Model-based Visual Tracking: the OpenTL framework

Giorgio Panin
Contents

• Object tracking: theory

• Building applications with OpenTL
Object tracking
Goal: Multi-Target / -Sensor / -Modal localization
Model-based tracking

- Model
- Visual processing
- Localization (tracking)

Video surveillance

- Control/navigation
- Face tracking

G. Panin 27.04.2010
Ground shape

O₁

O₂

w
# Pose parameters – single-body transforms

<table>
<thead>
<tr>
<th></th>
<th>Base</th>
<th>Euclidean</th>
<th>Similarity</th>
<th>Affine</th>
<th>Homography</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2D</strong></td>
<td><img src="image" alt="Base 2D" /></td>
<td><img src="image" alt="Euclidean 2D" /></td>
<td><img src="image" alt="Similarity 2D" /></td>
<td><img src="image" alt="Affine 2D" /></td>
<td><img src="image" alt="Homography 2D" /></td>
</tr>
<tr>
<td><strong>3D</strong></td>
<td><img src="image" alt="Base 3D" /></td>
<td><img src="image" alt="Euclidean 3D" /></td>
<td><img src="image" alt="Similarity 3D" /></td>
<td><img src="image" alt="Affine 3D" /></td>
<td><img src="image" alt="Homography 3D" /></td>
</tr>
<tr>
<td><strong>Invariant properties</strong></td>
<td>Distances</td>
<td>Angles</td>
<td>Parallel lines</td>
<td>Straight lines</td>
<td>Parallel lines</td>
</tr>
</tbody>
</table>
Pose parameters – articulated body

\[ \bar{x}_W = T_{W,l_0} \bar{x}_{l_3} \]

\[ T_{l_0,l_1} \]

\[ T_{l_1,l_2} \]

\[ T_{l_2,l_3} \]
Pose parameters – deformable shapes
Active Shape Model (2D) – face tracking

Learning deformation modes from examples (PCA)

Shape, \( V \)

\[ V_0 + p_1 V_1 + p_2 V_2 + p_3 V_3 + \ldots \]
Object appearance

Key-frames

Texture map

Active Appearance Model

\[ A = A_0 + a_1 A_1 + a_2 A_2 + a_3 A_3 + \ldots \]
Object dynamics

2nd order, Auto-Regressive model

Damped-spring motion

\[ x_t - \bar{x} = F_1(x_{t-1} - \bar{x}) + F_2(x_{t-2} - \bar{x}) + W_0 w_t \]

Motion type: specified by three main parameters

- **Average state:** \( \bar{x} \)
- **Oscillation frequency:** \( f \)
- **Damping rate:** \( \beta \)
Object dynamics - examples

Brownian Motion (f, β undefined)

\[ F^1 = 1 \quad F^2 = 0 \quad W^0 = 0.1 \]

White Noise Acceleration

\[ f = 0 \quad \beta = 0 \]

\[ F^1 = 2 \quad F^2 = -1 \quad W^0 = 0.1 \]

Periodic, undamped:

\[ f = 0.1 \quad \beta = 0 \]

\[ F^1 = 1.99 \quad F^2 = -1 \quad W^0 = 0.1 \]

Periodic, damped:

\[ f = 0.1 \quad \beta = 0.05 \]

\[ F^1 = 1.98 \quad F^2 = -0.99 \quad W^0 = 0.88 \]

Aperiodic (critically damped):

\[ f = 0 \quad \beta = 0.5 \]

\[ F^1 = 1.9 \quad F^2 = -0.9 \quad W^0 = 2.12 \]
Object dynamics – multi-dim.  \[ s_t = Fs_{t-1} + (I - F)s + Ww_t \]

Constrained

\[ F = \begin{pmatrix} 1.9I & -0.9I \\ I & 0 \end{pmatrix} \]

\[ W = \begin{pmatrix} 2.12I \\ 0 \end{pmatrix} \]

Unconstrained

\[ F = \begin{pmatrix} I & 0 \\ I & 0 \end{pmatrix} \]

\[ W = \begin{pmatrix} 2.12I \\ 0 \end{pmatrix} \]
Object model

Shape

Appearance

Pose

Dynamics
Camera model

Intrinsic parameters

Pin-hole model

Radial distortion
Camera model

Extrinsic parameters
Camera model

Object-to-image projection (and back-projection)
Visual modalities

- Shape moments
- Intensity gradients
- Contour lines
- Color statistics

- Texture template
- Optical flow
- Local keypoints

(and others: Background subtraction / CCD / Harris keypoints / Histogram of oriented gradients / SIFT)
The „tracking pipeline“

- Data acquisition
- Pre-processing
- Rendering view
- Matching
- Targets update
- On-line features update
- Output

- Targets prediction
- Off-line features sampling
- Data fusion
- New features

- Prediction
- Rendering model features
- Back-projection

- Data acquisition
- Image features
- Pre-processing
- Matching
- Re-projection
- Update model features
Abstraction: visual modality processing

Rule: any *modality* class must implement

- Model free pre-processing
- Off-line and on-line features sampling and back-projection
- Multi-level data association (Pixel-, Feature-, State-space)
- Residuals, covariances and Jacobians computation

Additional classes:

- Multi-modal, multi-sensor data fusion (cascade, parallel)
- Likelihood computation
Features sampling – GPU assisted
Multi-level visual processing

\[ h(s) \]
\[ z \]
\[ e = z - h \]

Pixel-level measurement

Feature-level measurement

Object-level measurement

\[ z = s^* \text{ (LSE estimate)} \]

\[ h = s^- \text{ (predicted pose)} \]
Measurement: pixel- vs. feature-level

Analogy with fluid mechanics

Eulerian = pixel-level

Lagrangian = feature-level

Dense optical flow (HS)

Sparse optical flow (LK)
Feature-level: re-projection vs. tracking

Model feature re-projection

Feature tracking (= „flow“)

Features sampling

Features sampling

Incorrect matching

Incorrect tracking
Feature-level: validation gates (local search)

Prior density (state-space)

$$(s^-, P^-)$$

Innovation densities (measurement space)

$$(y_i^-, S_i)$$

$x_1, x_2, x_3$
Example: color histograms

Pixel-level matching

Feature-level matching
= histogram distance

Object-level matching
= mean-shift optimization
Example: intensity edges

Pixel-level

Draw the silhouette

Match silhouette to the Distance Transform

Feature-level

sample contour points

Re-project and search in the image
Building in OpenTL

Multi-camera, multi-level data fusion

View 1
- Color segmentation
- Motion

View 2
- Edges
- Keypoints

Weighted Average

Blobs

Joint MLE

Joint likelihood

Joint MLE

\[ P(Z | s^-) = P(Z_{blobs} | s^-) \cdot P(Z_{obj-MLE} | s^-) \]
Building processing trees in OpenTL

Generalization of the tree

\[
P(Z \mid s^-)
\]
Data fusion – multi-modal

Benefits:

- Combine independent information sources
- Increase robustness (tracking fails if ALL modalities fail)

Drawbacks:

- Need to define a proper fusion scheme, and parameters
- Higher computational effort → slower frame rate
Data fusion – multi-camera

Complimentary setup

Redundant setup
Data fusion – multi-camera

Complimentary setup: Indoor people tracking

Redundant setup: 3D hand tracking
Multi-target – occlusion handling

Pixel-level

Model #1
Model #2
Model #3

Same model
Different models
Target #1
Target #2
Target #3
Target #4
Target #5

Data
(multi-class segmentation)

Feature-level

h₁
z₁
...
h₅
z₅

#1
#2
#3
#4
#5
Bayesian Tracking: prediction - correction

Gaussian filters

- (Extended) Kalman filter
- Information filter
- Unscented Kalman/Information filter

Monte-Carlo filters

- S-I-R particle filter
- MCMC particle filter
Representing the target distribution

Prior density $P(s_t \mid z_{t-1}, ..., z_0)$

Posterior density $P(s_t \mid z_t, z_{t-1}, ..., z_0)$
Flow diagram of OpenTL-based applications

- Track Initiation:
  - $I_t$ → Detection/Recognition
  - $Obj_{t-1}^+$
  - $Obj_{t-1}$

- Track Maintainance:
  - $t$
  - $I_t$
  - Local processing
  - $Meas_t$
  - $Obj_t^-$
  - Bayesian tracking
  - $Obj_t$
  - Post-processing

- Models:
  - Shape
  - Appearance
  - Degrees of freedom
  - Dynamics
  - Sensors
  - Environment

- $\Delta t$
Object detection (examples)

- General-purpose Monte-Carlo sampling in state-space
- People detection based on foreground blobs clustering
- Invariant keypoints matching (for textured objects)
- Marker detection based on intensity edges
- Hand detection based on color and edge lines
- Face detection based on a trained classifier (with Haar features)
Resume: model-based object tracking
Building applications with OpenTL
Features of OpenTL

- Modularized, object-oriented software architecture
- Common abstractions for layers
- Real-time performance
- Different Bayesian filters
- A large variety of visual modalities, with multiple processing levels
- Robust improvements (multi-hypotheses, data fusion, …)
- Generic sensor abstraction
- Support multi-camera, multi-target and multi-modal applications
- Support for GPU acceleration
Application: planar object tracking

Single-target, single-camera, color-based

Meas. processing

Color histograms → Likelihood $P(Z_{\text{feat}} | s^-)$ → Particle Filter

Tracking
Application: planar object tracking

Fusion color+background (pixel-level), and blob detection
People tracking with distributed cameras

- Distributed, multi-camera setup
- People detection and tracking in real-time
- 3D localization
- Goal: Human-Robot Interaction (coffee-break scenario)
Hierarchical grid-based localization
Stereo tracking of a quadcopter
Application: face tracking

Cascade of two pipelines, with 3D pose upgrade

2D→3D pose upgrade

Template matching

Object-level upgrade

Kalman Filter

2D tracking

3D model

3D tracking
Application: 3D hand tracking

Input

Pre-process

Modality processing

Cam Left

Cam Top

Cam Right

Data Fusion: upgrade to object Level

feature parts

non feature parts

G. Panin
Our Tracking Projects

- Pedestrian Tracking
- Vivid cell tracking
- Human Robot Interaction
- VR TV Studio Automation
- Quadcopter tracking
- ITrackU (CCRL)
Our Team

- graduate and post-doc coworkers
- student coworkers
- manifold research projects
- funded by DFG, EU, industry partners
- focus on robust, real-world applications