Model-based 3D tracking

Prof. Didier Stricker
Doz. Gabriele Bleser
Two different levels

Recognition (2D)
Tracking (2D)

Pose estimation, Tracking in 3D

Where in the image … ?

Where in the world … ?
Initial Pose Estimation

Recognition/Tracking
(x, y)

Pose estimation
(X, Y, Z, \( \phi \), \( \psi \), \( \gamma \))
Problem statement

To compute the position and orientation of the object by tracking it through the video sequence

- The description of the object consists of a 3D CAD model
- Objects can be complex and have little structures (texture)
  → use contours / edges
Tracking with a 3D contour model
Tracking with a 3D contour model

- Tracking using the silhouette
- The contour has to be recomputed online
- The CAD and model does not coincide completely – this will create outliers.

(Howaldtwerke-Deutsche Werft)
Outline

• Part I: Camera model and pose computation

• Part II: 3D countour tracking (RAPID)
Image formation

Let’s design a camera

• Idea 1: put a piece of film in front of an object
• Do we get a reasonable image?
Add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the **aperture**
- How does this transform the image?
Camera Obscura

The first camera
• Known to Aristotle
Camera coordinate system

- **Principal axis**: line from the camera center perpendicular to the image plane
- **Normalized (camera) coordinate system**: camera center is at the origin and the principal axis is the z-axis
- **Principal point (p)**: point where principal axis intersects the image plane (origin of normalized coordinate system)
Modeling projection

The coordinate system

- We will use the pin-hole model as an approximation
- Put the optical center (Center Of Projection) at the origin
- Put the image plane (Projection Plane) in front of the COP
- The camera looks down the negative z axis
  - we need this if we want right-handed-coordinates
Modeling projection

\[ (x, y, z) \rightarrow \left( f \frac{x}{z}, f \frac{y}{z} \right) \]
Homogeneous coordinates

Is this a linear transformation?
• no—division by z is nonlinear

Trick: add one more coordinate:

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]
\[(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \]

homogeneous image coordinates
homogeneous scene coordinates

Converting from homogeneous coordinates:

\[
\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)
\]
\[
\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)
\]
Pinhole camera model

\[(X, Y, Z) \mapsto (fX/Z, fY/Z)\]

\[
\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & X \\ f & 0 & Y \\ 1 & 0 & Z \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}
\]

\[x = PX\]
Pinhole camera model

\[
\begin{pmatrix}
    fX \\
    fY \\
    Z
\end{pmatrix} = \begin{bmatrix}
    f & 0 & 0 & 0 \\
    0 & f & 0 & 0 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
    X \\
    Y \\
    Z \\
    1
\end{pmatrix}
\]

\[x = PX\quad P = \text{diag}(f, f, 1)[I \mid 0]\]
Principal point offset

- Camera coordinate system: origin is at the principal point
- Image coordinate system: origin is in the corner

principal point: \((p_x, p_y)\)
Principal point offset

principal point: \((p_x, p_y)\)

\((X, Y, Z) \mapsto \left( \frac{f}{Z}X + p_x, \frac{f}{Z}Y + p_y \right)\)

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} \mapsto \begin{pmatrix}
\frac{f}{Z}X + Zp_x \\
\frac{f}{Z}Y + Zp_x \\
Z \\
1
\end{pmatrix} = \begin{bmatrix}
f & p_x & 0 \\
f & p_y & 0 \\
1 & 0 & 1
\end{bmatrix} \begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]
Principal point offset

principal point: \((p_x, p_y)\)

\[
\begin{pmatrix}
  fX + Zp_x \\
  fY + Zp_y \\
  Z
\end{pmatrix} = \begin{bmatrix} f & p_x \\ f & p_y \\ 1 \end{bmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}
\]

\[
K = \begin{bmatrix} f & p_x \\ f & p_y \\ 1 \end{bmatrix}
\]

calibration matrix

\[
P = K[I | 0]
\]
World Coordinates and Camera Coordinates

- Image point $(x_i, y_i, f)$
- Image plane
- $(x_s, y_s, z_s)$
- $(X_s, Y_s, Z_s)$
Camera Frame to World Frame

CO = T translation vector

CM = CO + OM

\[
\begin{align*}
x_s i + y_s j + z_s k &= T_x i + T_y j + T_z k + X_s I + Y_s J + Z_s K \\
x_s &= T_x + X_s I.i + Y_s J.i + Z_s K.i
\end{align*}
\]

\[
\begin{bmatrix}
x_s \\ y_s \\ z_s
\end{bmatrix}
= \begin{bmatrix}
T_x \\ T_y \\ T_z
\end{bmatrix} + \begin{bmatrix}
I.i & J.i & K.i \\
I.j & J.j & K.j \\
I.k & J.k & K.k
\end{bmatrix} \begin{bmatrix}
X_s \\ Y_s \\ Z_s
\end{bmatrix}
\]
Definition: 3D-Rotation

Linear Algebra

Definition: a matrix $R$ is a rotation matrix if and only if it is an orthogonal matrix with determinant $+1$.

Orthogonal Matrix: a square matrix with real entries whose columns and rows are orthogonal vectors with length 1.

That means: $RR^T = I$
Or: $R^{-1} = R^T$
3D Transformations - Rotation

- Euler Angles for Rotation $R = R_z R_y R_x$:

  rotation by $\psi$ about the z axis: $R_z = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$

  rotation by $\theta$ about the y axis: $R_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$

  rotation by $\phi$ about the x axis: $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}$

Sequence not standardized! Many different conventions!
Homogeneous Coordinates

\[
\begin{bmatrix}
    x_s \\
    y_s \\
    z_s \\
    1
\end{bmatrix}
= 
\begin{bmatrix}
    T_x \\
    T_y \\
    T_z \\
    1
\end{bmatrix}
+ 
\begin{bmatrix}
    I.i & J.i & K.i \\
    I.j & J.j & K.j \\
    I.k & J.k & K.k \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    X_s \\
    Y_s \\
    Z_s \\
    1
\end{bmatrix}
\]

Let \( C = -R^t \cdot T \)
Putting Everything Together

\[
\begin{bmatrix}
u' \\
v' \\
w'
\end{bmatrix} = \mathbf{K} \begin{bmatrix} I_3 & 0_3 \\ 0_3 & 1 \end{bmatrix}
\begin{bmatrix}
x_s \\
y_s \\
z_s \\
1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
u' \\
v' \\
w'
\end{bmatrix} = \mathbf{K} \begin{bmatrix} I_3 & 0_3 \\ 0_3 & 1 \end{bmatrix}
\begin{bmatrix}
\mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\
0_3^T & 1 \\
\end{bmatrix}
\begin{bmatrix}
\mathbf{X}_S \\
\mathbf{Y}_S \\
\mathbf{Z}_S \\
1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
u' \\
v' \\
w'
\end{bmatrix} = \mathbf{P}
\begin{bmatrix}
\mathbf{X}_S \\
\mathbf{Y}_S \\
\mathbf{Z}_S \\
1
\end{bmatrix}
\mathbf{x} = \mathbf{P}{\mathbf{X}}
Camera calibration

- Given $n$ points with known 3D coordinates $X_i$ and known image projections $x_i$, estimate the camera parameters.
Calibration

1. Estimate matrix $P$ using scene points and their images

2. Estimate the intrinsic and extrinsic parameters

$$P = KR \begin{bmatrix} I_3 & -\tilde{C} \end{bmatrix}$$

Left 3x3 sub-matrix is the product of an upper triangular matrix and an orthogonal matrix.
Computing the Matrix $P$

- Use corresponding image and scene points
  - 3D points $X_i$ in world coordinate system
  - Images $x_i$ of $X_i$ in image
- Write $x_i = P X_i$ for all $i$
Computing the Matrix $P$

- $x_i = P X_i$ involves homogeneous coordinates, thus $x_i$ and $P X_i$ just have to be proportional: $x_i \times P X_i = 0$

- Let $p_1^T, p_2^T, p_3^T$ be the 3 row vectors of $P$

\[
PX_i = \begin{bmatrix}
p_1^T X_i \\
p_2^T X_i \\
p_3^T X_i \\
\end{bmatrix}
\]

\[
x_i \times PX_i = \begin{bmatrix}
v'_i p_3^T X_i - w'_i p_2^T X_i \\
w'_i p_1^T X_i - u'_i p_3^T X_i \\
u'_i p_2^T X_i - v'_i p_1^T X_i \\
\end{bmatrix}
\]

\[
\Rightarrow \begin{bmatrix}
0_4^T \\
w'_i X_i^T \\
-v'_i X_i^T \\
\end{bmatrix}
\begin{bmatrix}
-v'_i X_i^T \\
u'_i X_i^T \\
0_4^T \\
\end{bmatrix}
\begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
\end{bmatrix}
= 0
\]

$\begin{bmatrix}
p_1 \\
p_2 \\
p_3 \\
\end{bmatrix}$ is a $12 \times 1$ vector
Computing the Matrix $P$

- Third row can be obtained from sum of $u_i'$ times first row - $v_i'$ times second row

$$\begin{bmatrix}
0^T_4 & -w_i' X_i^T & v_i' X_i^T \\
-w_i' X_i^T & 0^T_4 & -u_i' X_i^T \\
-v_i' X_i^T & u_i' X_i^T & 0^T_4
\end{bmatrix}\begin{bmatrix}
p_1 \\
p_2 \\
p_3
\end{bmatrix} = 0$$

- So we get 2 independent equations in 11 unknowns (ignoring scale)

- With 6 point correspondences, we get enough equations to compute matrix $P$

$$Ap = 0$$
Computing the Matrix $P$

- Linear system $A \ p = 0$
- When possible, have at least 5 times as many equations as unknowns (28 points)
- Minimize $\| A \ p \|$ with the constraint $\| p \| = 1$
  - $P$ is the unit singular vector of $A$ corresponding to the smallest singular value (the last column of $V$, where $A = U \ D \ V^T$ is the SVD of $A$)
Further Improvement

Use as initialization for nonlinear minimization of $\sum d(x_i, PX_i)^2$

Most popular non-linear minimization algorithm is the Levenberg-Marquardt minimization

LM is more robust to local minima than e.g. the Gauss–Newton algorithm and the method of gradient descent.
Iterative minimization

General nonlinear problem formulation for n measurements:

\[ \hat{x} = \arg \min_x \sum_{i=1}^n r(i) \]

Concrete problem of camera pose estimation from 2D/3D correspondences

- Parameters to estimate: camera pose (rotation and translation)

\[
(R, t) = \arg \min_{\delta(R,t)} \sum_i \text{dist}(\text{proj}(m_{wi};(R,t)), m_{ni})
\]
Least squares (LS) estimation

Typical problem formulation (error in normalized image space):

\[
\hat{s} = \underset{s}{\text{arg min}} \sum_{i=1}^{n} r^{(i)LS}_n = \underset{s}{\text{arg min}} \sum_{i=1}^{n} \| d_n(m^{(i)}_n, m^{(i)}_w, s) \|^2
\]

All measurements are incorporated with equal weight

If information about the quality of measurements is given, how to incorporate this?
Robust estimation (M-estimators)

Minimize the sum of a function of residuals:

$$\hat{s} = \arg \min \limits_{s} \sum_{i=1}^{n} \rho(r^{(i)})$$

How could a simple robust error function be constructed?

$$\rho(r) = \begin{cases} |r|, & \text{if } |r| < r_{max} \\ r_{max}, & \text{if } |r| \geq r_{max} \end{cases}$$

Assumptions:
- Only few outliers
- Initialization close to solution

Typical M-estimators: TUKEY, Cauchy, Huber, ...
Der Tukey- Estimator

\[ \rho_{\text{Tuk}}(x) = \begin{cases} \frac{c^2}{6} \left[ 1 - \left(1 - \left(\frac{x}{c}\right)^2\right)^3 \right] & |x| \leq c \\ \frac{c^2}{6} & |x| > c \end{cases} \]
The Tukey Estimator

The threshold $c$ depends on the standard deviation of the error function, e.g. $c = 4\sigma$

Correspondences with $|x| \leq c$ influence the minimization results.

Correspondences with $|x| > c$ are outliers and have no influence.
Outline

• Part I: Camera model and pose computation

• Part II: 3D contour tracking (RAPID)
Contour-based tracking

Definition of contour-based tracking

A procedure to estimate the pose of an object by using contours, selected from a full wire-frame (CAD) model

**Model contours** = boundaries along the object surface, that we expect to see in the image, from a given viewpoint

Giorgio Panin
Examples of model contours

Rendered view  CAD model  Model contours

Giorgio Panin
Contours vs. keypoints

Advantages

1. Edges are usually well-defined (sharp transitions), for many different poses, light conditions, etc.

   Advantage: more robustness and precision w.r.t. pose and light

2. We can obtain good tracking with low computational time, because contours are rather easy to detect and to track (while point features require complex selection-description-matching)

   Advantage: faster detection and tracking (real-time guaranteed)

Giorgio Panin
Contour vs. Point-based Tracking

3. Contours depend only on the object shape (geometry), while feature points require a distinctive surface appearance (not good for uniform colors or textures)

Advantage: we can track a large class of objects or environment items, that have no feature points on the surface
Contour vs. Point-based Tracking

**Disadvantages**

1. It works only if the 3D object has a *sufficiently asymmetric* shape, in order to identify the pose *without ambiguity*.

   - **Cube**: ambiguous, because of symmetries (multiple solutions).
   - **Revolving surface**: cannot estimate axial rotation (infinite solutions).

**Giorgio Panin**
Contour vs. Point-based Tracking

Disadvantages

2. Edges are less distinctive features than key-points

They have a weak identity
Less robust matching: we may easily follow the wrong edge!

Example: two edges parallel and close together – which one is from the model?

Giorgio Panin
Goal
Computer-graphics (CG) and computer vision CV

Goal: to work in a and feature space common to CV and CG representation of the object
The RAPiD algorithm

Edge-based tracking = the „RAPiD“ algorithm [Harris, 1992]

**RAPiD = Real-time Attitude and Position Determination**

**A.** Pre-processing: compute the edge map

**B.** Iterate two steps: 1. data association, 2. pose update
A. Computing the edge map: Contour features

Existing type of edges – which ones are good to track?

**Crease edges:** have a sharp angle (e.g. polyhedral faces)
**Boundary edges:** define the border of a planar part of the surface
**Silhouette edges:** define the horizon of a round part of the surface
**Texture edges:** separate different texture parts (reflectance properties)
Contour model: visible contours

Use computer graphics techniques
Generating a 3D contour model

The edges can be computed in once at initialization but also in real-time during tracking using the graphics card.
Using computer graphics

Texture / Material Edges:

- Render into texture with ambient lighting
- Edge image
- Store z-value in alpha-channel
- 2½D edge image
B. Association and pose computation

The 3D edge model is projected onto the image, using the last pose.
Control points $p_i$ are defined at regular distance for each edge $E$ of the model.
B. Association and pose computation

An orthogonal search segment is defined at each $p_i$. Candidates $q_{i,j}$ are defined along this segment.
B. Association and pose computation

Single hypothesis solution:

\( p_i \) thecontrol point and \( q_i \) the detected point in the image

The following error function is minimized:

\[ err = \sum_i \rho_{Tuk}(\Delta(p_i, q_i)) \]

With \( \rho_{Tuk} \) the Tukey Estimator Function.

The distance function \( \Delta \) is defined as:

\[ \Delta(p_i, q_i) = \left| (q_i - p_i) \cdot n_i \right| \]

With \( n_i \) die normal vector of the projected line.
B. Association and pose computation

Problem with the single hypothesis solution

The „first“ gradient in the image is not necessary the right one. Ambiguity for complex objects and cluttered background.

-> object edges are stuck on the wrong image edges.

Idea: Multiple hypotheses

- Take into account several hypothesis
- For a given control point $p_i$ several $q_{ij}$ are considered.
- The choice of the correct candidate happens implicitly during the minimization

(Vacchetti & Lepetit’04)
B. Association and pose computation

For multiple hypothesis, we have:

\[ p_i \] the control points of the edge and \( q_{i,j} \) seien multiple potential candidates.

The following error function is minimized:

\[ err = \sum_i \rho_{Tuk} \left( \min_j \Delta(p_i, q_{i,j}) \right) \]

The Estimator-Function is computed for the hypothesis, which has the smallest distance to the projected point.
B. Association and pose computation

Summary multiple hypothesis:

- More computation expensive, but still real-time
- More robust!
- Ambiguity are solved through consideration of multiple candidates.
Results
Results

Thank you!