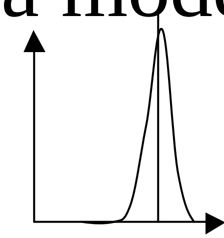
Object Pose from a Single Image

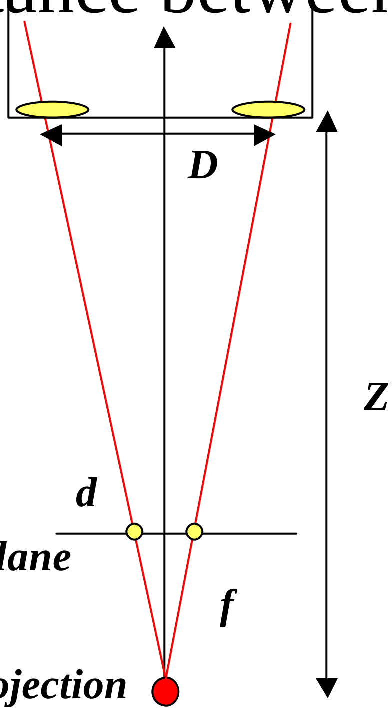
How Do We See Objects in Depth?

- Stereo
  - Use differences between images in our left and right eye
  - How much is this difference for a car at 100 m?
- Move our head sideways
  - Or, the scene is moving
  - Or we are moving in a car
- We know the size and shape of objects
  - Traffic lights, car headlights and taillights

Headlights in the Dark

- A robot could evaluate its distance from incoming cars at night partly from a model of cars
  - Distance between headlights known

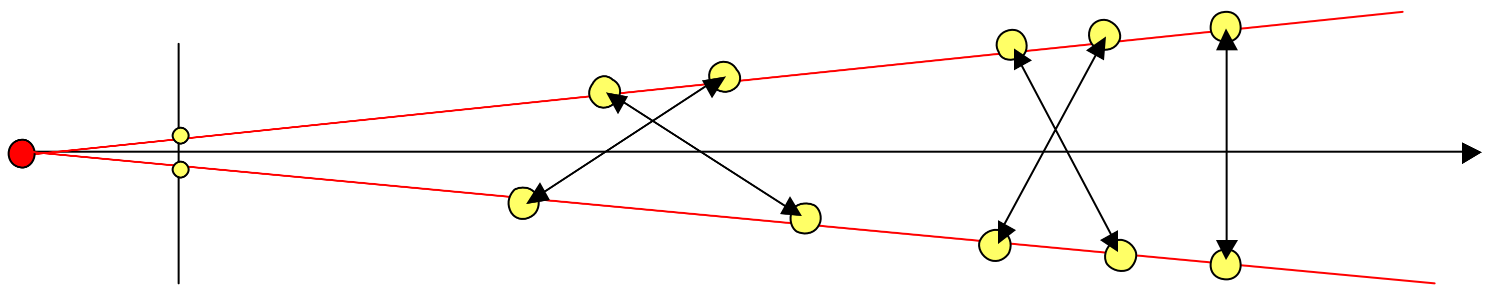




$$Z = \frac{f\,D}{d}$$

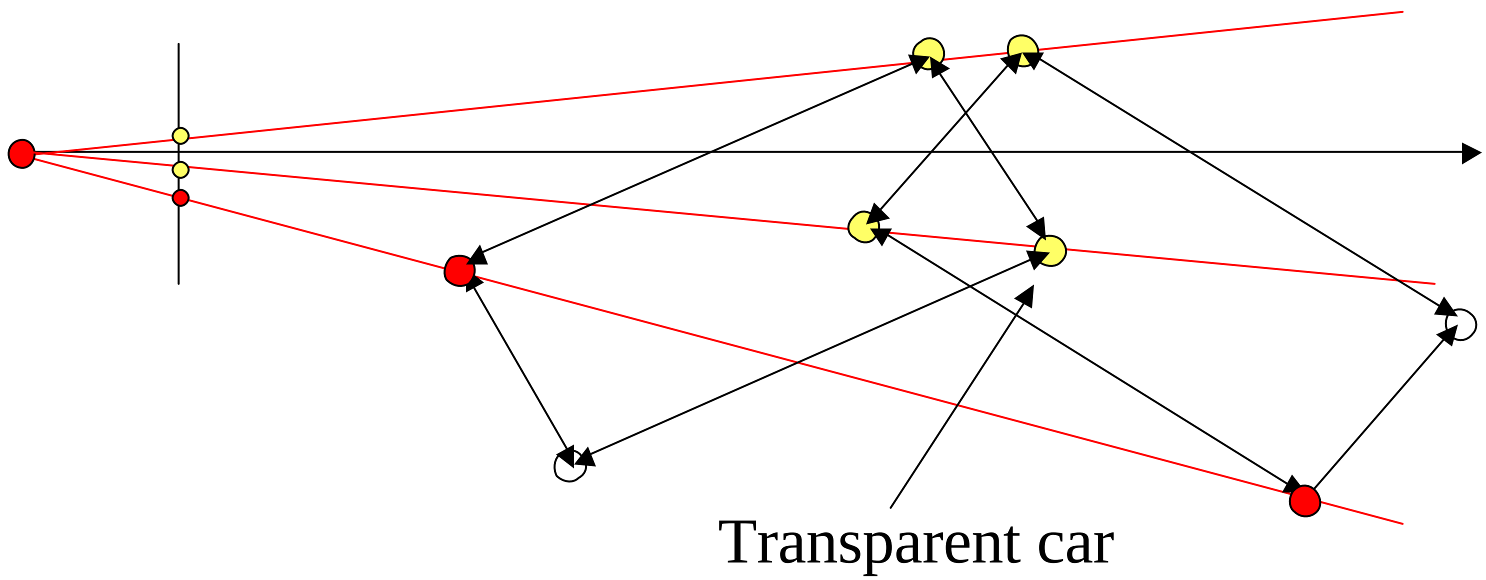
Object Pose with 1D Image Plane

- What happens if we don't know object's angle?



More Points

- Limited number of object poses (2 or 1)
  - Head lights and one taillight



Correspondence Problem

- When we know correspondences (i.e. matchings), pose is easier to find
- When we know the pose, correspondences are easier to find.
- But we need to find both at the same time
- Below, we first assume we *know* correspondences and describe how to solve the pose given *n* corresponding points in image and object
  - Perspective *n*-Point Problem
- Then we explore what to do when we don't know correspondences

Pose vs. Calibration Problem

- Now we know calibration matrix **K** → $K = \begin{bmatrix} a_x & s & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$
- We can transform image points by $\mathbf{K}^{-1}$ transformation

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} u' \\ v' \\ w' \end{bmatrix}$$

Canonical perspective projection with $f = 1$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{K}^{-1}\mathbf{K}\left[\mathbf{I}_3 \quad | \quad \mathbf{0}_3\right] \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}_3^{\mathbf{T}} & 1 \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \left[\mathbf{R} \quad \mathbf{T}\right] \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix}$$

- Projection matrix is now $\mathbf{P} = [\mathbf{R} \quad \mathbf{T}]$
- Solving pose problem consists of finding **R** and **T**
- **6 unknowns**

Iterative Pose Calculation

- First we derive a linear system for the unknown parameters of rotation and translation that contains the known world coordinates of points and the homogenous coordinates of their images.
  - Problem: Does not contain the wi components
  - The wi components are required for computing homogeneous coordinates of images from the pixel locations
  - They can be computed once the rotation and translation parameters are estimated
  - Solution: Make a guess on wi, compute R and T, then recompute wi, and recompute R and T, etc

Iterative Pose Calculation

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{r_1^T} & T_x \\ \mathbf{r_2^T} & T_y \\ \mathbf{r_3^T} & T_z \end{bmatrix} \begin{bmatrix} X_S \\ Y_S \\ Z_S \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{r_1^T}/T_z & T_x/T_z \\ \mathbf{r_2^T}/T_z & T_y/T_z \\ \mathbf{r_3^T}/T_z & 1 \end{bmatrix} \mathbf{X}$$

$$\Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mathbf{r_1^T}/T_z & T_x/T_z \\ \mathbf{r_2^T}/T_z & T_y/T_z \end{bmatrix} \mathbf{X} \quad \Rightarrow [u \quad v] = \mathbf{X}^{\mathbf{T}} \begin{bmatrix} \mathbf{r_1}/T_z & \mathbf{r_2^T}/T_z \\ T_x/T_z & T_y/T_z \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \\ u_4 & v_4 \end{bmatrix} = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \\ X_2 & Y_2 & Z_2 & 1 \\ X_3 & Y_3 & Z_3 & 1 \\ X_4 & Y_4 & Z_4 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r_1}/T_z & \mathbf{r_2}/T_z \\ T_x/T_z & T_y/T_z \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \mathbf{r_1}/T_z & \mathbf{r_2}/T_z \\ T_x/T_z & T_y/T_z \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \\ u_4 & v_4 \end{bmatrix}$$

Non coplanar points needed (otherwise matrix **M** is singular). At least 4 points.

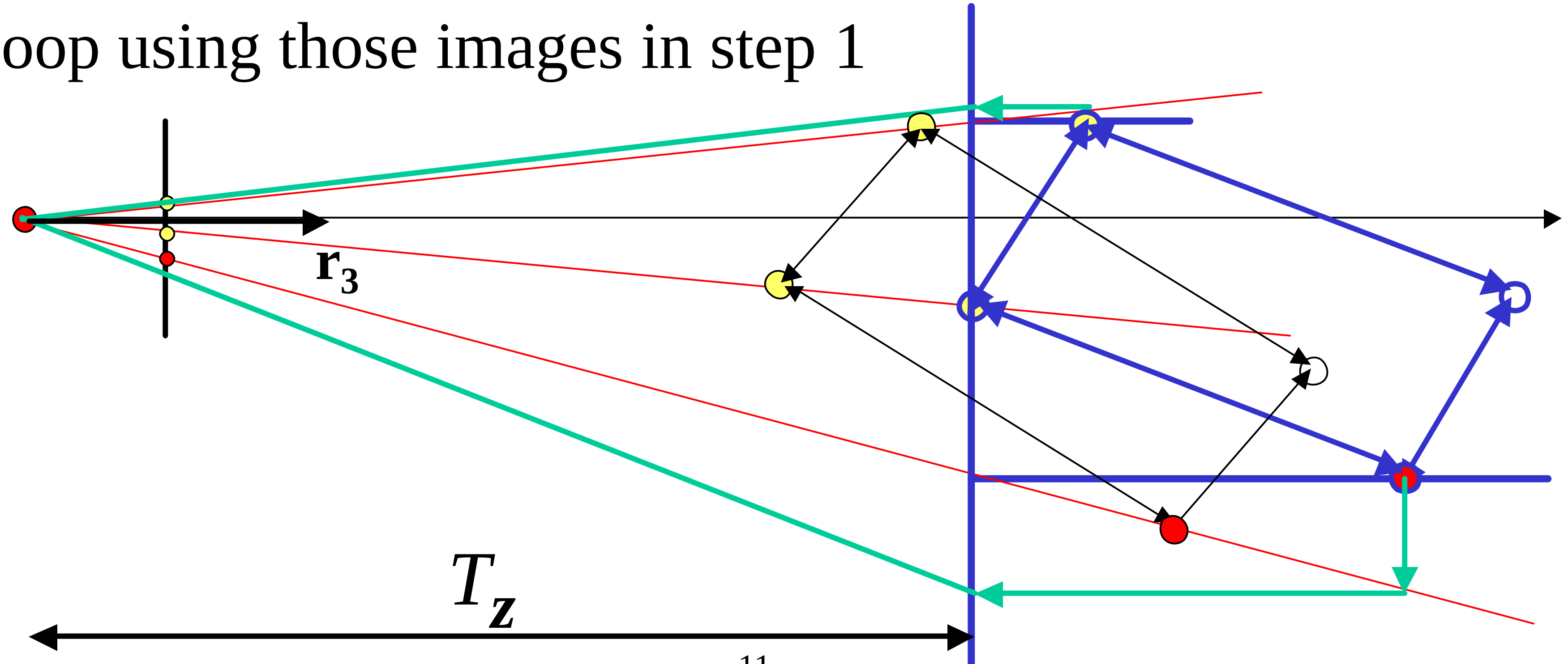
$$w_i = 1 + \mathbf{r_3} . (X_i, Y_i, Z_i)/T_z$$

Iterative Pose Calculation

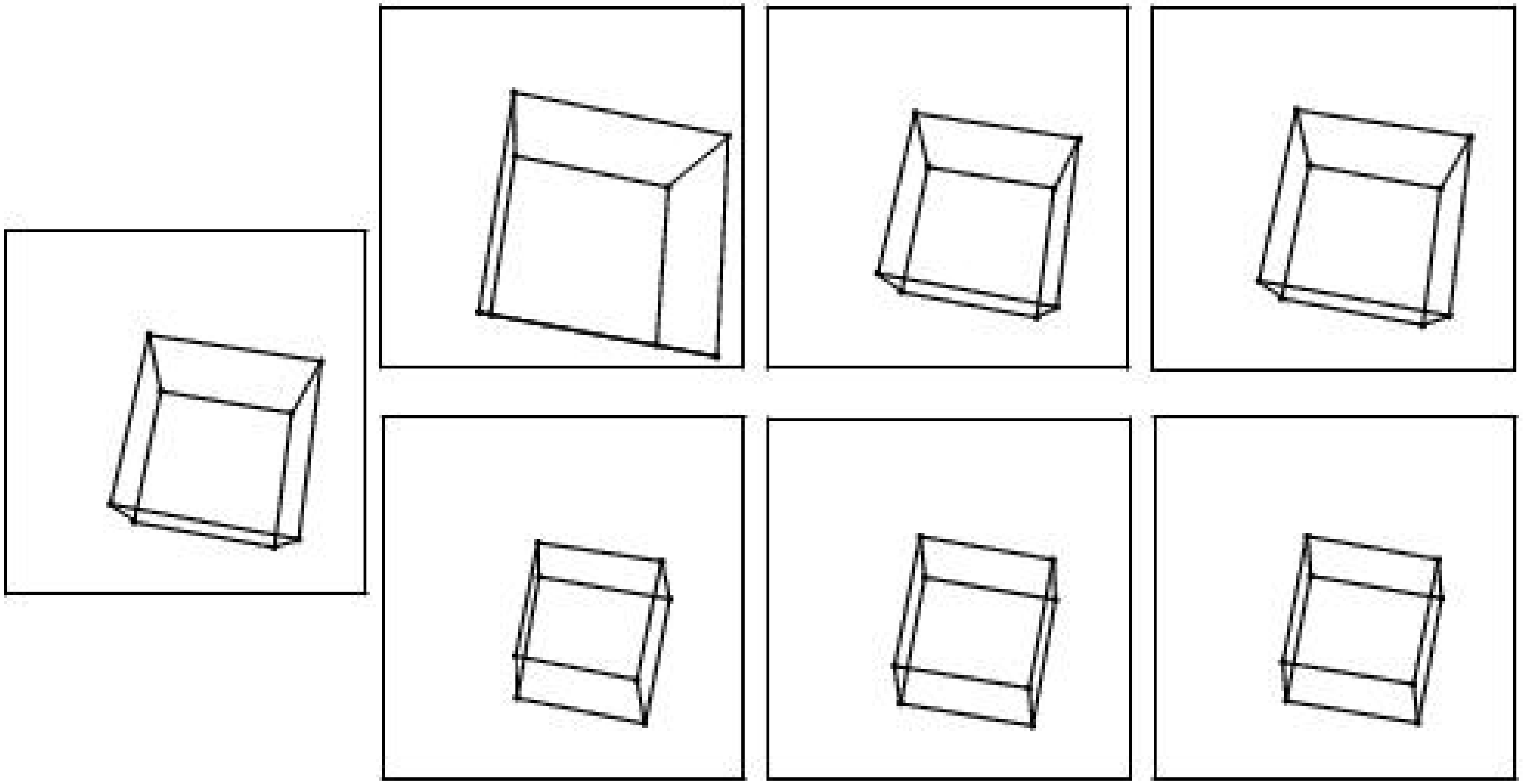
- Compute model matrix **M** and its inverse
- Assume $\mathbf{r_3}\,\mathbf{.}(X_i, Y_i, Z_i)/T_z = 1 \Rightarrow w_i = 1$
- Compute $u_i = w_i\, x_i$, $v_i = w_i\, y_i$
- Compute $$\begin{bmatrix} \mathbf{r_1}/T_z & \mathbf{r_2}/T_z \\ T_x/T_z & T_y/T_z \end{bmatrix} = \mathbf{M}^{\mathbf{-1}} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \\ u_3 & v_3 \\ u_4 & v_4 \end{bmatrix}$$
- Compute T_z, T_x, T_y, $\mathbf{r_1}$, $\mathbf{r_2}$, then $\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}$
- Compute $w_i = 1 + \mathbf{r_3}\,\mathbf{.}(X_i, Y_i, Z_i)/T_z$
- Go back to step 2 and iterate until convergence

Iterative Pose Calculation

1. Find object pose under scaled orthographic projection
2. Project object points on lines of sight
3. Find scaled orthographic projection images of those points
4. Loop using those images in step 1



POSIT for a Cube



Left: Actual perspective image for cube with known model

Top: Evolution of perspective image during iteration

Bottom: Evolution of scaled orthographic projection

Application: 3D Mouse

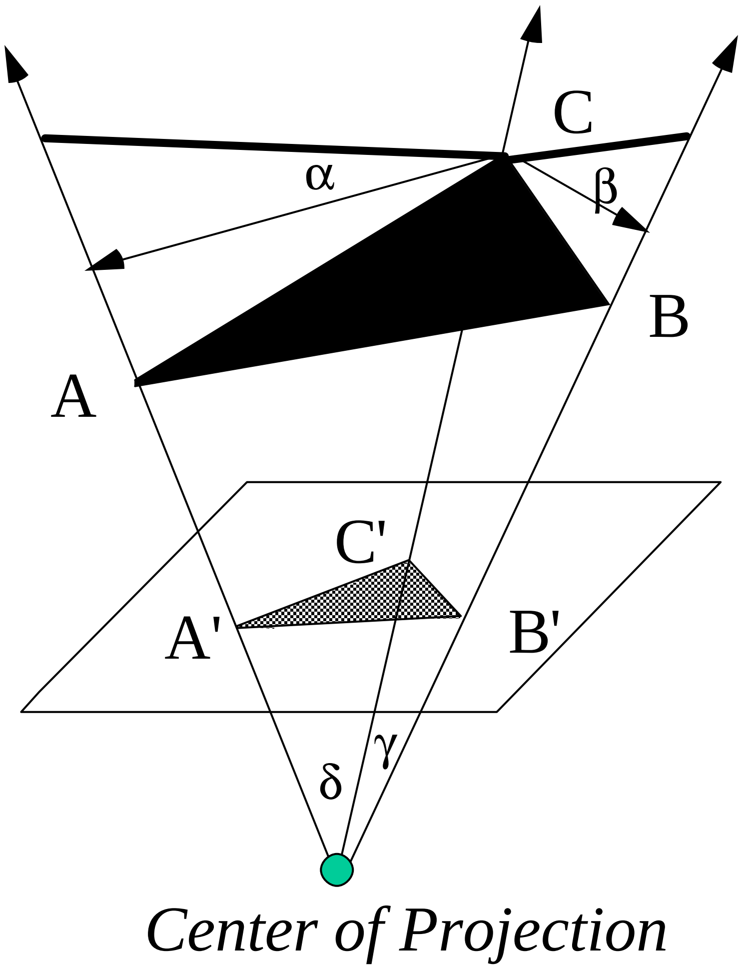
3 Points

- Each correspondence between scene point and image point determines 2 equations
- Since there are 6 degrees of freedom in the pose problems, the correspondences between 3 scene points in a known configuration and 3 image points should provide enough equations for computing the pose of the 3 scene points
- the pose of a triangle of known dimension is defined from a single image of the triangle
  - But nonlinear method, 2 to 4 solutions

Triangle Pose Problem

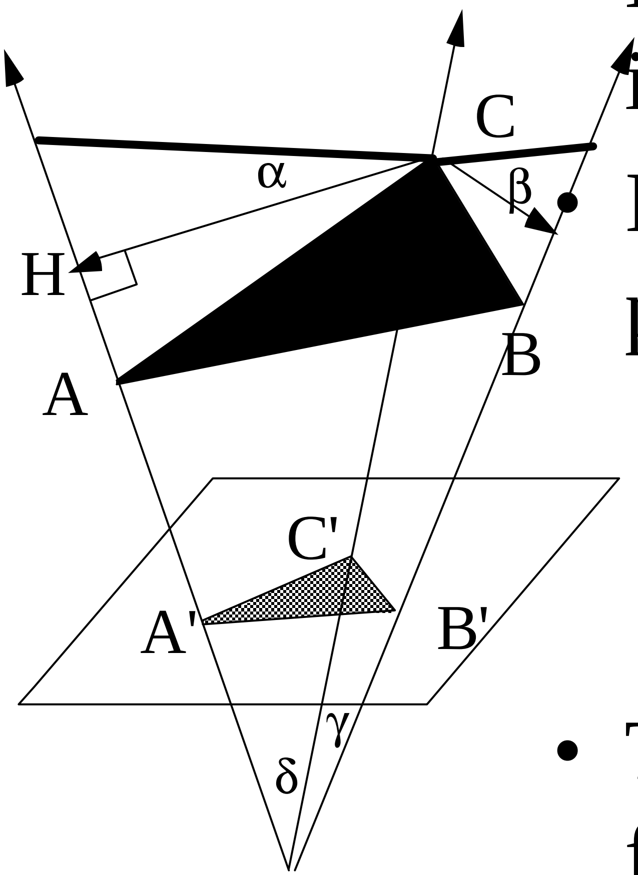
- There are two basic approaches
  - Analytically solving for unknown pose parameters
    - Solving a 4th degree equation in one pose parameter, and then using the 4 solutions to the equation to solve for remaining pose parameters
    - problem: errors in estimating location of image features can lead to either large pose errors or failure to solve the 4th degree equation
  - Approximate numerical algorithms
    - find solutions when exact methods fail due to image measurement error
    - more computation

Numerical Method for Triangle Pose



- If distance R_c to C is known, then possible locations of A (and B) can be computed
  - they lie on the intersections of the line of sight through A' and the sphere of radius AC centered at C
  - Once A and B are located, their distance can be computed and compared against the actual distance AB

Numerical Method for Triangle Pose

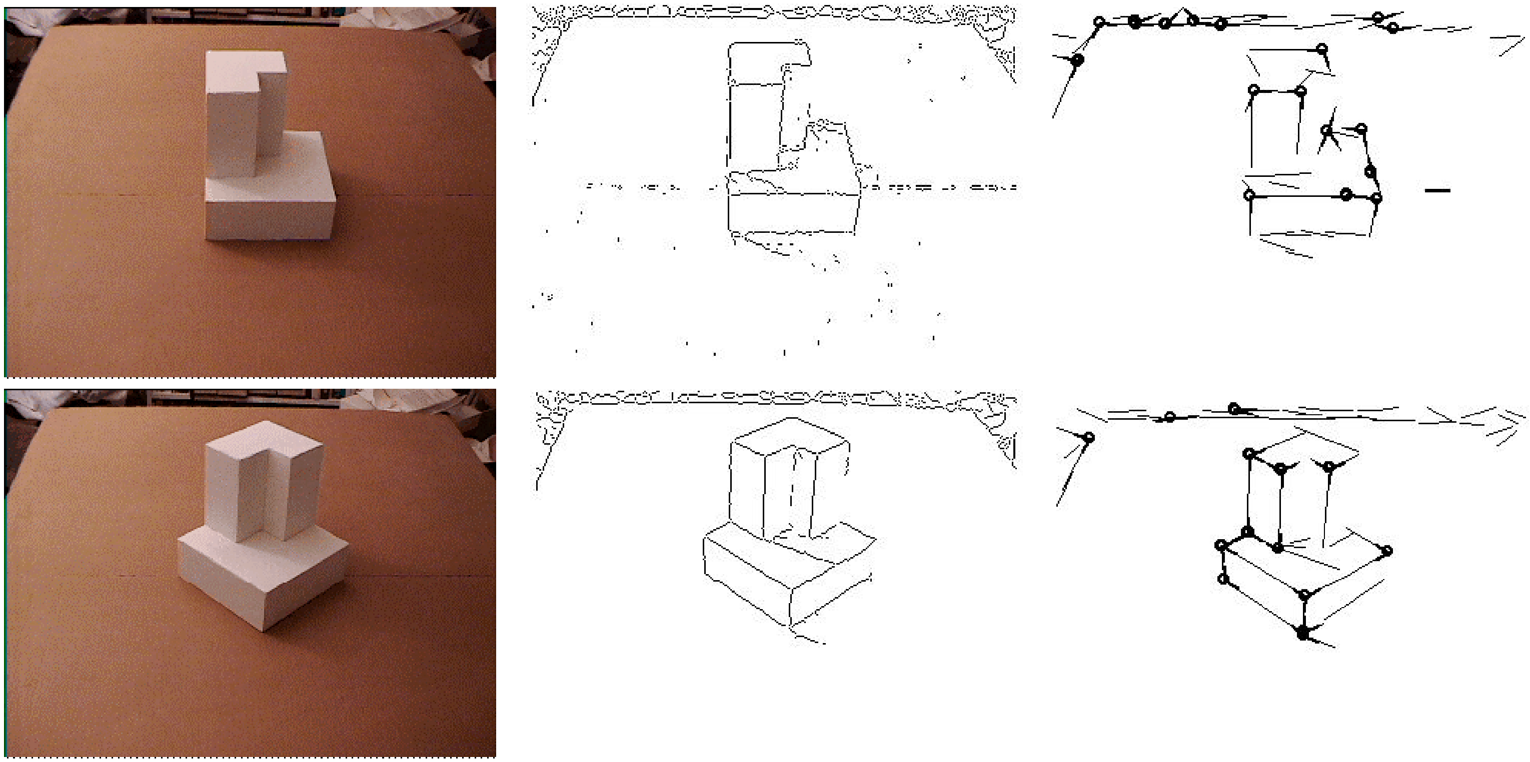


- Not practical to search on R_c since it is unbounded
- Instead, search on one angular pose parameter, α.
  - $R_c = AC \cos\alpha / \sin\delta$
  - $R_a = R_c \cos\delta \pm AC \sin\alpha$
  - $R_b = R_c \cos\gamma \pm [(BC^2 - (RC \sin\gamma)^2]^{1/2}$
- This results in four possible lengths for side AB
- Keep poses with the right AB length

Choosing Points on Objects

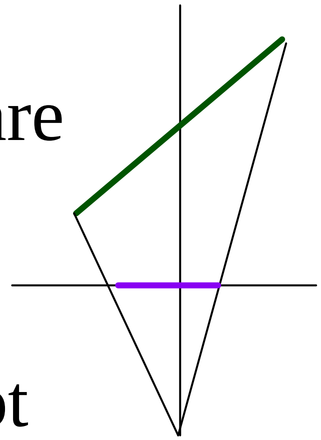
- Given a 3-D object, how do we decide which points from its surface to choose for its model?
  - Choose points that will give rise to detectable features in images
  - For polyhedra, the images of its vertices will be points in the images where two or more long lines meet
    - These can be detected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

Example images



Choosing the Points

- 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



Objects and Unknown Correspondences

- Strategy:
  - Pick up a small group of points (3 or 4) on object, and candidate image points in image
  - Find object pose for these correspondences
  - Check or accumulate evidence by one of following techniques:
    - Clustering in pose space
    - Image-Model Alignment and RANSAC

4-3-2-?

- 4 - point perspective solution
  - Unique solution for 6 pose parameters
  - Computational complexity of n^4m^4
- 3 - point perspective solution
  - Generally two solutions per triangle pair, but sometimes four.
  - Reduced complexity of n^3m^3

Reducing the Combinatorics of Pose Estimation

- How can we reduce the number of matches
  - Consider only quadruples of object features that are simultaneously visible
    - extensive preprocessing

Reducing the Combinatorics of Pose Estimation

- Reducing the number of matches
  - Consider only quadruples of image features that
    - Are connected by edges
    - Are "close" to one another
      - But not too close or the inevitable errors in estimating the position of an image vertex will lead to large errors in pose estimation
  - Generally, try to group the image features into sets that are probably from a single object, and then only construct quadruples from within a single group

Image-Model Alignment

- Given:
  - A 3-D object modeled as a collection of points
  - Image of a scene suspected to include an instance of the object, segmented into feature points
- Goal
  - **Hypothesize** the pose of the object in the scene by matching (collections of) *n* model points against *n* feature points, enabling us to solve for the rigid body transformation from the object to world coordinate systems, and
  - **Verify** that hypothesis by projecting the remainder of the model into the image and matching
    - Look for edges connecting predicted vertex locations
    - Surface markings

RANSAC

- **RAN**dom **SA**mple **C**onsensus
- Randomly select a set of 3 points in the image and a select a set of 3 points in the model
- Compute triangle pose and pose of model
- Project model at computed pose onto image
- Determine the set of projected model points that are within a distance threshold t of image points, called the *consensus set*
- After N trials, select pose with largest consensus set

Clustering in Pose Space

- Each matching of *n* model points against *n* feature points provides **R** and **T**
- **Each correct matching provides a similar rotation and translation**
- Represent each pose by a point in a 6D space. Then points from correct matchings should cluster
- Or find clusters for points **T** and find the cluster where the rotations are most consistent
  - "Generalized Hough transform" if bins are used

Scope of the Problem

- Flat objects vs. 3D objects
  - Grabbing flat tools on a tray vs. grabbing handle of cup
- Rounded objects vs. polyhedral objects
  - Cup vs. keyboard or CD cassette
- Rigid objects vs. deformable objects

Pose and Recognition

- Solving the Pose Problem can be used to solve the Recognition Problem for 3D objects:
  - Try to find the pose of each item in the database of objects we want to identify
  - Select the items whose projected points match the largest amounts of image points in the verification stage, and label the corresponding image regions with the item names.
  - But many alternative recognition techniques do not provide the pose of the recognized item.

References

- VAST literature on the subject (larger than for calibration)
- Search for pose & object & model & USC
- Search in citeseer.nj.nec.com