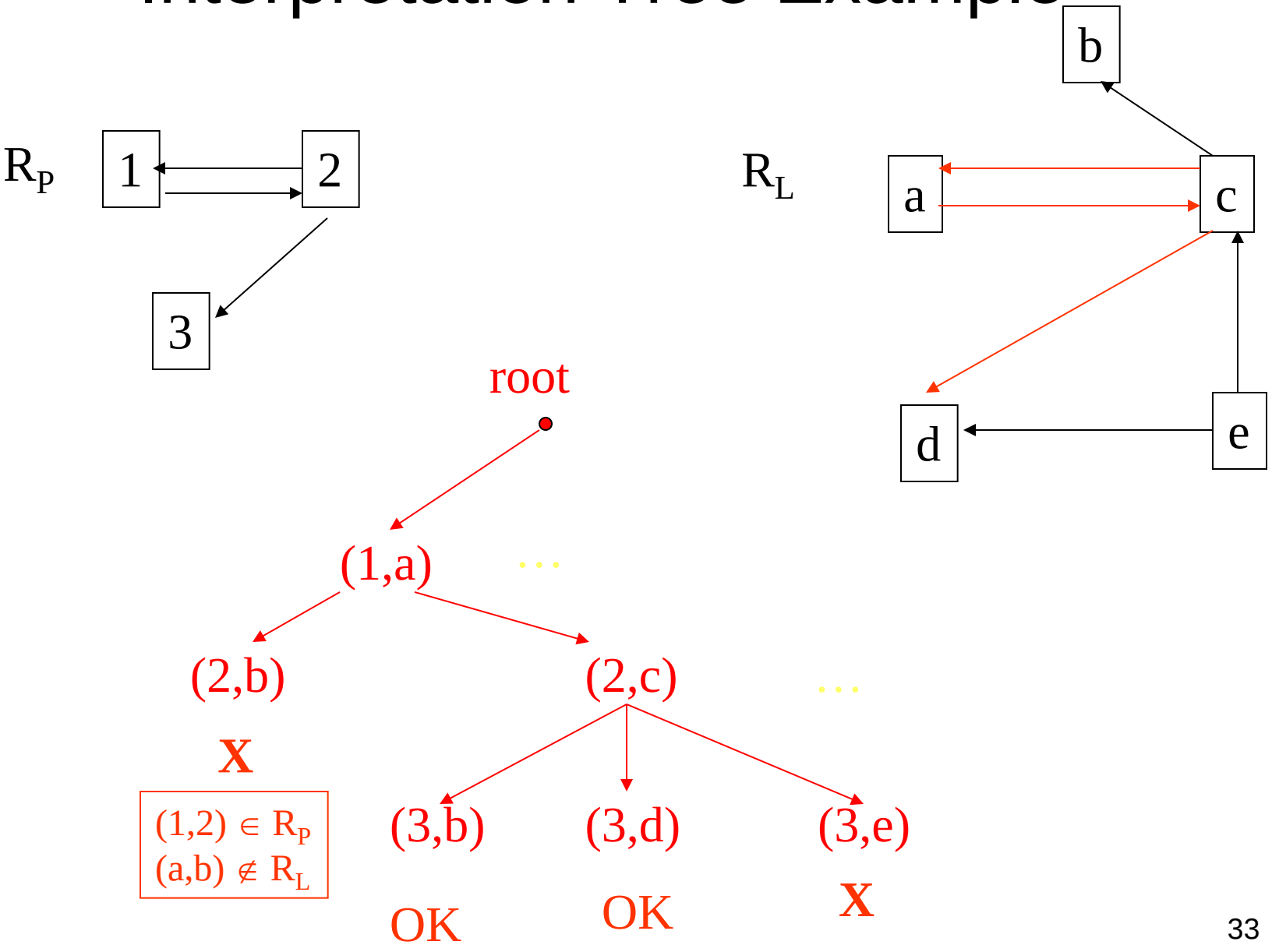
f(S1)=Sj
f(S2)=Sa
f(S3)=Sb
f(S4)=Sn
f(S5)=Si
f(S6)=Sk
f(S7)=Sg
f(S8) = Sl
f(S9)=Sd
f(S10)=Sf
f(S11)=Sh

1. Interpretation Tree

- An interpretation tree is a tree that represents all assignments of labels to parts.
- Each path from the root node to a leaf represents a (partial) assignment of labels to parts.
- Every path terminates as either

  1. a complete consistent labeling
  2. a failed partial assignment

Interpretation Tree Example



2. Discrete Relaxation

- Discrete relaxation is an alternative to (or addition to) the interpretation tree search.
- Relaxation is an iterative technique with polynomial time complexity.
- Relaxation uses local constraints at each iteration.
- It can be implemented on parallel machines.

How Discrete Relaxation Works

1. Each unit is assigned a set of initial possible labels.

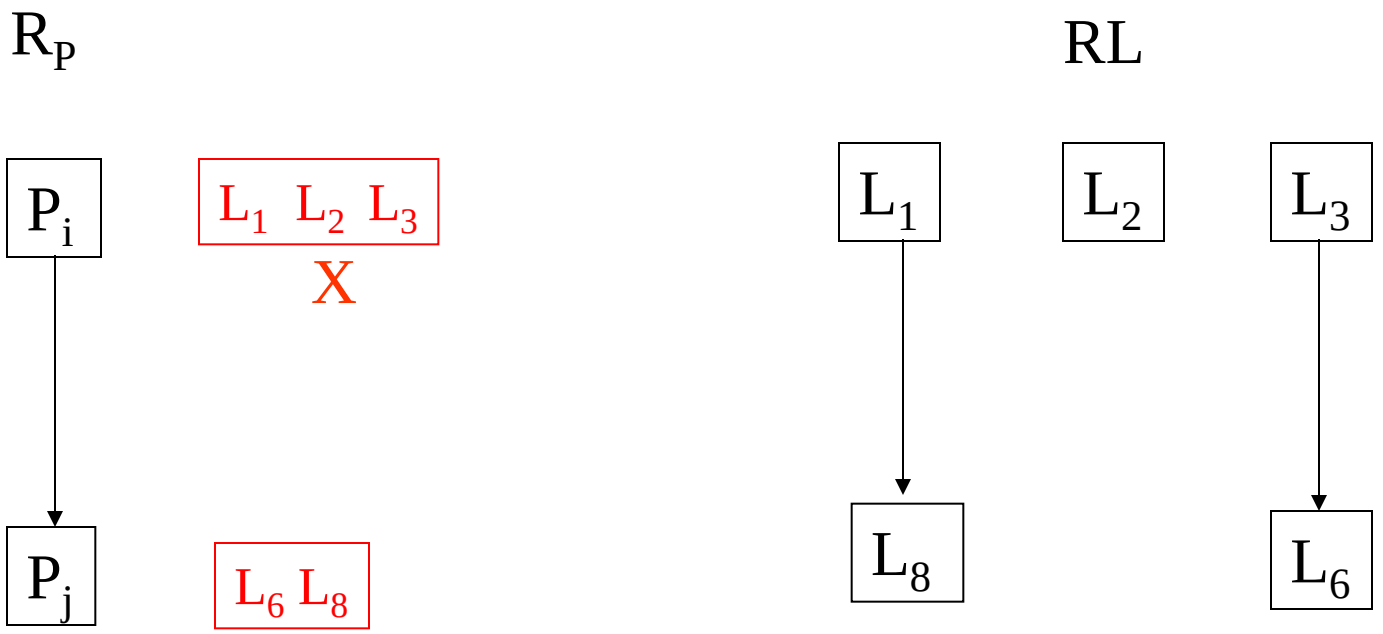
2. All relations are checked to see if some pairs of labels are impossible for certain pairs of units.

3. Inconsistent labels are removed from the label sets.

4. If any labels have been filtered out
   then another pass is executed
   else the relaxation part is done.

5. If there is more than one labeling left, a tree search can be used to find each of them.

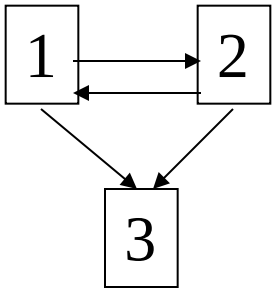
Example of Discrete Relaxation

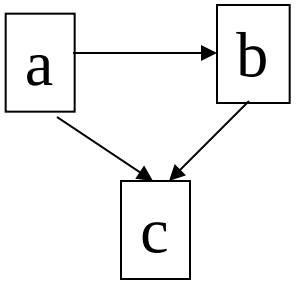


There is no label in P_j's label set that is connected to L_2 in P_i's label set. L_2 is inconsistent and filtered out.

3. Relational Distance Matching

- A fully consistent labeling is unrealistic.
- An image may have missing and extra features; required relationships may not always hold.
- Instead of looking for a consistent labeling, we can look for the best mapping from P to L, the one that preserves the most relationships.





Preliminary Definitions

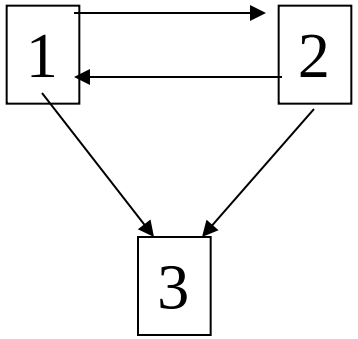
Def: A relational description D_P is a sequence of relations over a set of primitives P.

- Let $D_A = \{R_1,\ldots,R_I\}$ be a relational description over A.
- Let $D_B = \{S_1,\ldots,S_I\}$ be a relational description over B.
- Let f be a 1-1, onto mapping from A to B.
- For any relation R, the composition $R \circ f$ is given by

$$R \circ f = \{(b_1,\ldots,b_n) \mid (a_1,\ldots,a_n) \text{ is in R and } f(a_i)=(b_i),\ i=1,n\}$$

Example of Composition

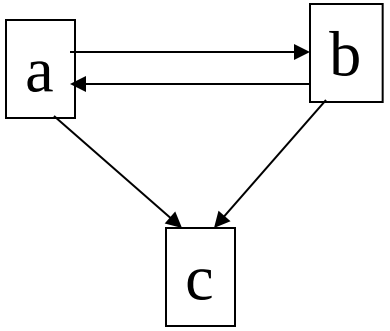
$R \circ f = \{(b_1,\ldots,b_n) \mid (a_1,\ldots,a_n) \text{ is in } R \text{ and } f(a_i)=(b_i),\ i=1,n\}$



R

1	a
2	b
3	c

f



R°f

R°f is an isomorphic copy of R with nodes renamed by f.

Relational Distance Definition

Let D_A be a relational description over set A,
D_B be a relational description over set B,
and f : A -> B.

- The **structural error of f** for R_i in D_A and S_i in D_B is

$$E_S^i(f) = | R_i \circ f - S_i | + | S_i \circ f^{-1} - R_i |$$

- The **total error of f** with respect to D_A and D_B is

$$E(f) = \sum_{i=1}^{I} E_S^i(f)$$

- The **relational distance** GD(DA,DB) is given by

$$GD(DA,DB) = \min_{f : A \rightarrow B,\ f \text{ is 1-1 and onto}} E(f)$$

Example

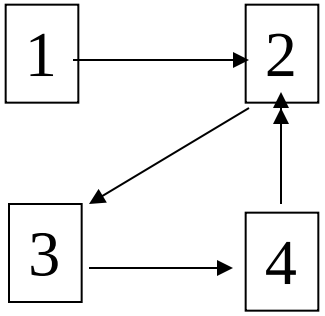
What is the best mapping?

What is the error of the best mapping?

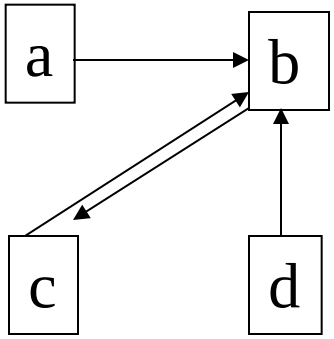
Example

Let f = {(1,a),(2,b),(3,c),(4,d)}

R



S



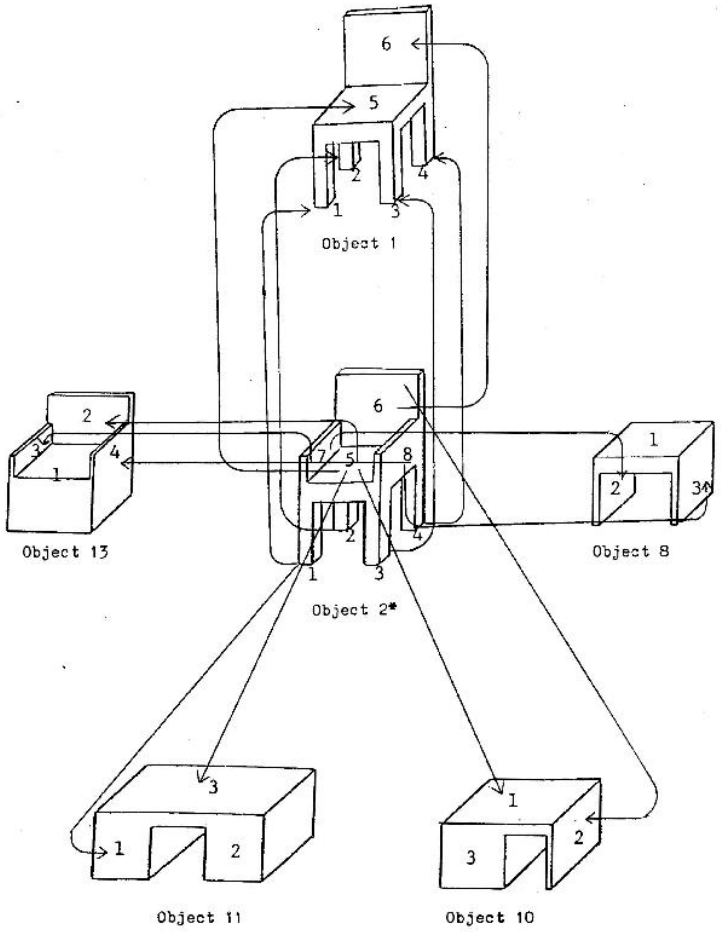
$| R \circ f - S | = |\{(a,b)(b,c)(c,d)(d,b)\} - \{(a,b)(b,c)(c,b)(d,b)\}|$
$= |\{(c,d)\}| = 1$

$|S \circ f^{-1} - R| = |\{(1,2)(2,3)(3,2)(4,2)\} - \{(1,2)(2,3)(3,4)(4,2)\}|$
$= |\{(3,2)\}| = 1$

$E(f) = 1+1 = 2$

Is there a better mapping?

Example of a Cluster of Sticks-Plates-Blobs Objects



Variations

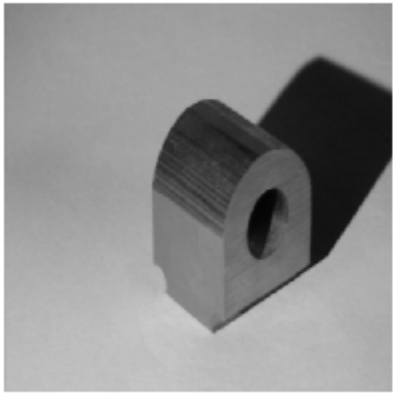
- Different weights on different relations
- Normalize error by dividing by total possible
- Attributed relational distance for attributed relations
- Penalizing for NIL mappings

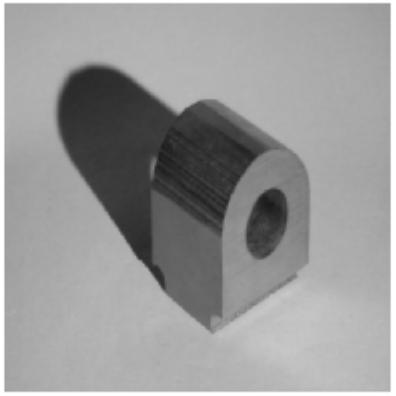
Implementation

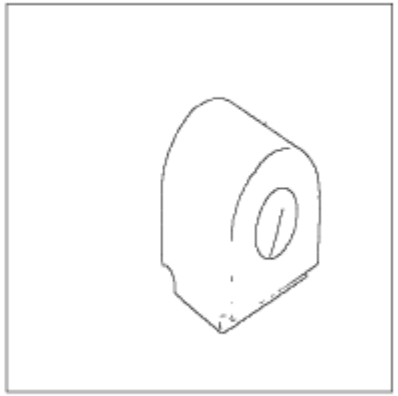
- Relational distance requires finding the lowest cost mapping from object features to image features.
- It is typically done using a branch and bound tree search.
- The search keeps track of the error after each object part is assigned an image feature.
- When the error becomes higher than the best mapping found so far, it backs up.
- It can also use discrete relaxation or forward checking (see Russel AI book on Constraint Satisfaction) to prune the search.

RIO: Relational Indexing for Object Recognition

- RIO worked with industrial parts that could have
  - planar surfaces
  - cylindrical surfaces
  - threads







Review of Alignment

Alignment is the most common paradigm for matching 3D models to either 2D or 3D data. The steps are:

1. **hypothesize a correspondence** between a set of model points and a set of data points

2. From the correspondence **compute a transformation** from model to data

3. **Apply the transformation** to the model features to produce transformed features

4. **Compare** the transformed model features to the image features to verify or disprove the hypothesis

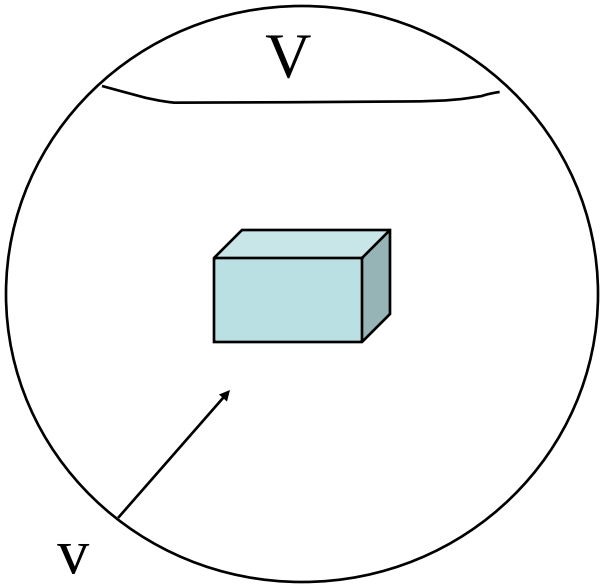
2D-3D Alignment

- single 2D images of the objects
- 3D object models
  - full 3D models, such as GC or SEV
  - view class models representing characteristic views of the objects

View Classes and Viewing Sphere

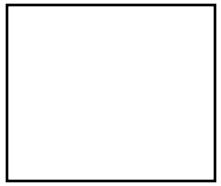
- The space of view points can be partitioned into a finite set of characteristic views.

- Each view class represents a set of view points that have something in common, such as:

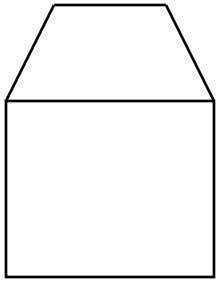


1. same surfaces are visible
2. same line segments are visible
3. relational distance between pairs of them is small

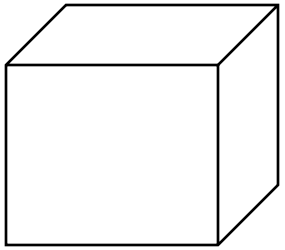
3 View Classes of a Cube



1 surface



2 surfaces

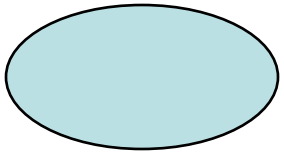


3 surfaces

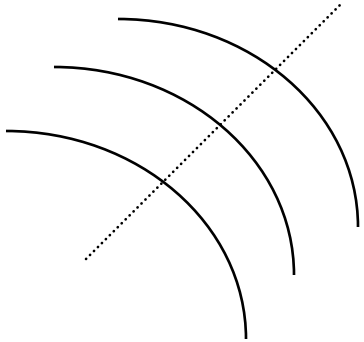
Object Representation in RIO

- 3D objects are represented by a 3D mesh and set of 2D view classes.
- Each view class is represented by an attributed graph whose nodes are features and whose attributed edges are relationships.
- For purposes of indexing, attributed graphs are stored as sets of 2-graphs, graphs with 2 nodes and 2 relationships.

RIO Features

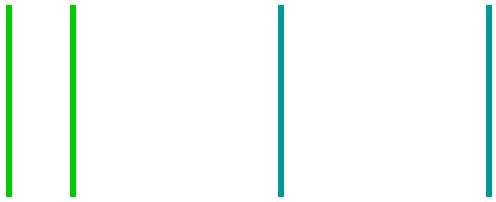


ellipses

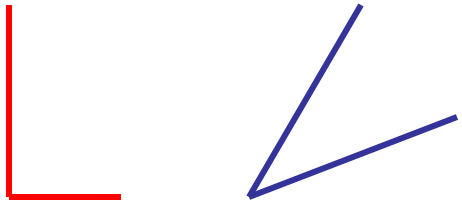


coaxials

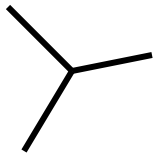
coaxials-multi



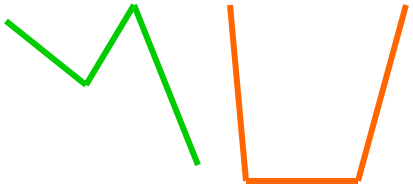
parallel lines
close and far



junctions
L V



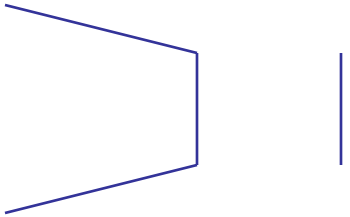
Y

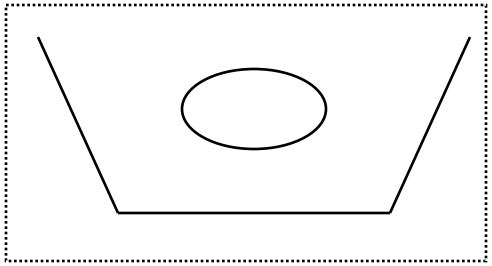


triples
Z U

RIO Relationships

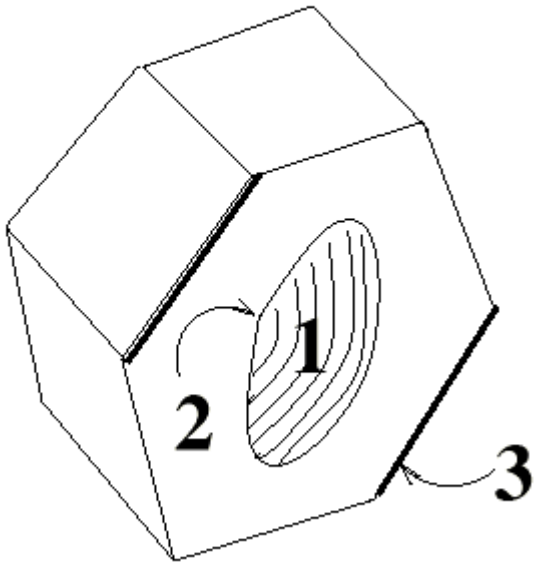
- share one arc
- share one line
- share two lines
- coaxial
- close at extremal points
- bounding box encloses / enclosed by





Hexnut Object

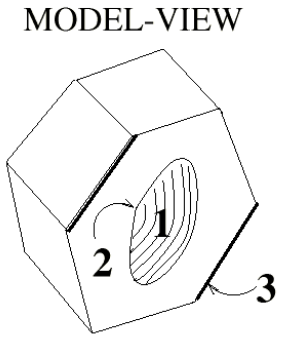
MODEL-VIEW



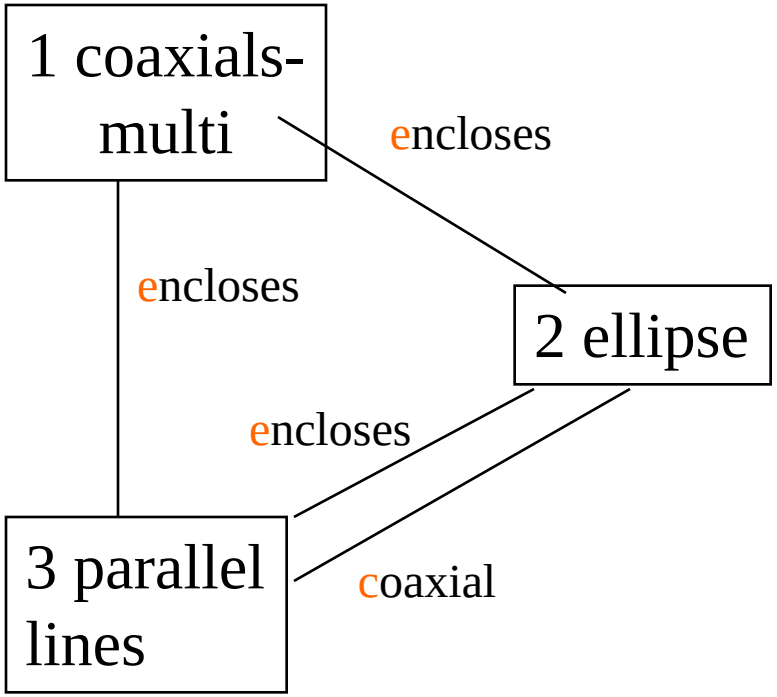
RELATIONS:
a: encloses
b: coaxial

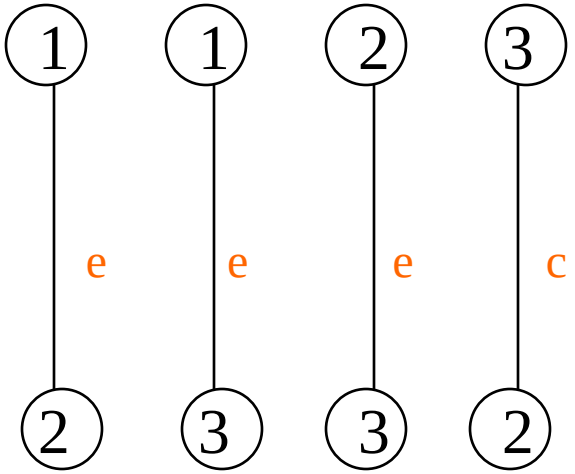
FEATURES:
1: coaxials-multi
2: ellipse
3: parallel lines

What other features and relationships can you find?



Graph and 2-Graph Representations





Relational Indexing for Recognition

Preprocessing (off-line) Phase

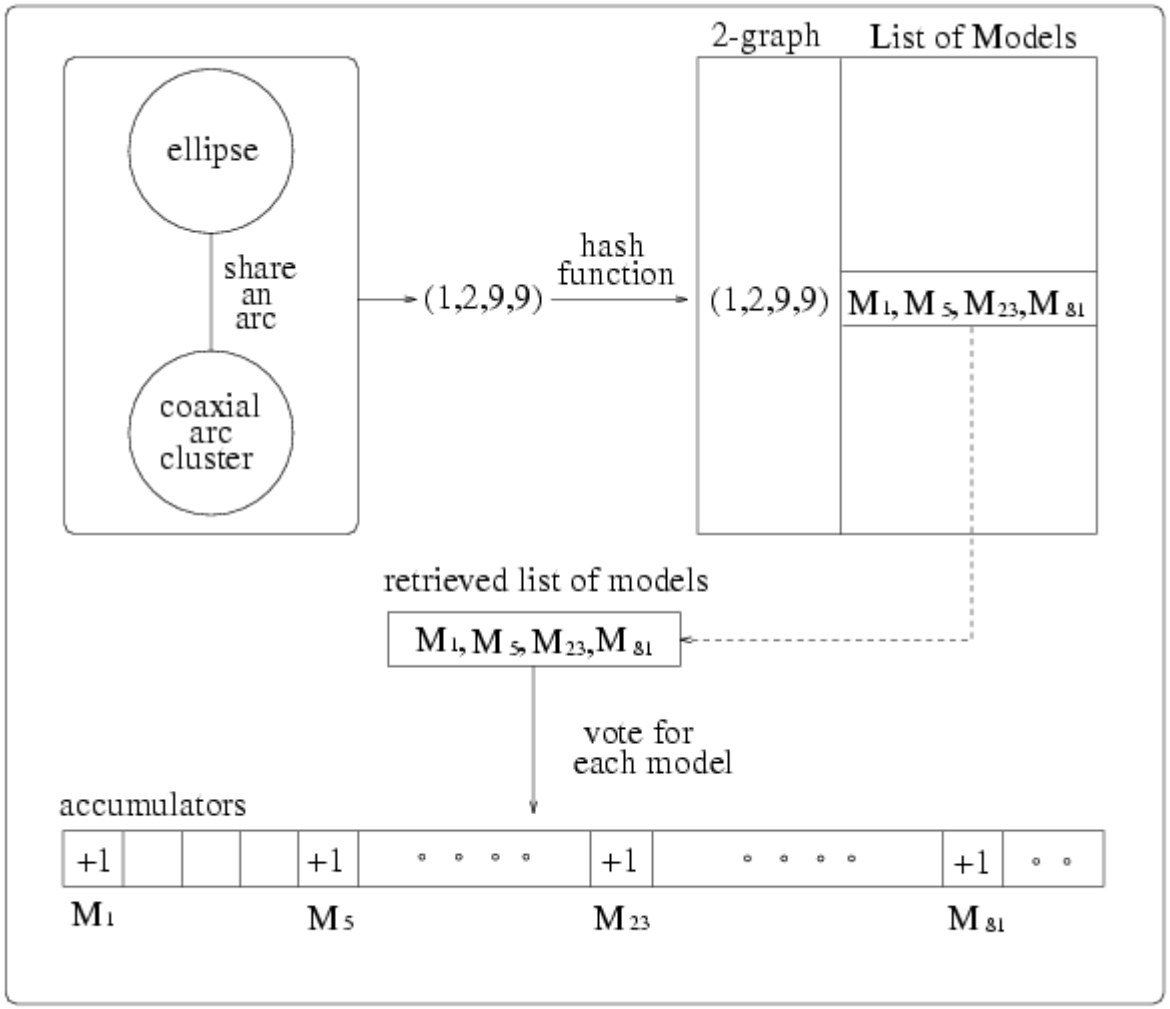
for each model view M_i in the database

- encode each 2-graph of M_i to produce an index
- store M_i and associated information in the indexed bin of a hash table H

Matching (on-line) phase

1. Construct a relational (2-graph) description D for the scene
2. For each 2-graph G of D
   - encode it, producing an index to access the hash table H
   - cast a vote for each M_i in the associated bin
3. Select the M_is with high votes as possible hypotheses
4. Verify or disprove via alignment, using the 3D meshes

The Voting Process



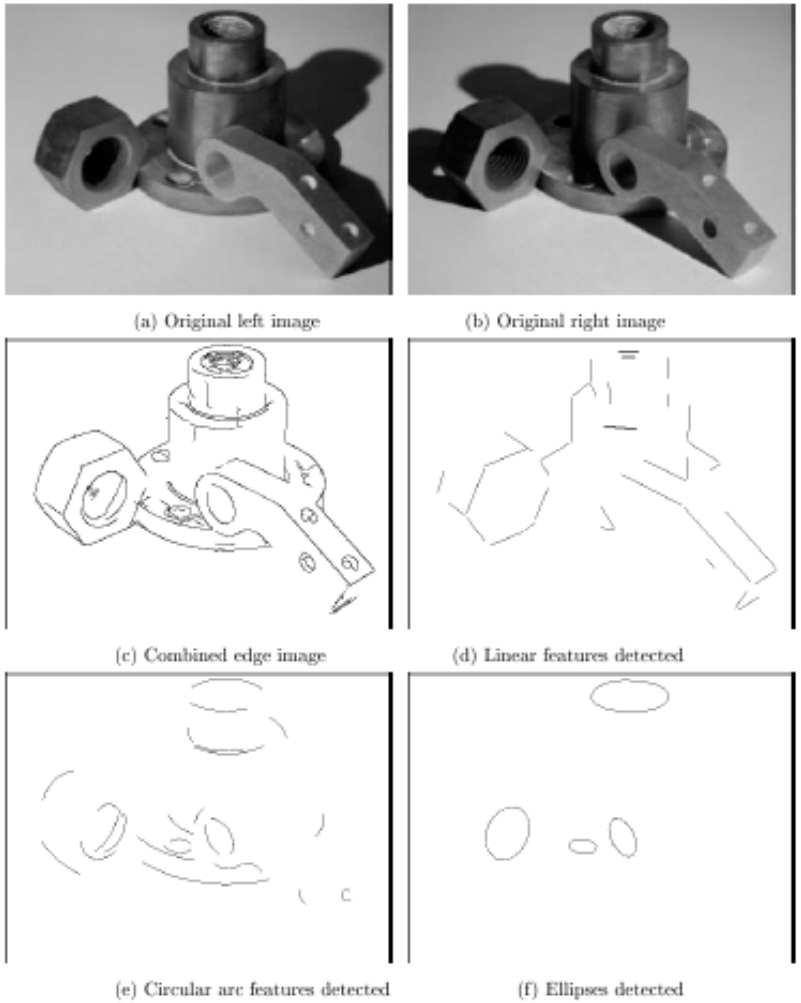
Verification

1. The matched features of the hypothesized object are used to determine its **pose**. Pose is computed from correspondences between 2D and 3D points, lines, and circles.

2. The **3D mesh** of the object is used to project all its features onto the image using perspective projection and hidden feature removal.

3. A **verification procedure** checks how well the object features line up with edges on the image, using a Hausdorf distance between expected and existing edges.

Feature Extraction



(a) Original left image

(b) Original right image

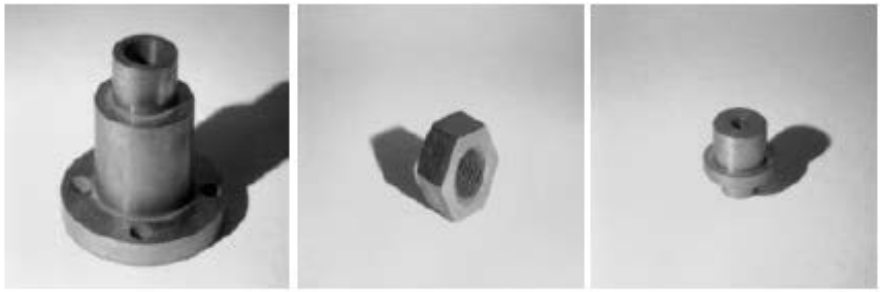
(c) Combined edge image

(d) Linear features detected

(e) Circular arc features detected

(f) Ellipses detected

Some Test Scenes

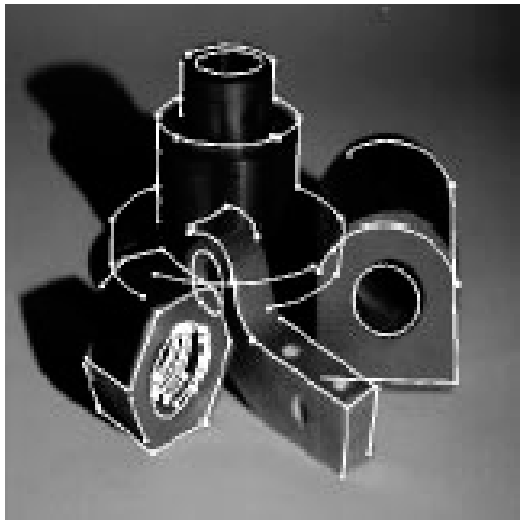


(a) Image 1 (left) (b) Image 2 (right) (c) Image 3 (left)

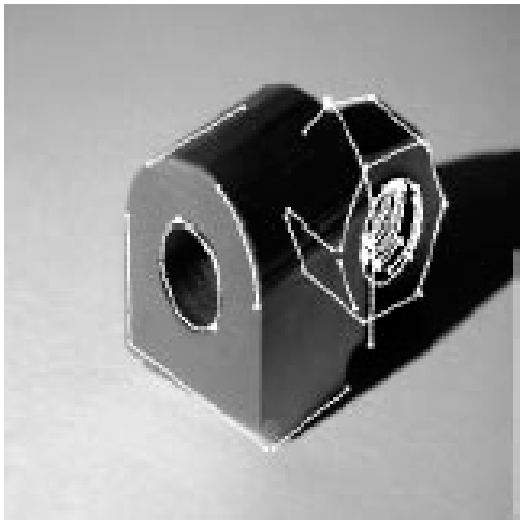
(d) Image 4 (left) (e) Image 5 (left) (f) Image 6 (right)

(g) Image 7 (left) (h) Image 8 (right) (i) Image 9 (right)

Sample Alignments
3D to 2D Perspective Projection



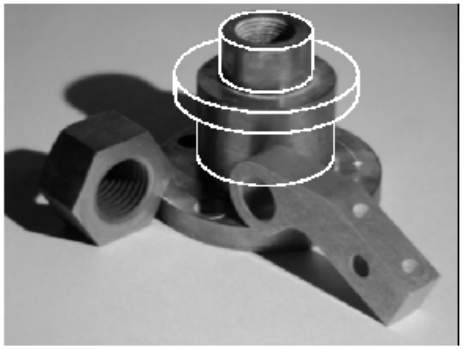
(a)

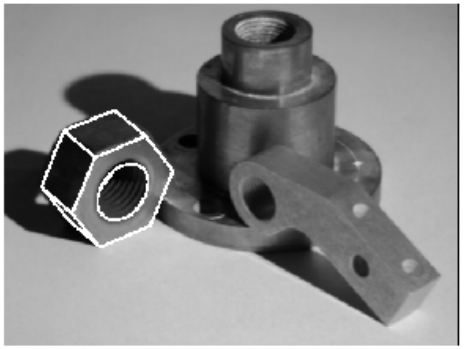


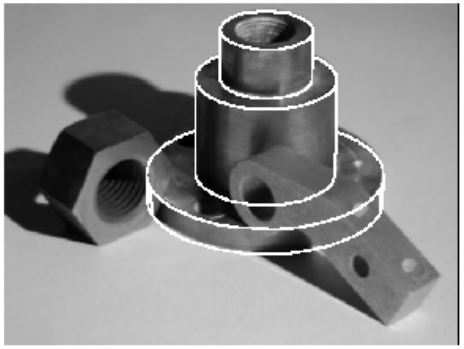
(b)

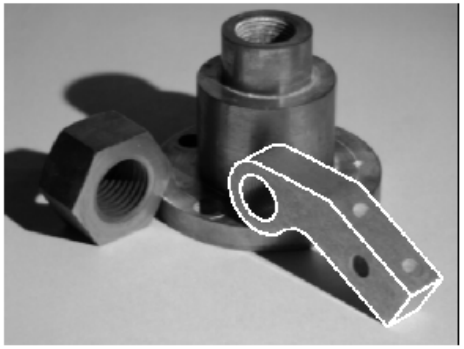
RIO Verifications

incorrect hypothesis









Summary

- 2D object recognition for specific objects (usually industrial) is done by alignment.

  -Affine transformations are usually powerful enough to handle objects that are mainly two-dimensional.

  -Matching can be performed by many different “graph-matching” methodologies.

  -Verification is a necessary part of the procedure.