

6.891

Computer Vision and Applications

Prof. Trevor. Darrell

Lecture 11: Model-based vision

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

Readings: F&P Ch 18.1-18.5

1

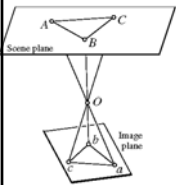
Last time

Projective SFM

- Projective spaces
- Cross ratio
- Factorization algorithm
- Euclidean upgrade

2

Projective transformations



Definition:

A *projectivity* is an invertible mapping h from P^2 to itself such that three points x_1, x_2, x_3 lie on the same line if and only if $h(x_1), h(x_2), h(x_3)$ do.

Theorem:

A mapping $h: P^2 \rightarrow P^2$ is a projectivity if and only if there exist a non-singular 3×3 matrix H such that for any point in P^2 represented by a vector x it is true that $h(x) = Hx$.

Definition: Projective transformation

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \text{or} \quad x' = Hx$$

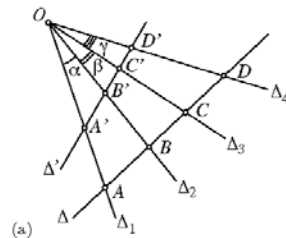
8DOF

projectivity=collineation=projective transformation=homography

[F&P, www.cs.unc.edu/~marc/mvc]

$$\{A, B, C, D\} \stackrel{\text{def}}{=} \frac{\overline{CA}}{\overline{CB}} \times \frac{\overline{DB}}{\overline{DA}}$$

The value of this cross ratio is independent of the intersecting line or plane:



4

[F&P]

Two-frame reconstruction

- Compute F from correspondences
- Compute camera matrices from F
- Compute 3D point for each pair of corresponding points

computation of F

use $x'Fx=0$ equations, linear in coeff. F
8 points (linear), 7 points (non-linear), 8+ (least-squares)
(more on this next class)

computation of camera matrices

Possible choice:

$$P = [I | 0] \quad P' = [[e'], F | e']$$

triangulation

compute intersection of two backprojected rays

5

[www.cs.unc.edu/~marc/mvc]

Perspective factorization

$$\lambda_j m_{ij} = P_i M_j, \quad i = 1, \dots, m, \quad j = 1, \dots, m$$

All equations can be collected for all i as

$$m = PM \quad \text{where,} \quad m = \begin{bmatrix} m_1 \Lambda_1 \\ m_2 \Lambda_2 \\ \dots \\ m_n \Lambda_n \end{bmatrix}, \quad P = \begin{bmatrix} P_1 \\ P_2 \\ \dots \\ P_m \end{bmatrix}$$

with:

$$m_i = [m_{i1}, m_{i2}, \dots, m_{im}], \quad M = [M_1, M_2, \dots, M_m]$$

$$\Lambda_i = \text{diag}(\lambda_{i1}, \lambda_{i2}, \dots, \lambda_{im})$$

m are known, but Λ, P and M are unknown...

Observe that PM is a product of a $3m \times 4$ matrix and a $4 \times m$ matrix, i.e. it is a rank 4 matrix

6

[www.cs.unc.edu/~marc/mvc]

Iterative perspective factorization

When Λ_i are unknown the following algorithm can be used:

1. Set $\lambda_{ij}=1$ (affine approximation).
2. Factorize \mathbf{PM} and obtain an estimate of \mathbf{P} and \mathbf{M} . If s_5 is sufficiently small then STOP.
3. Use \mathbf{m} , \mathbf{P} and \mathbf{M} to estimate Λ_i from the camera equations (linearly) $\mathbf{m}_i = \Lambda_i \mathbf{P}_i \mathbf{M}$
4. Goto 2.

In general the algorithm minimizes the *proximity measure* $P(\Lambda, \mathbf{P}, \mathbf{M}) = s_5$

Structure and motion recovered up to an arbitrary projective transformation

7

[www.cs.unc.edu/~marr/mv1]



Euclidean upgrade

Given a camera with known intrinsic parameters, we can take the calibration matrix to be the identity and write the perspective projection equation in some Euclidean world coordinate system as

$$\mathbf{p} = \frac{1}{z} (\mathcal{R} \quad \mathbf{t}) \begin{pmatrix} \mathbf{P} \\ 1 \end{pmatrix} = \frac{1}{\lambda z} (\mathcal{R} \quad \lambda \mathbf{t}) \begin{pmatrix} \lambda \mathbf{P} \\ 1 \end{pmatrix}$$

for any non-zero scale factor λ . If \mathcal{M}_i and \mathbf{P}_j denote the shape and motion parameters measured in some Euclidean coordinate system, there must exist a 4×4 matrix \mathcal{Q} such that $\hat{\mathcal{M}}_i = \mathcal{M}_i \mathcal{Q}$ and $\hat{\mathbf{P}}_j = \mathcal{Q}^{-1} \mathbf{P}_j$.

8

[F&P]

Today: “Model-based Vision”

Still feature and geometry-based, but now with moving objects rather than cameras...

Topics:

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

9

Approach

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

10

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)?

11

Features?

- Points
- but also,
- Lines
 - Conics
 - Other fitted curves
 - Regions (particularly the center of a region, etc.)

12

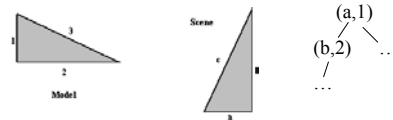
How to generate hypotheses?

- 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
 - $O(LM^N)$!
- Add geometric constraints to prune search, leading to *interpretation tree search*
- Try subsets of features (frame groups)...

13

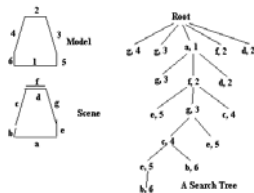
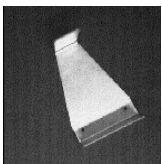
Interpretation Trees

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



14

Interpretation Trees



“Wild cards” handle spurious image features

[A.M. Wallace. 1988,₁]

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.)

16

Pose consistency / Alignment

- Given known camera type in some unknown configuration (pose)
 - Hypothesize configuration from set of initial features
 - Backproject
 - Test
- “Frame group” -- set of sufficient correspondences to estimate configuration, e.g.,
 - 3 points
 - intersection of 2 or 3 line segments, and 1 point

17

Alignment

```

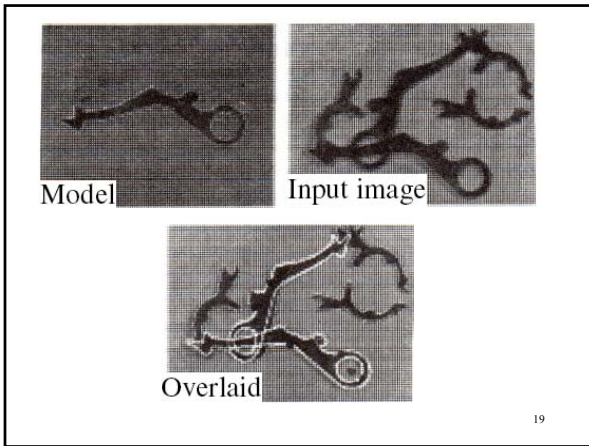
For all object frame groups O
  For all image frame groups F
    For all correspondences C between
      elements of F and elements
        of O

      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
    
```

18



Pose clustering

- Each model leads to many correct sets of correspondences, each of which has the same pose
- Vote on pose, in an accumulator array (per object)

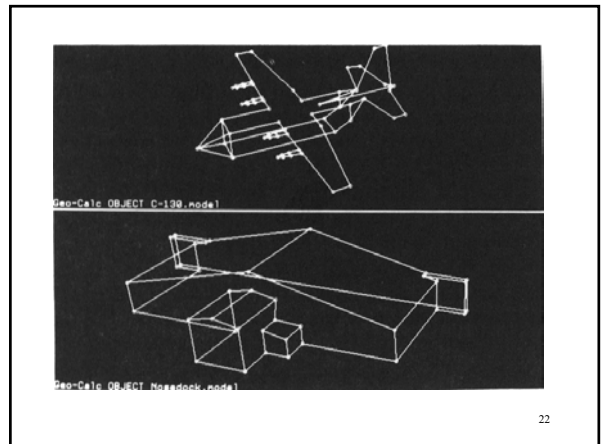
20

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
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
  
```

21



Pose clustering

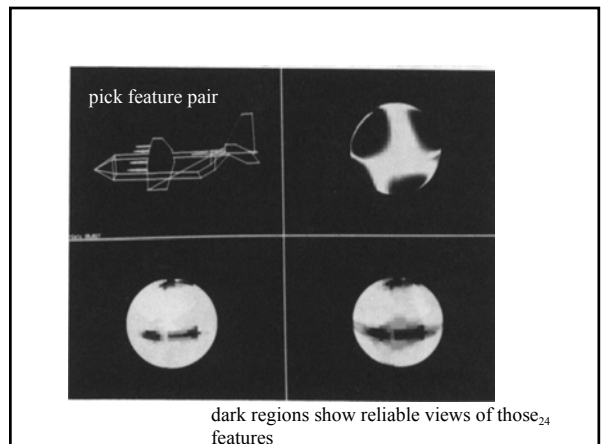
Problems

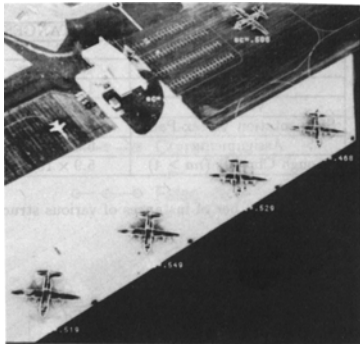
- Clutter may lead to more votes than the target!
- Difficult to pick the right bin size

Confidence-weighted clustering

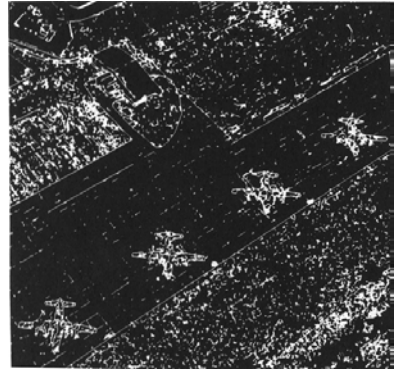
- See where model frame group is reliable (visible!)
- Downweight / discount votes from frame groups at poses where that frame group is unreliable...

23

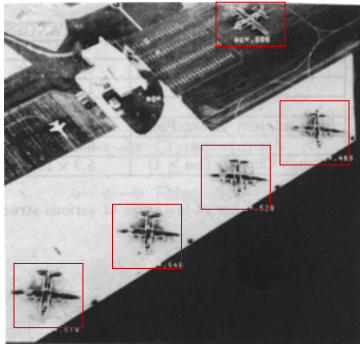




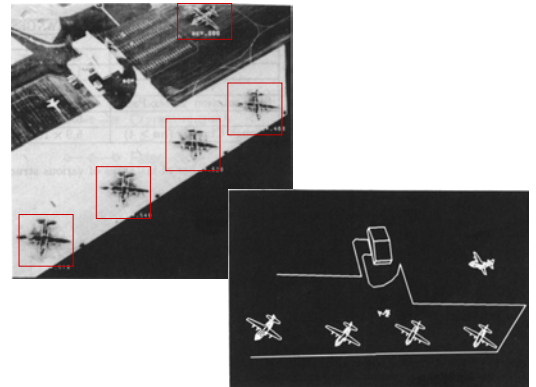
25



26



27



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

29

[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)

30

[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:

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

31 [Lowe]

Models for planar surfaces with SIFT keys:



32 [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

34 [Lowe]

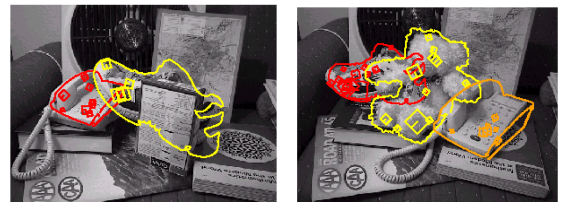
3D Object Recognition



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

35 [Lowe]

Recognition under occlusion



36 [Lowe]

Location recognition



37
[Lowe]

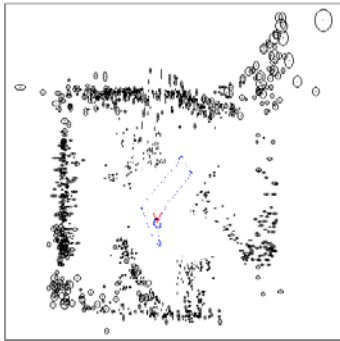
Robot Localization

- Joint work with Stephen Se, Jim Little



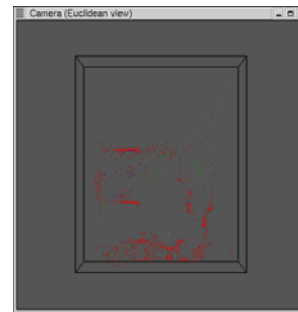
38
[Lowe]

Map continuously built over time



39
[Lowe]

Locations of map features in 3D



40
[Lowe]

Invariant recognition

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

41

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

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

42

Invariance

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

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

43

Invariance

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

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

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

44

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 45

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

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 46

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

Geometric Hashing

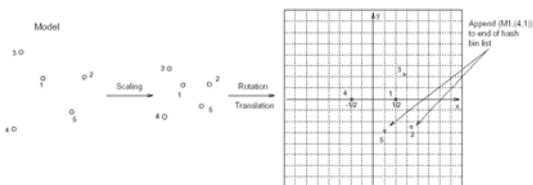


Figure 1. Determining the hash table entries when points 4 and 1 are used to define a basis. The models are allowed to undergo rotation, translation, and scaling. On the left of the figure, model M comprises five points.

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

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

Basis Geometric Hashing

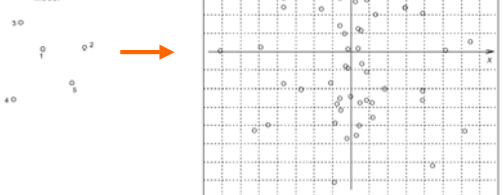


Figure 2. The locations of the hash table entries for model M . Each entry is labeled with the information “model M ” and the basis pair (i, j) used to generate the entry. The models are allowed to undergo rotation, translation, and scaling.

Wolfson and Rigoutsos, *Geometric Hashing, an Overview*, 1997 48

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

Geometric Hashing

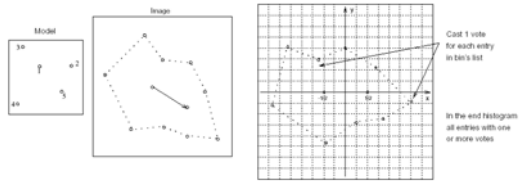


Figure 3. Determining the hash table bins that are to be notified when two arbitrary image points are selected as a basis. Similarity transformation is allowed.

Wolfson and Rigoutsos, *Geometric Hashing, an Overview*, 1997⁴⁹
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  $\mu$ '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
    
```

50

Indexing with invariants

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

51

Projective invariants

- Projective invariant for coplanar points
- Perspective projection of coplanar points is a plane perspective transform:
 $p = MP \rightarrow p = AP$, with 3×3 A
- determinant ratio of 5 point tuples is invariant

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

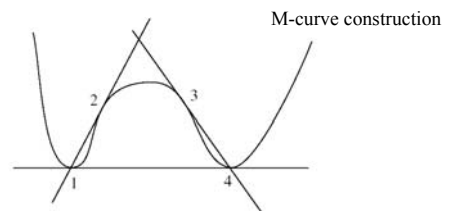
52

$$\begin{aligned} \frac{\det([p_i, p_j, p_k]) \det([p_i, p_l, p_m])}{\det([p_i, p_j, p_l]) \det([p_i, p_k, p_m])} &= \frac{\det([AP_i, AP_j, AP_k]) \det([AP_i, AP_l, AP_m])}{\det([AP_i, AP_j, AP_l]) \det([AP_i, AP_k, AP_m])} \\ &= \frac{\det(A [P_i, P_j, P_k]) \det(A [P_i, P_l, P_m])}{\det(A [P_i, P_j, P_l]) \det(A [P_i, P_k, P_m])} \\ &= \frac{(\det(A)^2) \det([P_i, P_j, P_k]) \det([P_i, P_l, P_m])}{(\det(A)^2) \det([P_i, P_j, P_l]) \det([P_i, P_k, P_m])} \\ &= \frac{\det([P_i, P_j, P_k]) \det([P_i, P_l, P_m])}{\det([P_i, P_j, P_l]) \det([P_i, P_k, P_m])} \end{aligned}$$

53

Tangent invariance

- Incidence is preserved despite transformation



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

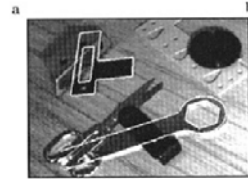
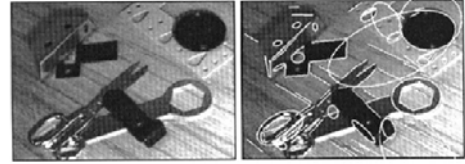
54

```

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

```

55

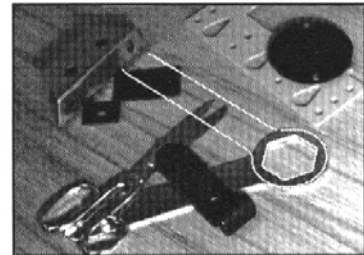


56

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?

57



58

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

59

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

60

[Lowe]

Today: “Model-based Vision”

- Hypothesize and test
- Interpretation Trees
- Alignment
- Pose Clustering
- Invariances
- Geometric Hashing
- *Tuesday: Project previews!*