Model-based Visual Tracking:

the *OpenTL* framework

Giorgio Panin

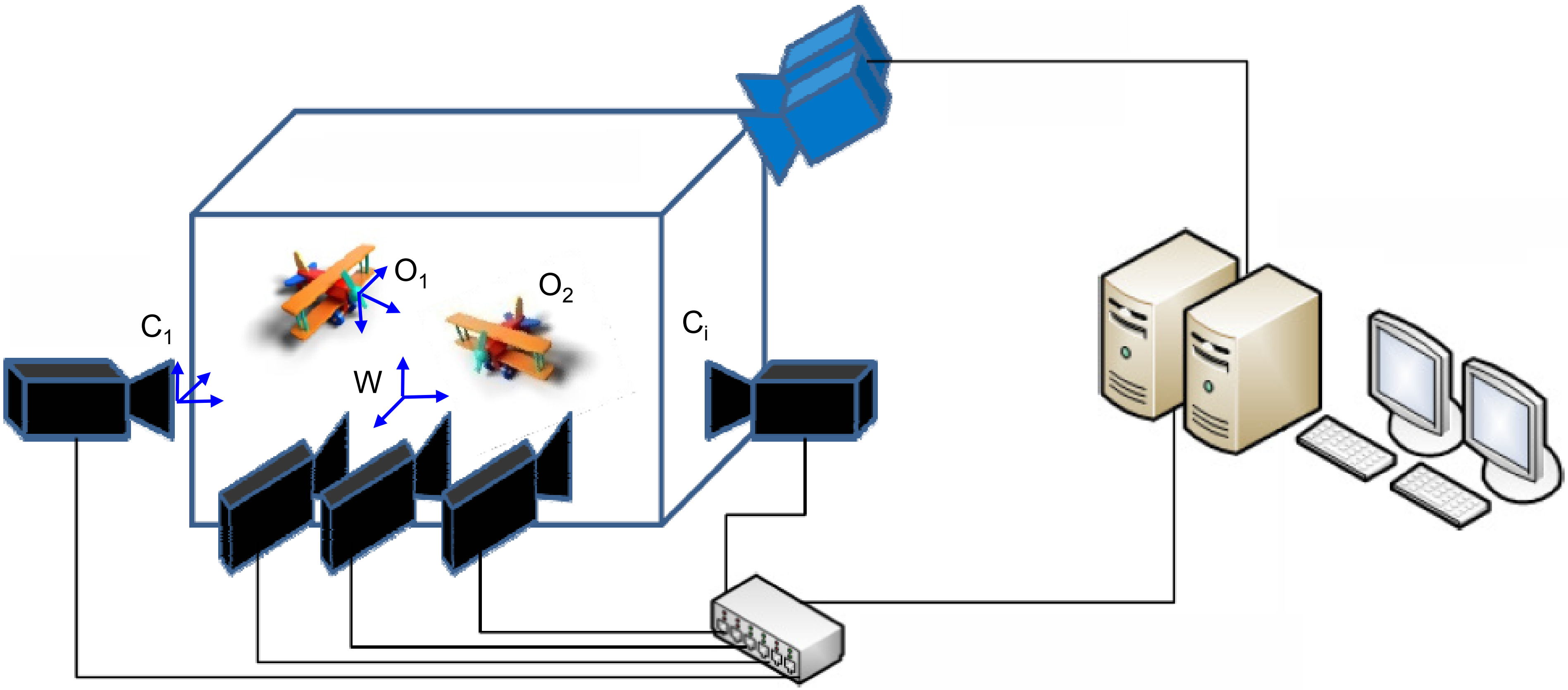
Technische Universität München · Institut für Informatik
Lehrstuhl für Echtzeitsysteme und Robotik (Prof. Alois Knoll)

Contents

- Object tracking: theory
- Building applications with OpenTL

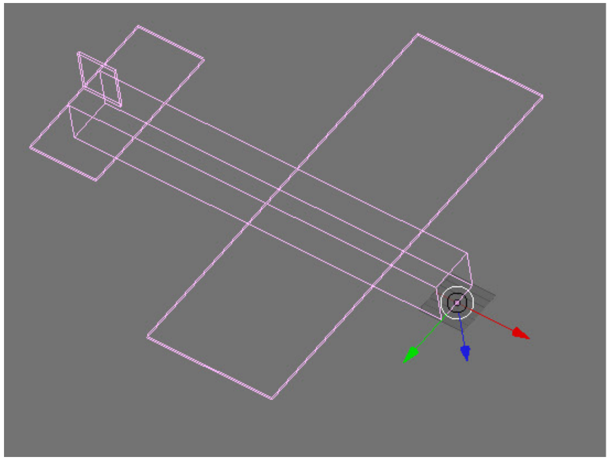
Object tracking

Goal: Multi-Target / -Sensor / -Modal localization

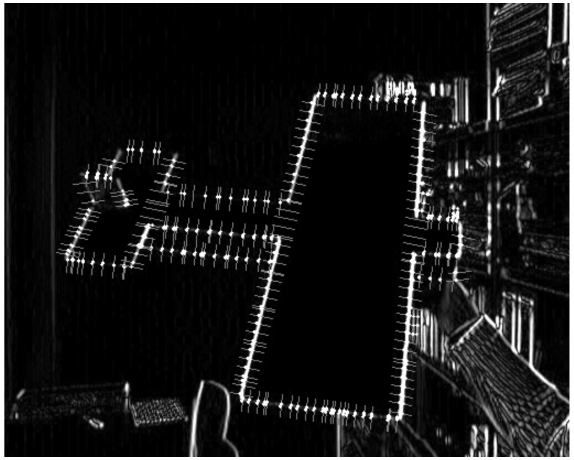


Model-based tracking

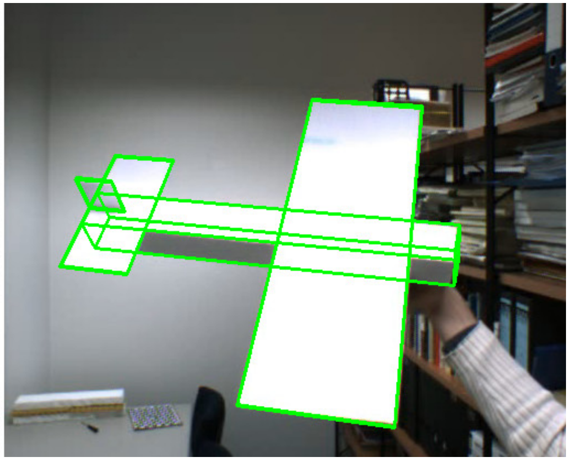
Model



Visual processing

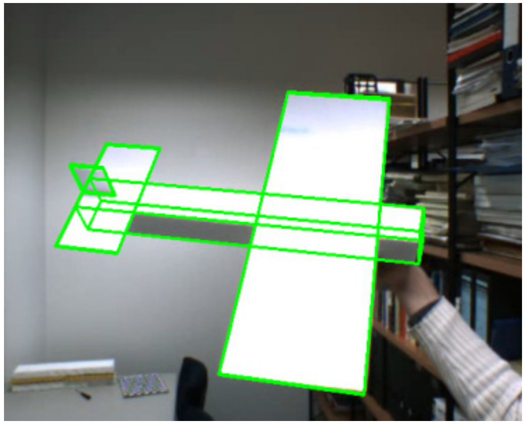


Localization (tracking)

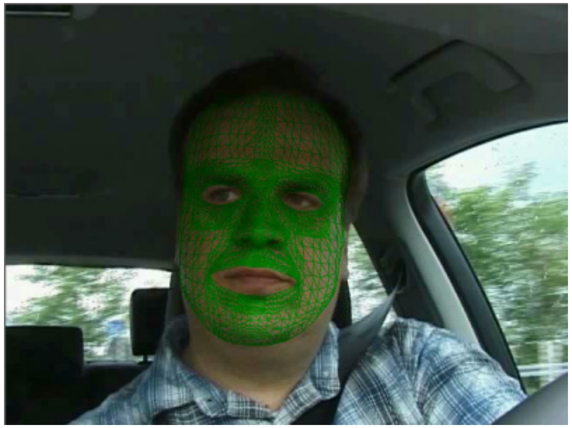


Video surveillance

Control/navigation

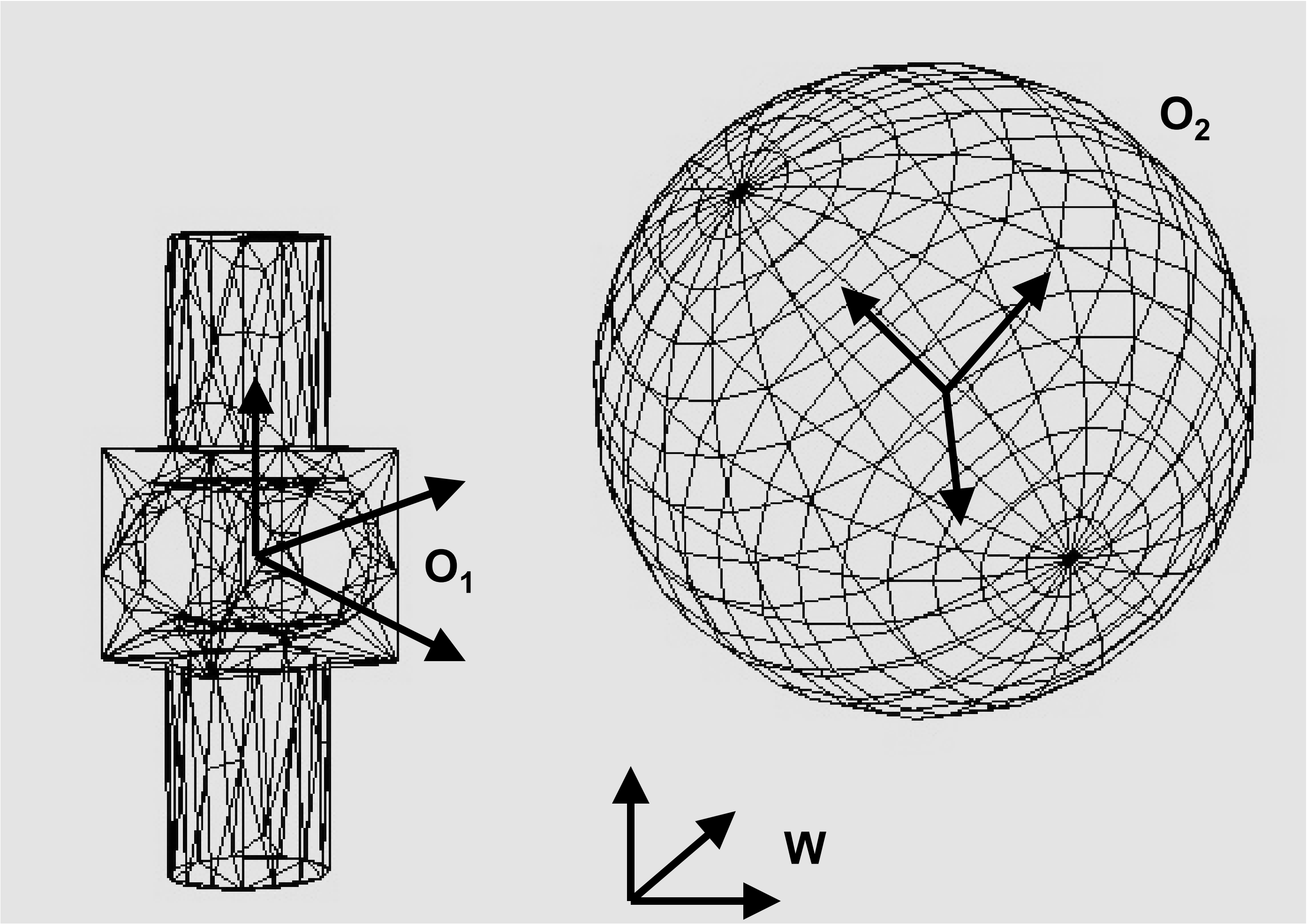


Face tracking

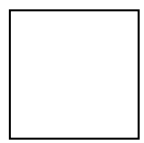
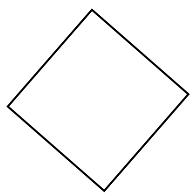
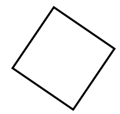
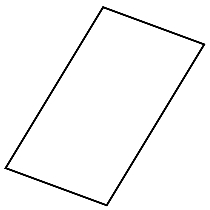
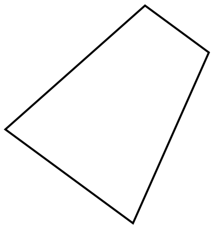
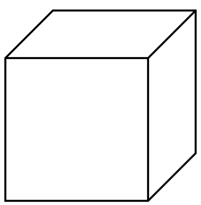
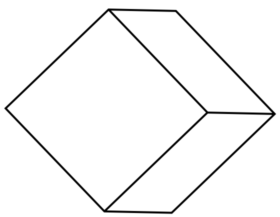
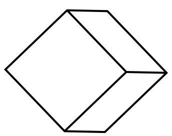
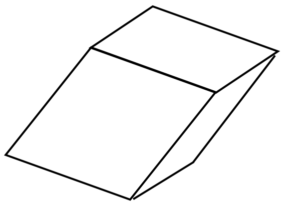
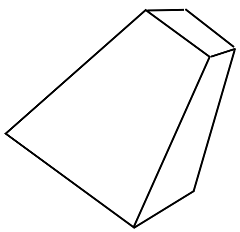


…

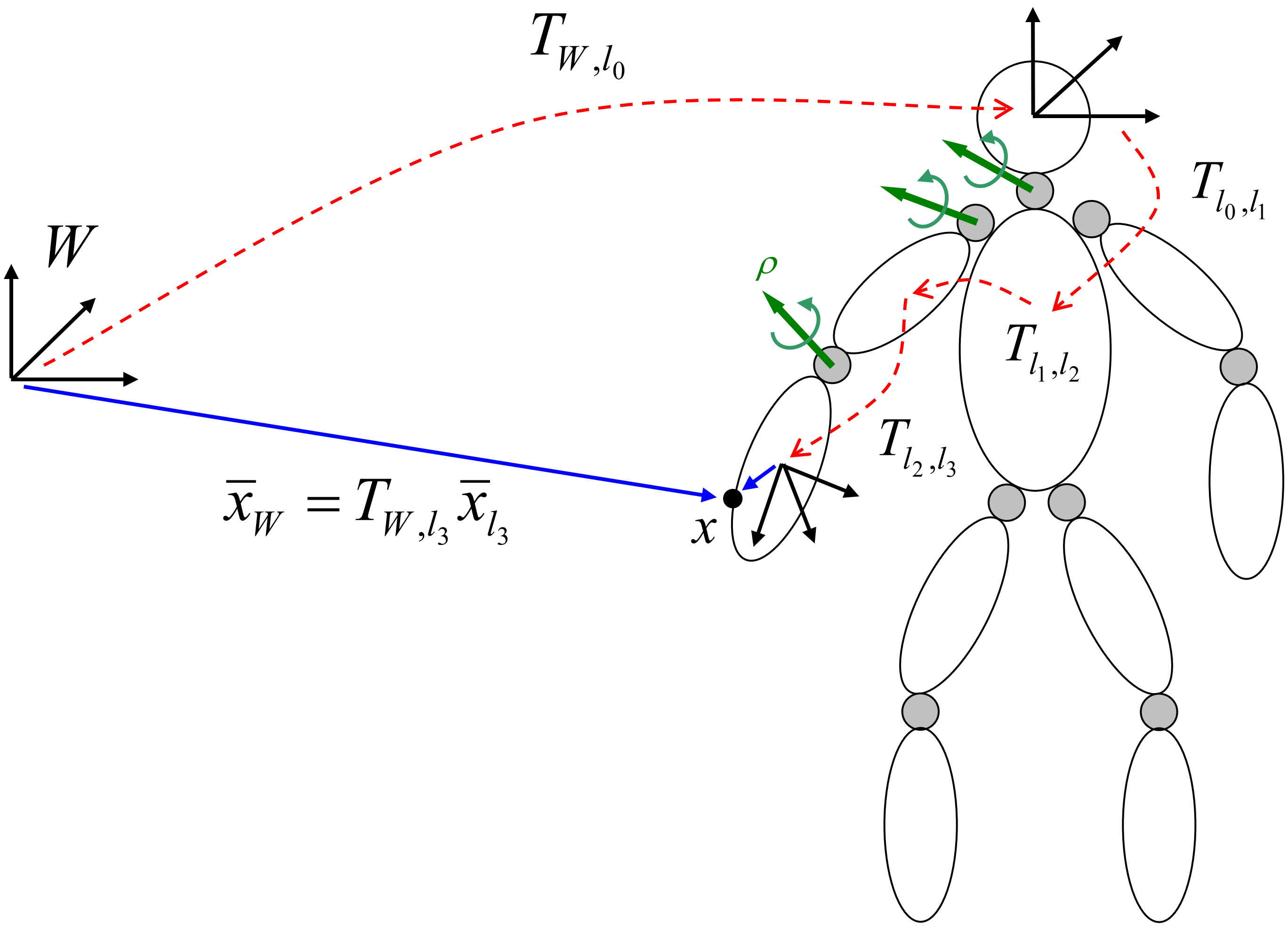
Ground shape



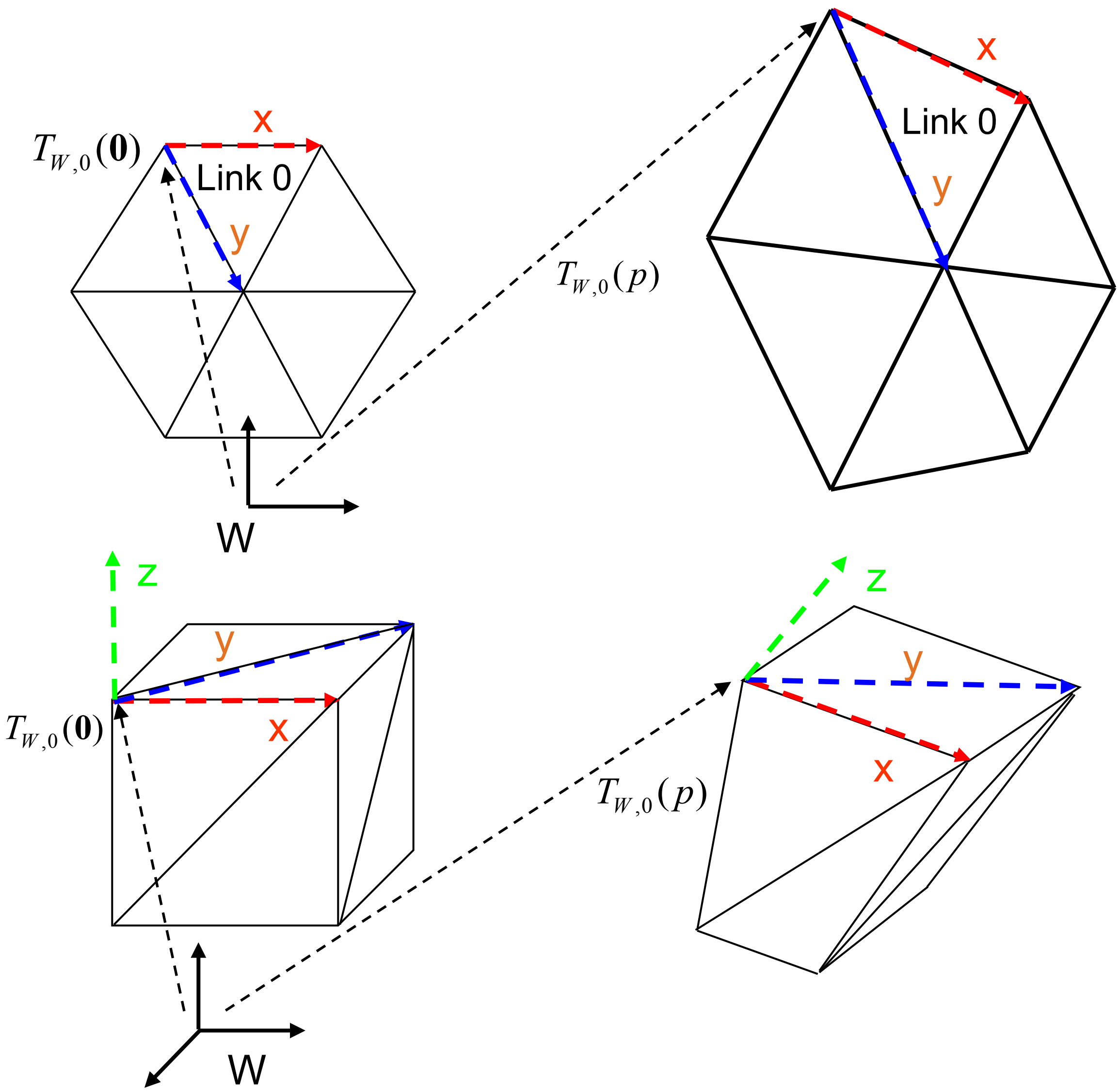
Pose parameters – single-body transforms

	Base	Euclidean	Similarity	Affine	Homography
2D					
3D					
Invariant properties		Distances Angles Parallel lines Straight lines	Angles Parallel lines Straight lines	Parallel lines Straight lines	Straight lines

Pose parameters – articulated body

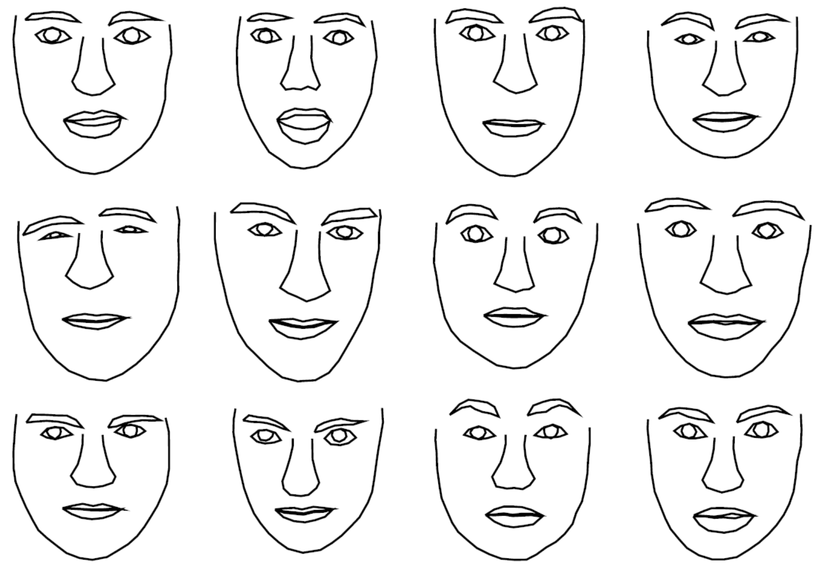


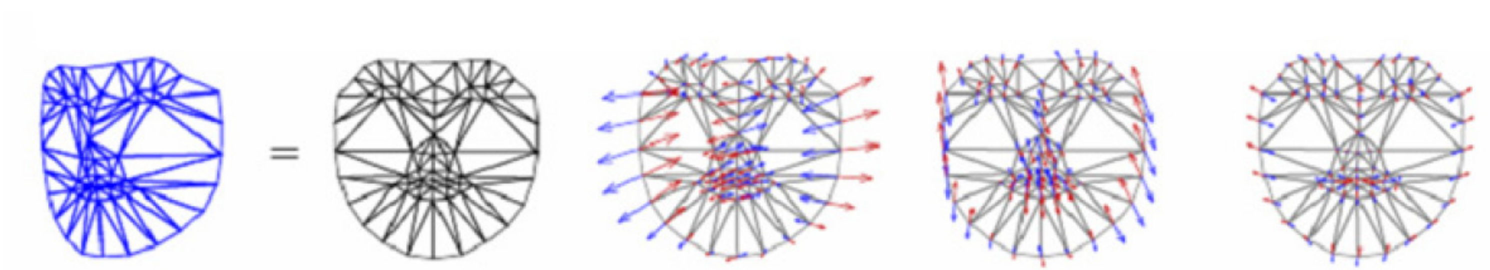
Pose parameters – deformable shapes



Active Shape Model (2D) – face tracking

Learning deformation modes from examples (PCA)

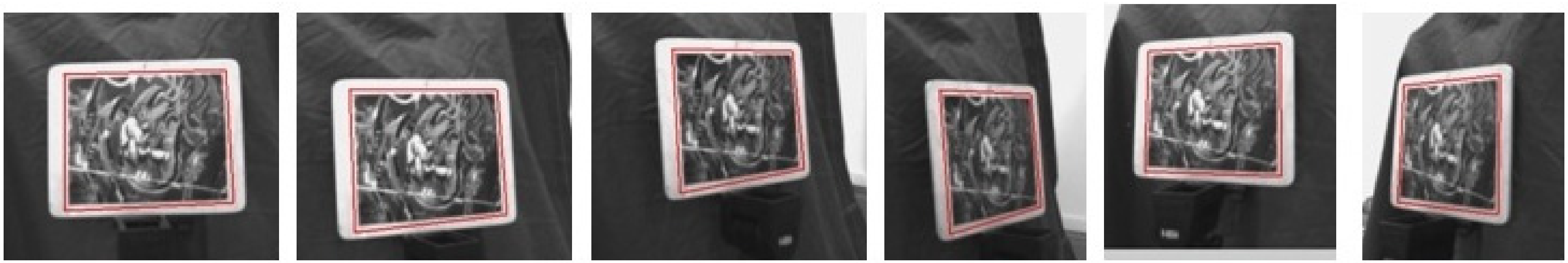




Shape, $V = V_0 + p_1V_1 + p_2V_2 + p_3V_3 \ldots$

Object appearance

Key-frames



Texture map

Active Appearance Model

Appearance, A $\quad A_0 \quad + \quad a_1A_1 \quad + \quad a_2A_2 \quad + \quad a_3A_3 \quad + \quad \cdots$

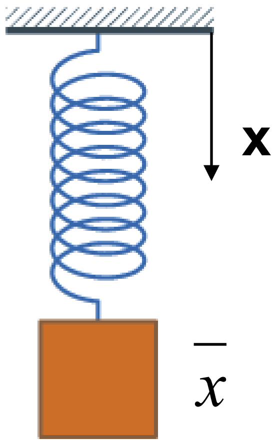
Object dynamics

2nd order, Auto-Regressive model

Damped-spring motion

Process noise

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



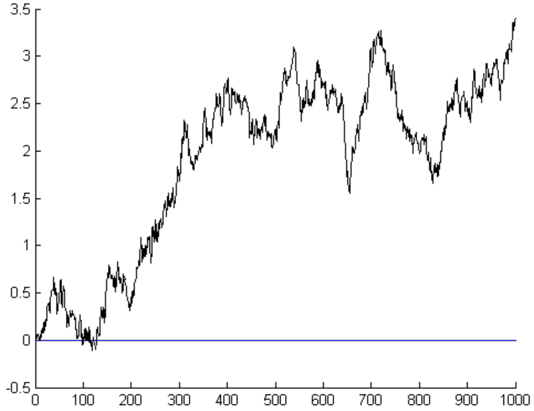
Motion type: specified by three main parameters

- **Average state:** $\bar{x}$
- **Oscillation frequency**: f
- **Damping rate**: β

Object dynamics - examples

Brownian Motion $(f, \beta \text{ undefined})$

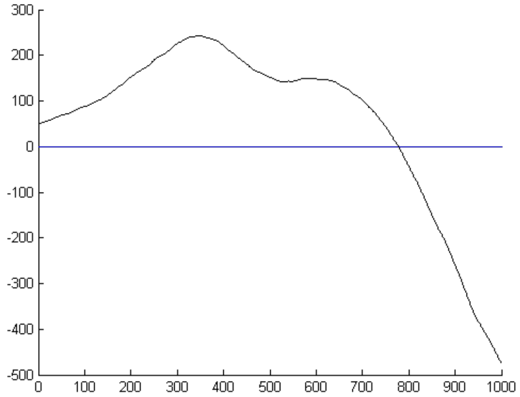
$F^1 = 1 \quad F^2 = 0 \quad W^0 = 0.1$



White Noise Acceleration

$f = 0 \quad \beta = 0$

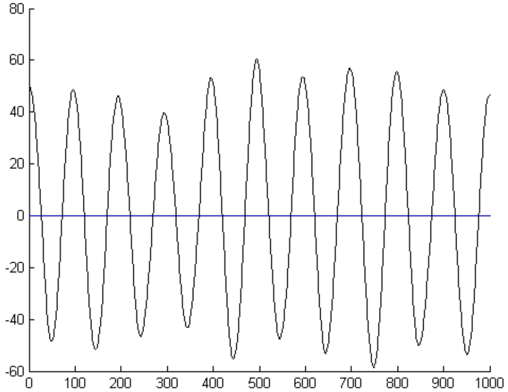
$F^1 = 2 \quad F^2 = -1 \quad W^0 = 0.1$



Periodic, undamped:

$f = 0.1 \quad \beta = 0$

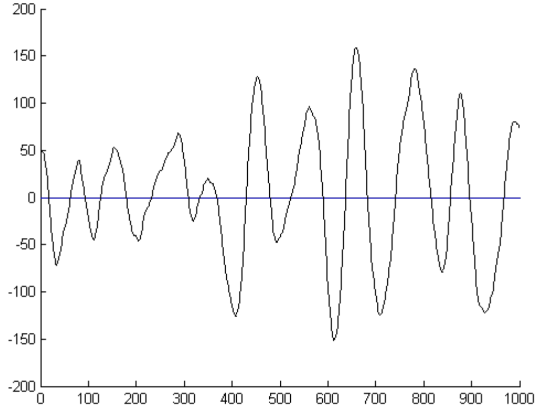
$F^1 = 1.99 \quad F^2 = -1 \quad W^0 = 0.1$



Periodic, damped:

$f = 0.1 \quad \beta = 0.05$

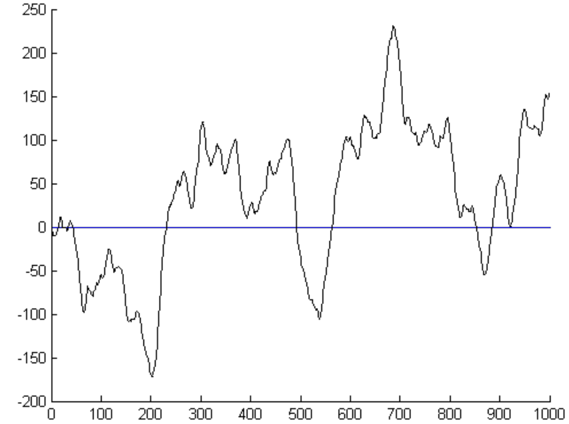
$F^1 = 1.98 \quad F^2 = -0.99 \quad W^0 = 0.88$



Aperiodic (critically damped):

$f = 0 \quad \beta = 0.5$

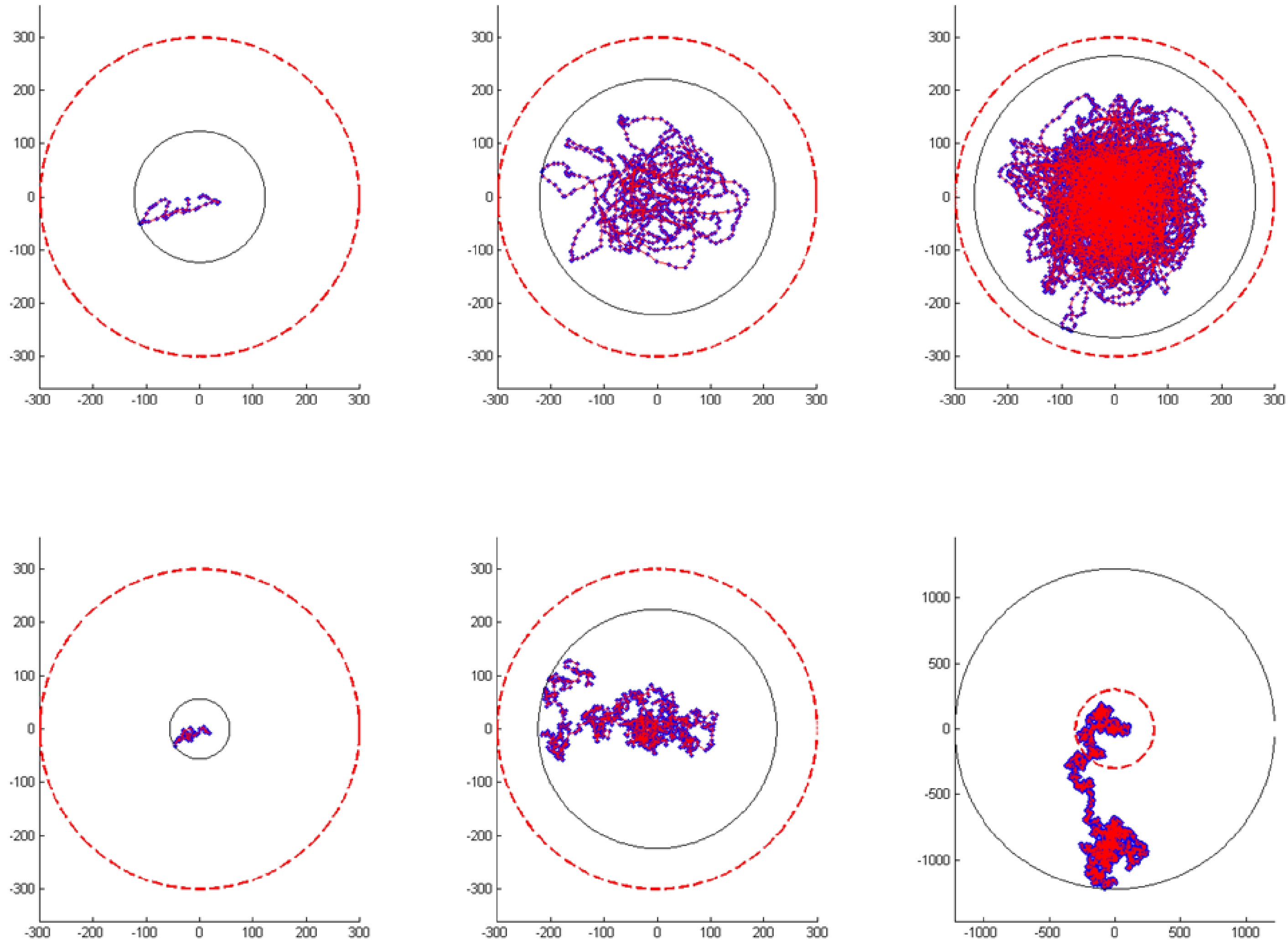
$F^1 = 1.9 \quad F^2 = -0.9 \quad W^0 = 2.12$



Object dynamics – multi-dim.

$$\mathbf{s}_t = F\mathbf{s}_{t-1} + (I - F)\bar{\mathbf{s}} + W\mathbf{w}_t$$

t=50 | t=1000 | t=10000



Constrained

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

$$W = \begin{pmatrix} 2.12I \\ 0 \end{pmatrix}$$

Unconstrained

$$F = \begin{pmatrix} I & 0 \\ I & 0 \end{pmatrix}$$

$$W = \begin{pmatrix} 2.12I \\ 0 \end{pmatrix}$$

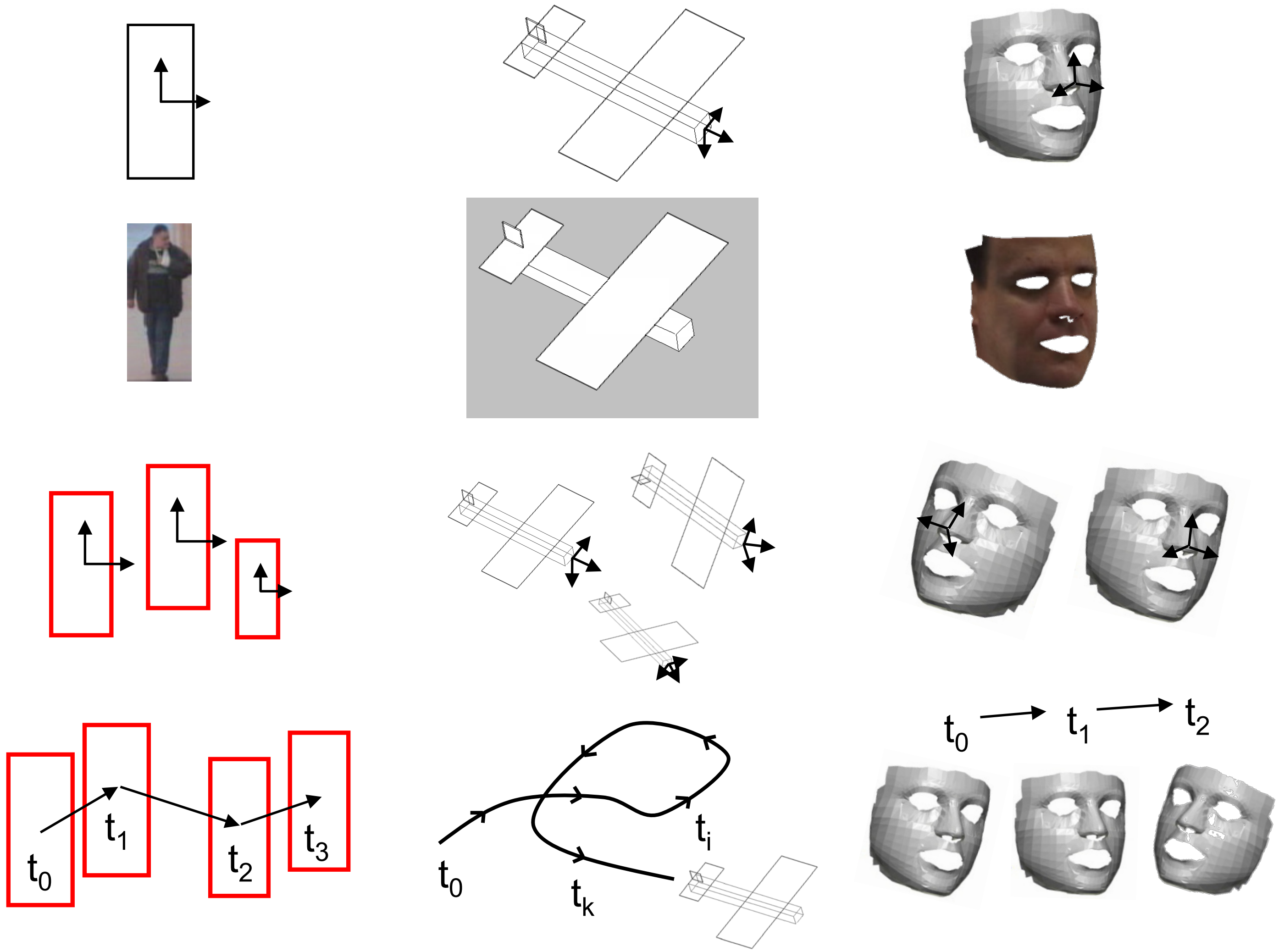
Object model

Shape

Appearance

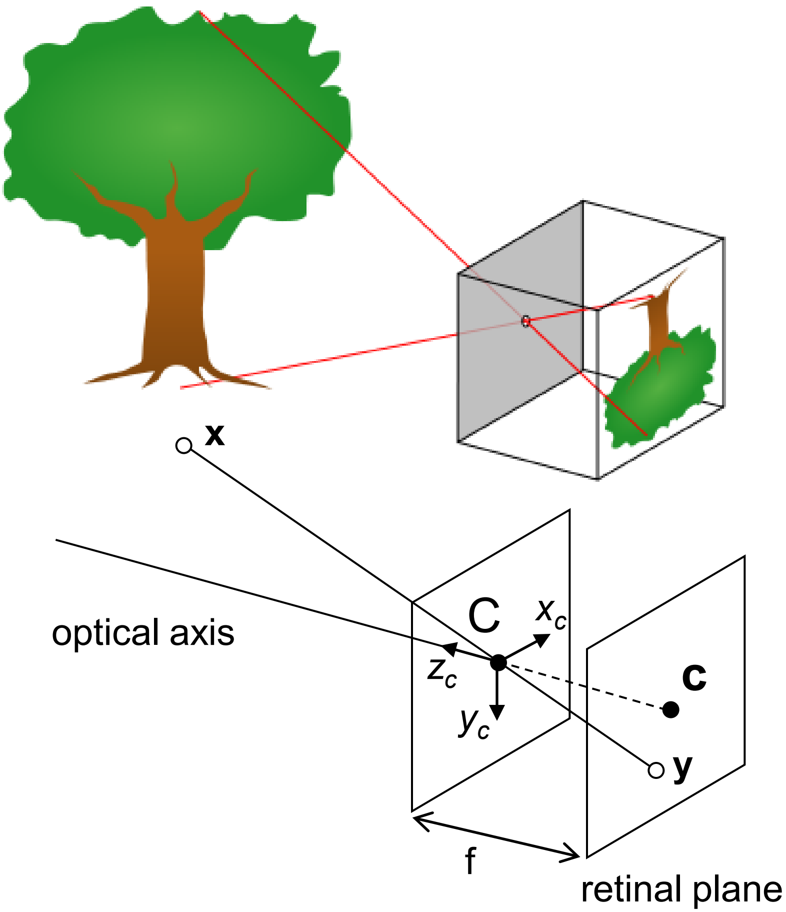
Pose

Dynamics

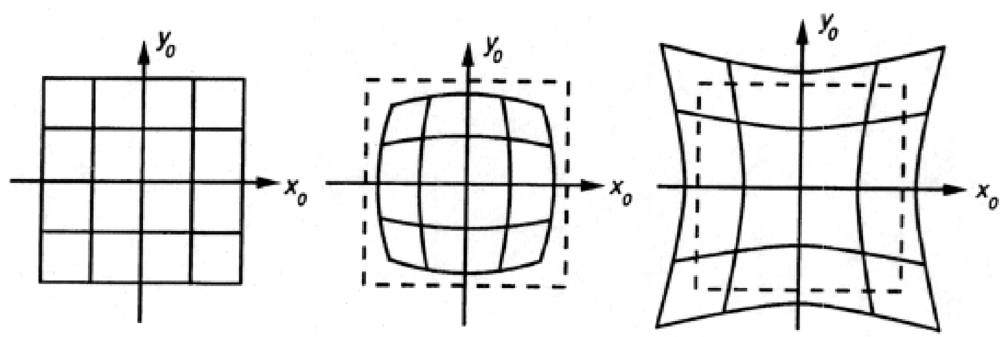


Intrinsic parameters

Pin-hole model

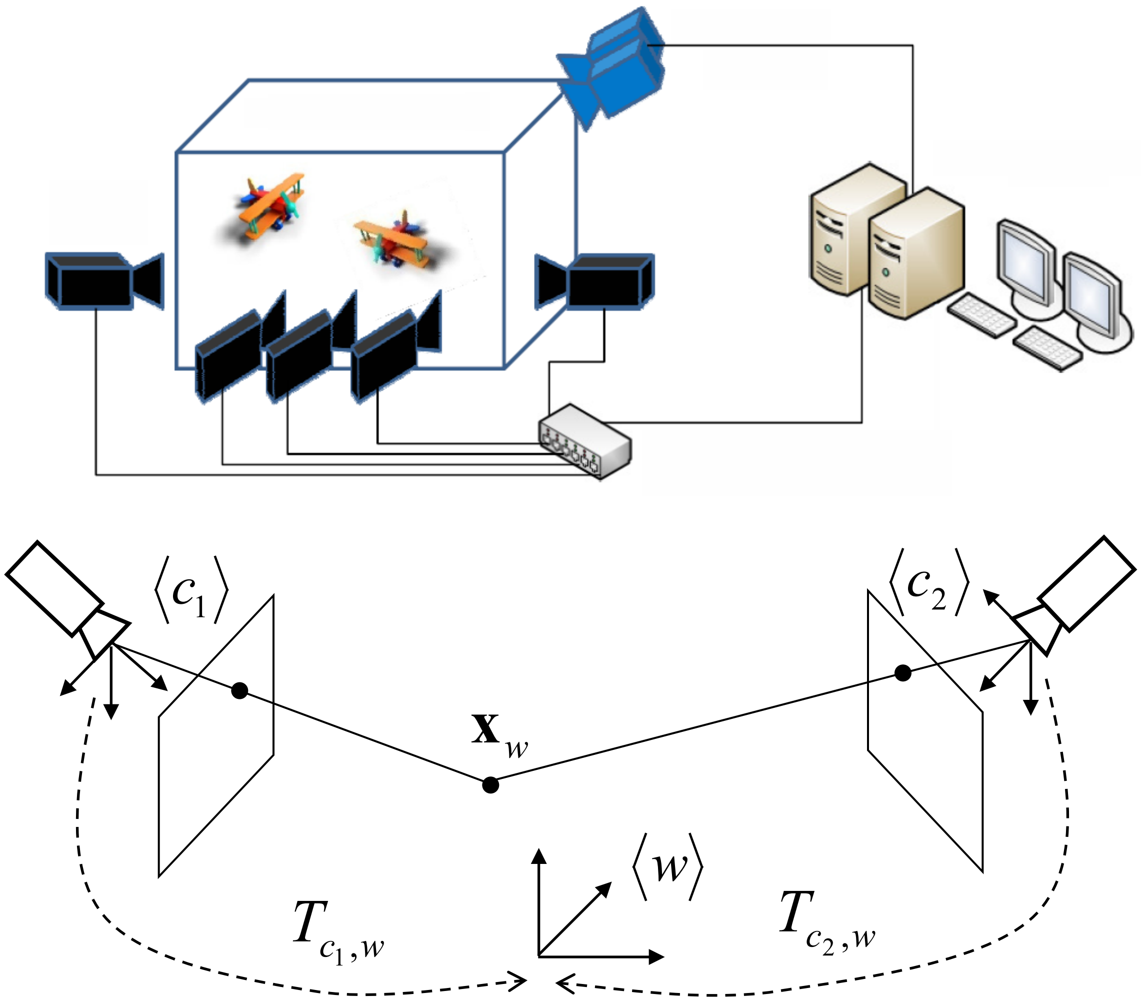


Radial distortion



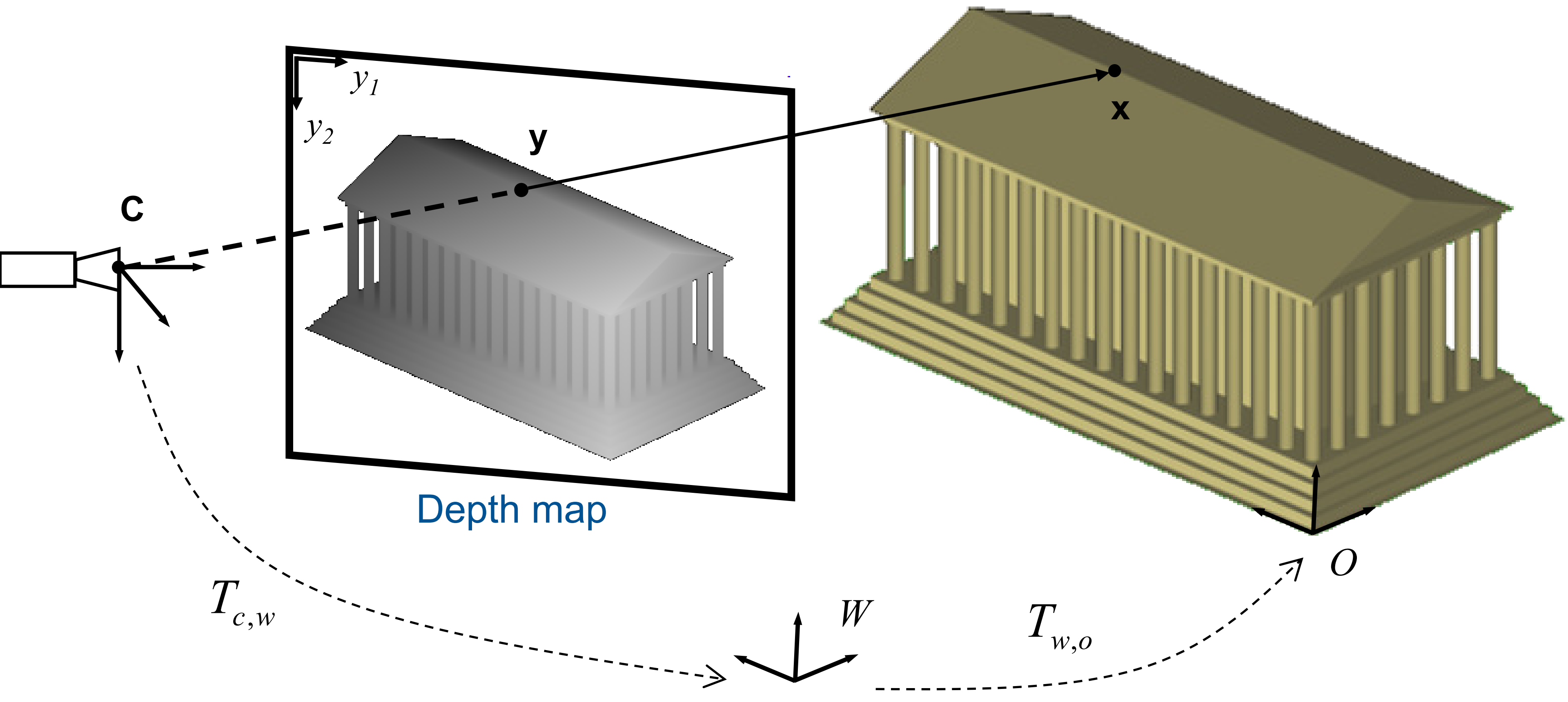
Camera model

Extrinsic parameters



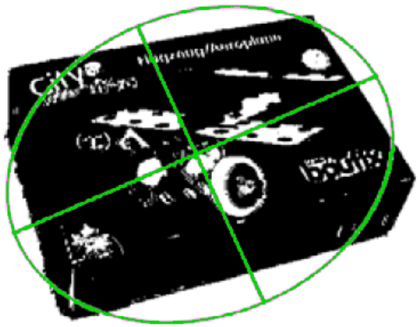
Camera model

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

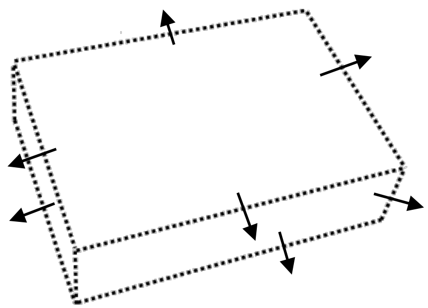


Visual modalities

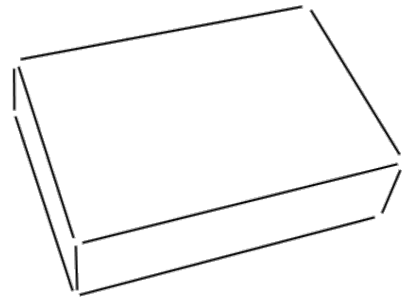
Shape moments



Intensity gradients



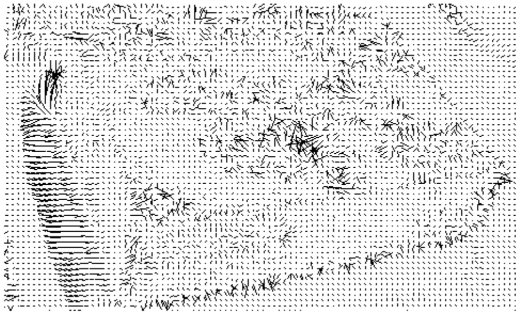
Contour lines



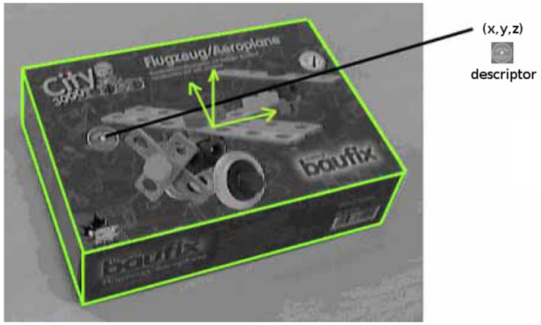
Color statistics

Texture template

Optical flow

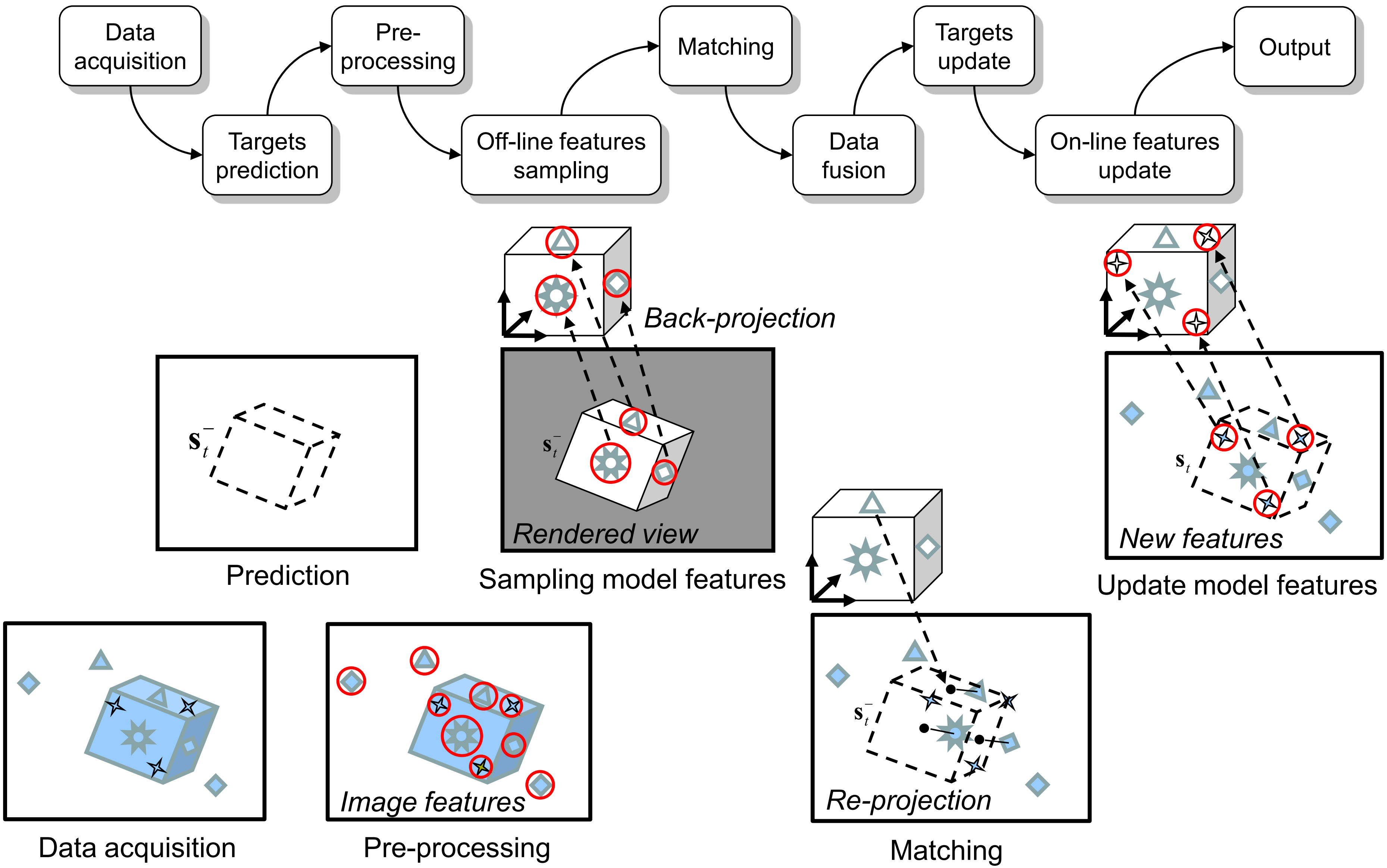


Local keypoints



(and others: Background subtraction / CCD / Harris keypoints / Histogram of oriented gradients / SIFT)

The „tracking pipeline“



Abstraction: visual modality processing

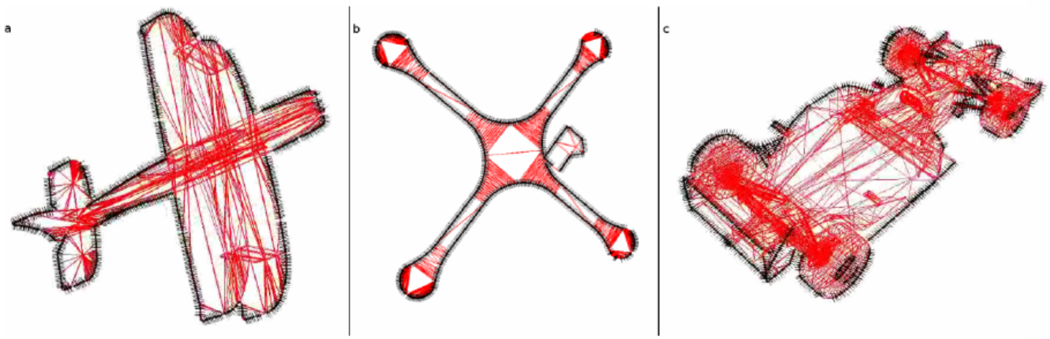
Rule: any *modality* class must implement

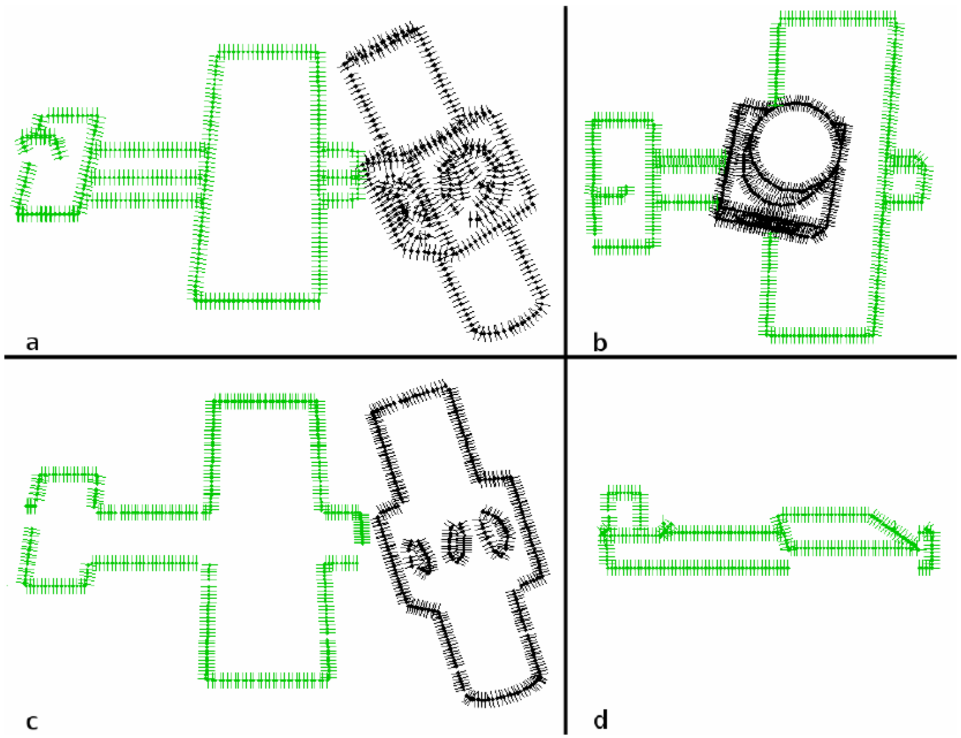
- Model free pre-processing
- Off-line and on-line features sampling and back-projection
- Multi-level data association (Pixel-, Feature-, State-space)
- Residuals, covariances and Jacobians computation

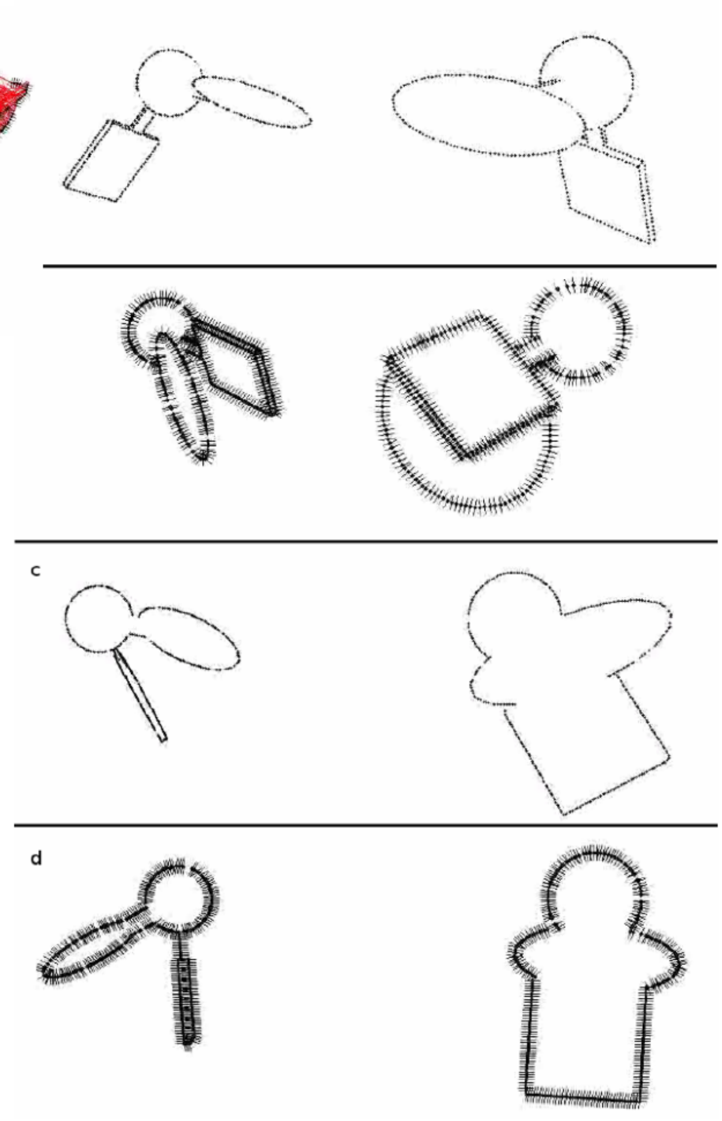
Additional classes:

- Multi-modal, multi-sensor data fusion (cascade, parallel)
- Likelihood computation

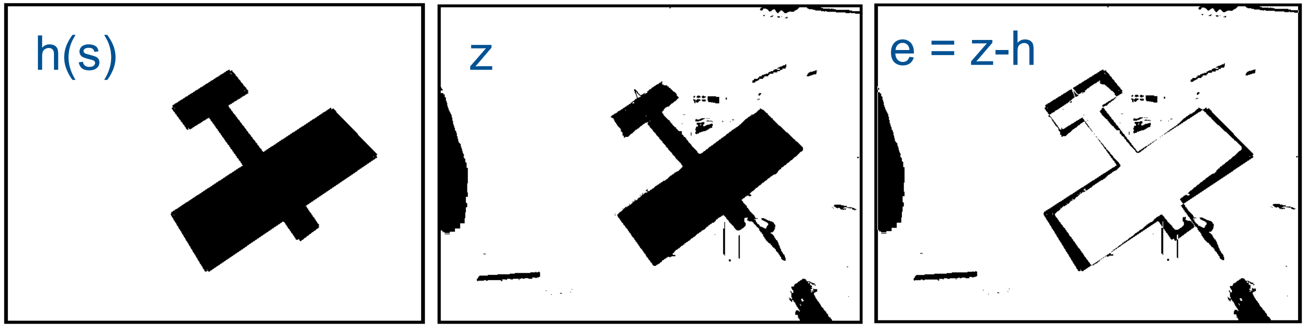
Features sampling – GPU assisted



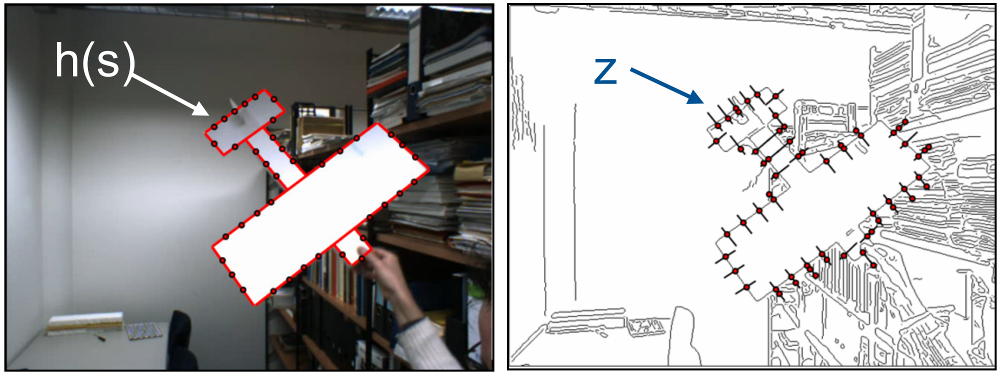




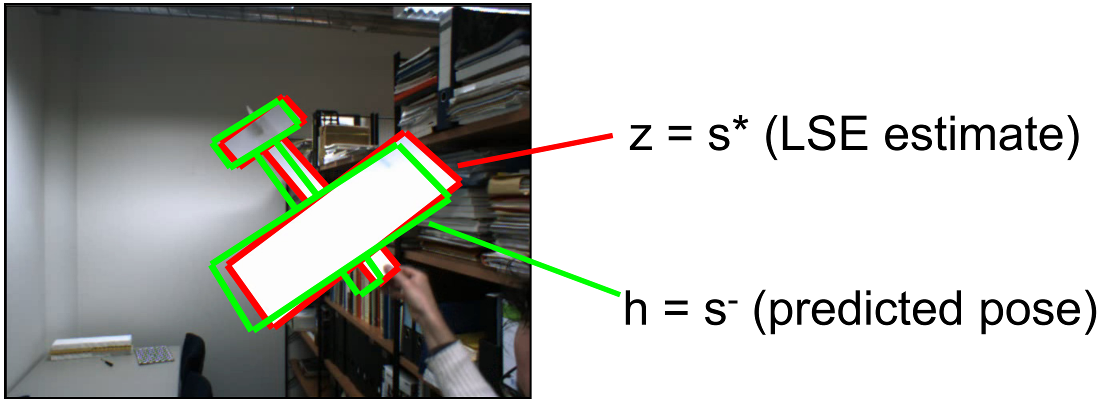
Multi-level visual processing



Pixel-level measurement



Feature-level measurement

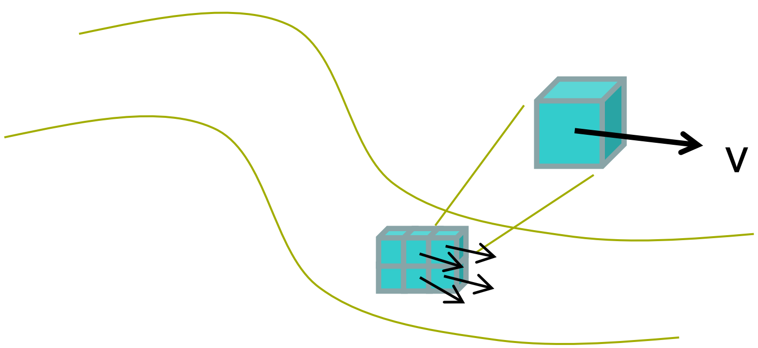


Object-level measurement

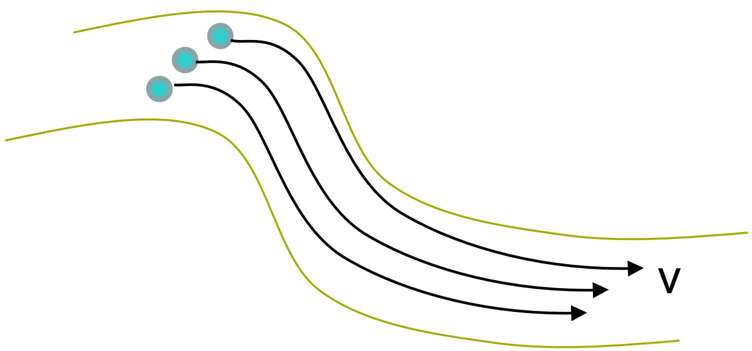
Measurement: pixel- vs. feature-level

Analogy with fluid mechanics

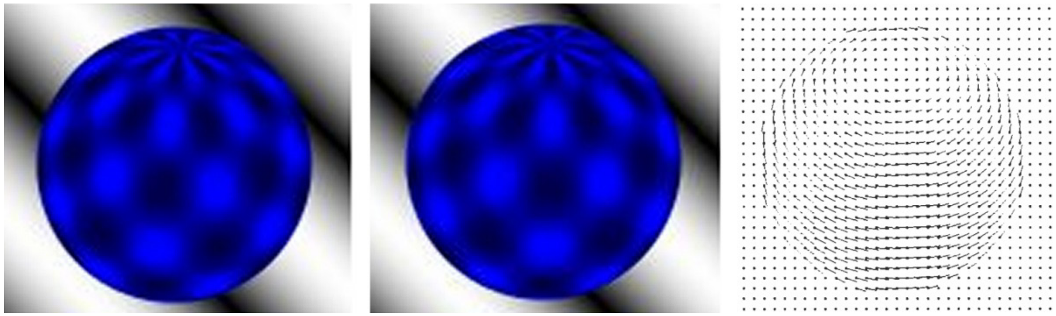
Eulerian = pixel-level



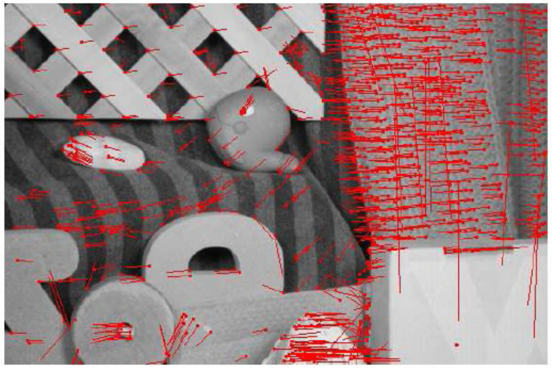
Lagrangian = feature-level



Dense optical flow (HS)



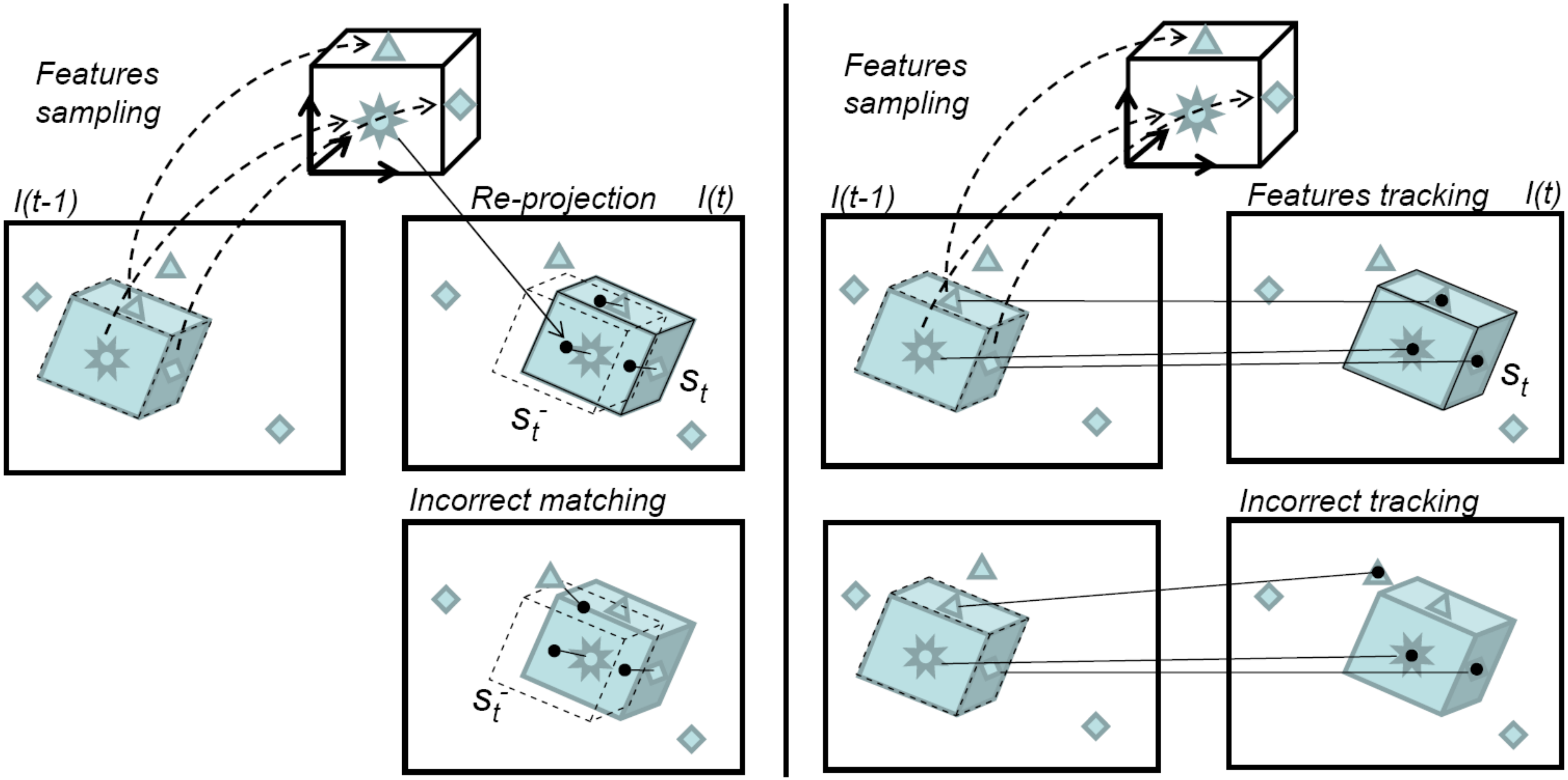
Sparse optical flow (LK)



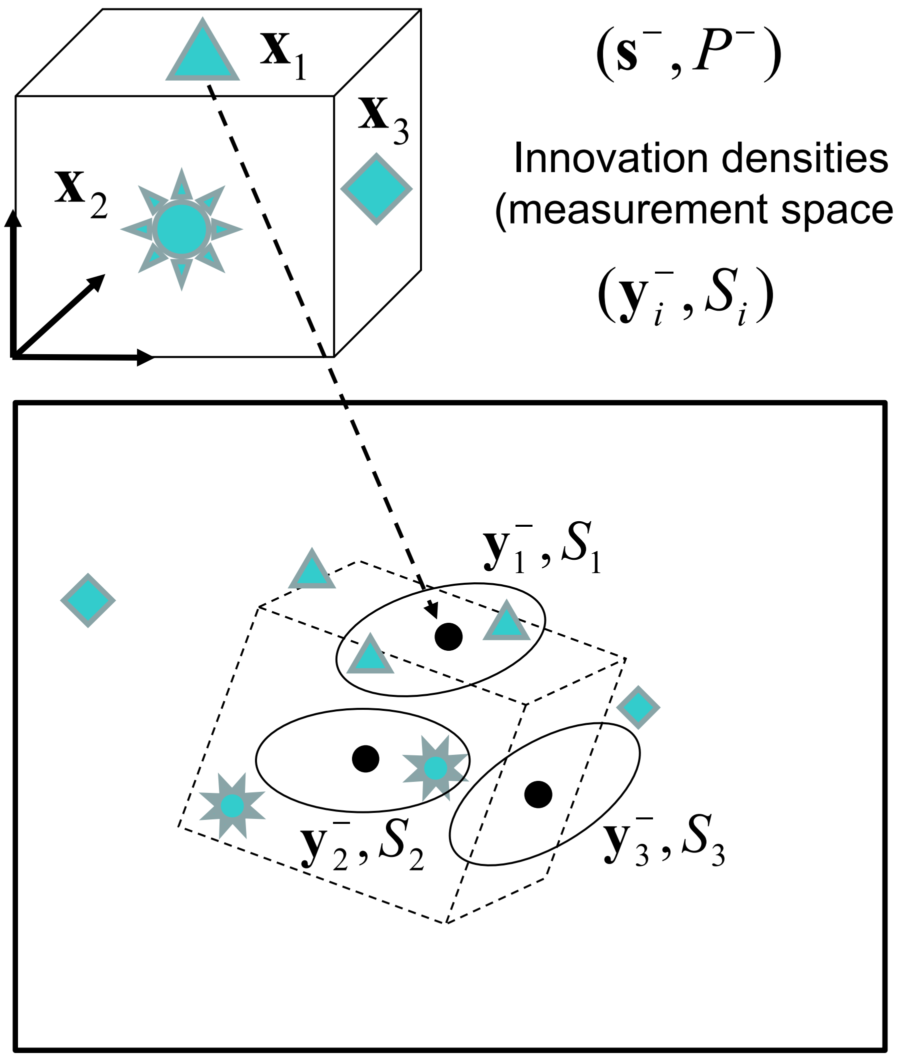
Feature-level: re-projection *vs.* tracking

Model feature re-projection

Feature tracking (= „flow“)



Feature-level: validation gates (local search)

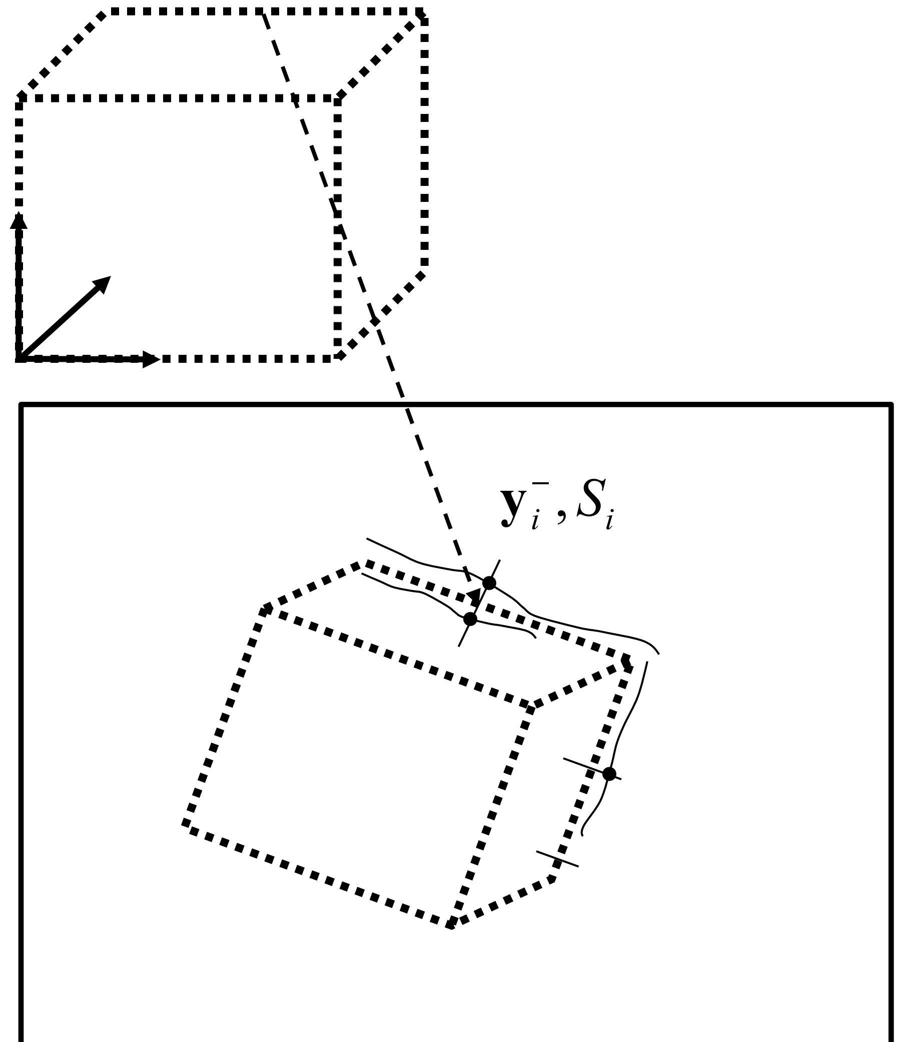


Prior density (state-space)

$$(\mathbf{s}^-, P^-)$$

Innovation densities (measurement space)

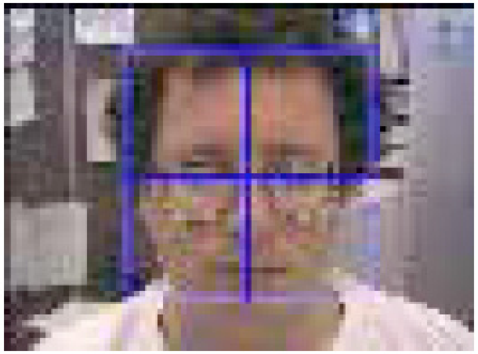
$$(\mathbf{y}_i^-, S_i)$$



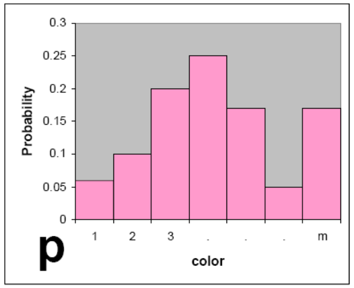
Example: color histograms

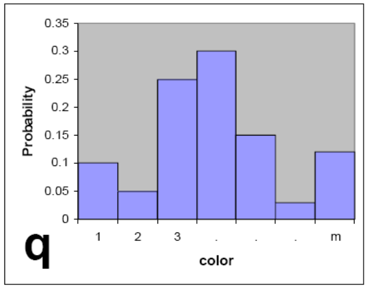
Model | Color segmentation

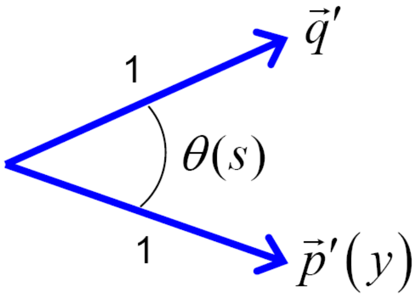
Pixel-level matching



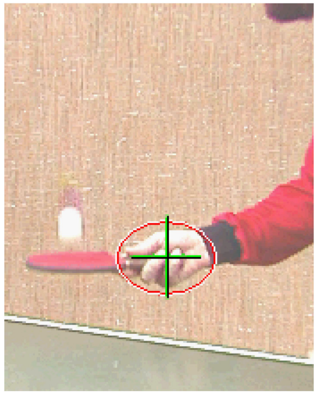
Feature-level matching
= histogram distance

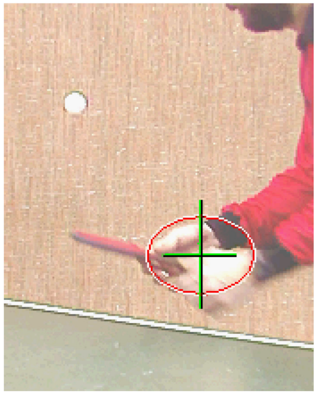


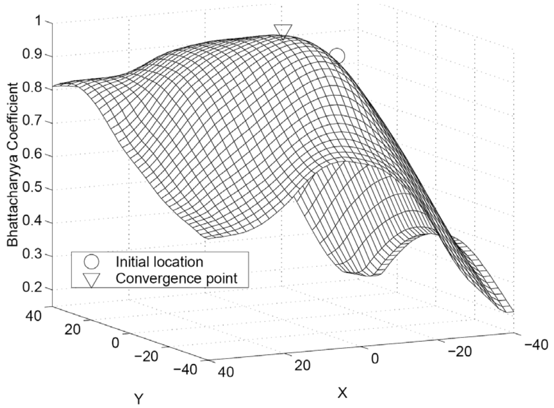




Object-level matching
= mean-shift optimization



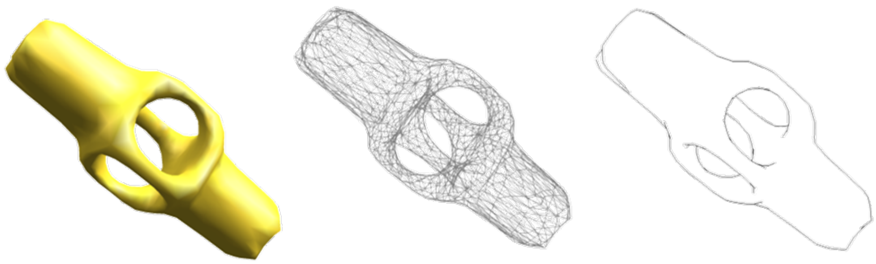




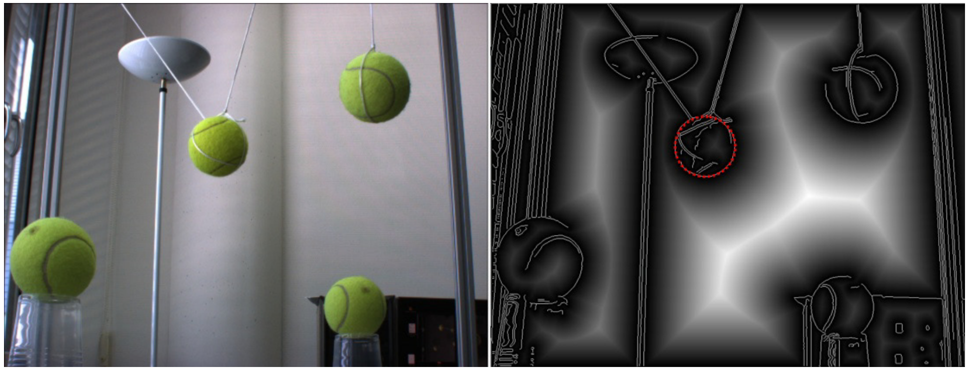
Example: intensity edges

Pixel-level

Draw the silhouette

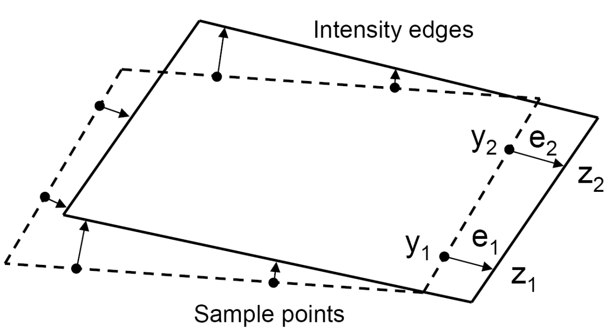


Match silhouette to the Distance Transform

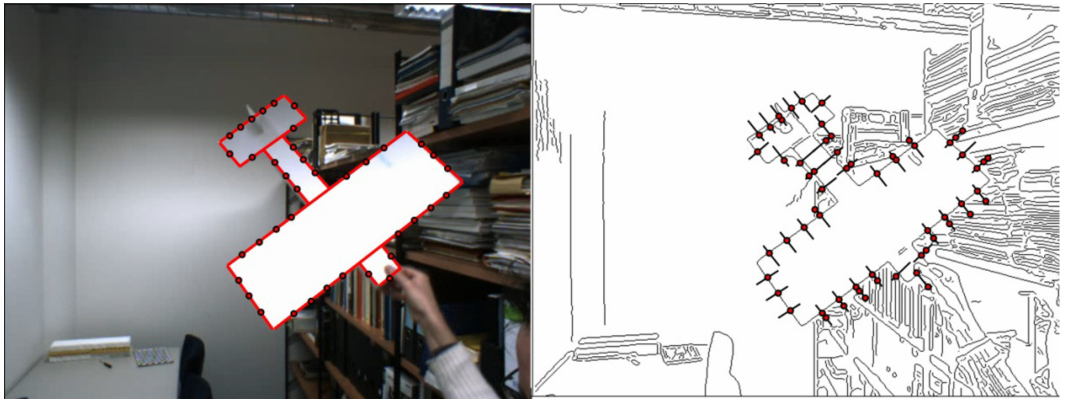


Feature-level

sample contour points

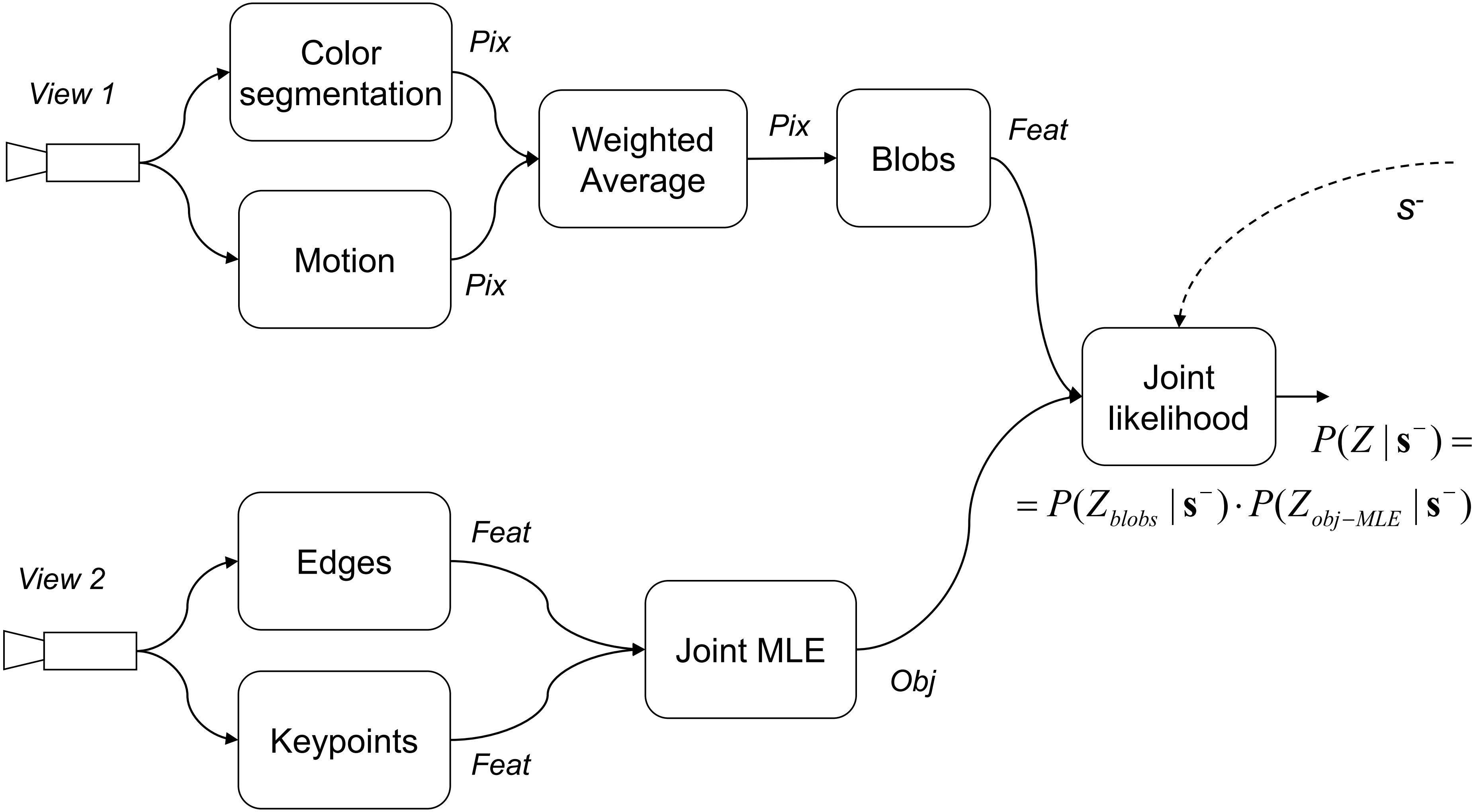


Re-project and search in the image



Building in OpenTL

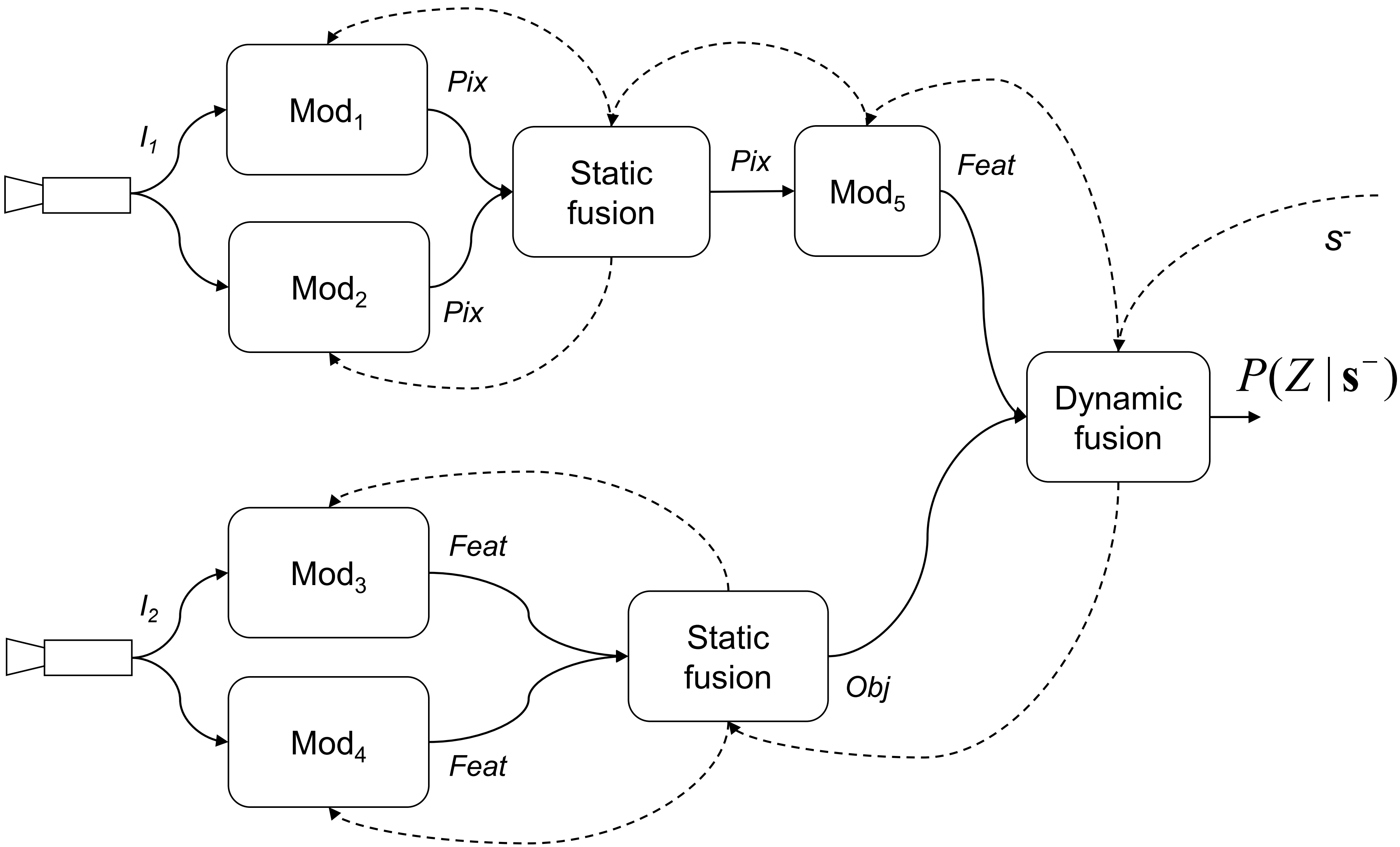
Multi-camera, multi-level data fusion



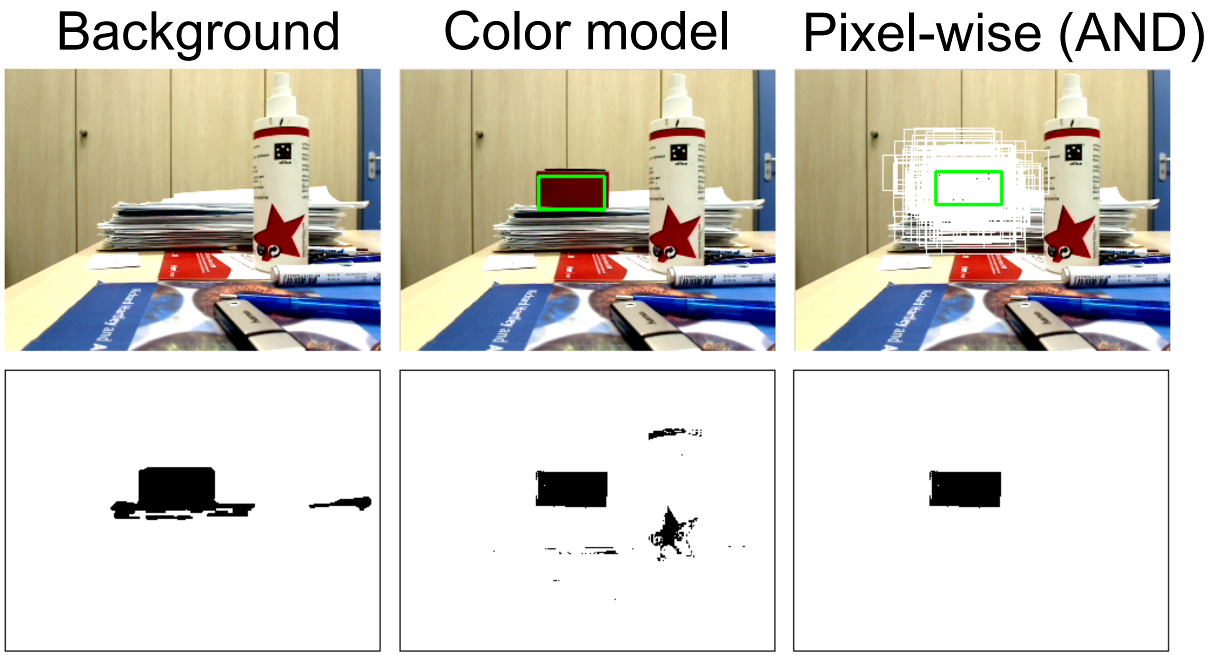
$$P(Z \mid \mathbf{s}^-) =$$
$$= P(Z_{blobs} \mid \mathbf{s}^-) \cdot P(Z_{obj-MLE} \mid \mathbf{s}^-)$$

Building processing trees in OpenTL

Generalization of the tree



Data fusion – multi-modal



Benefits:

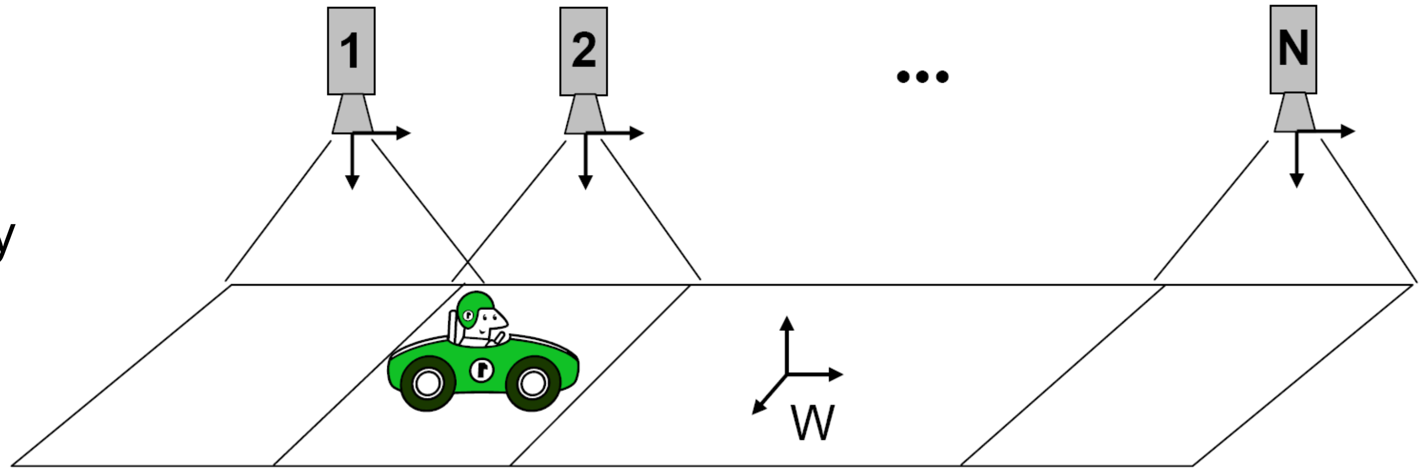
- Combine independent information sources
- Increase robustness (tracking fails if ALL modalities fail)

Drawbacks:

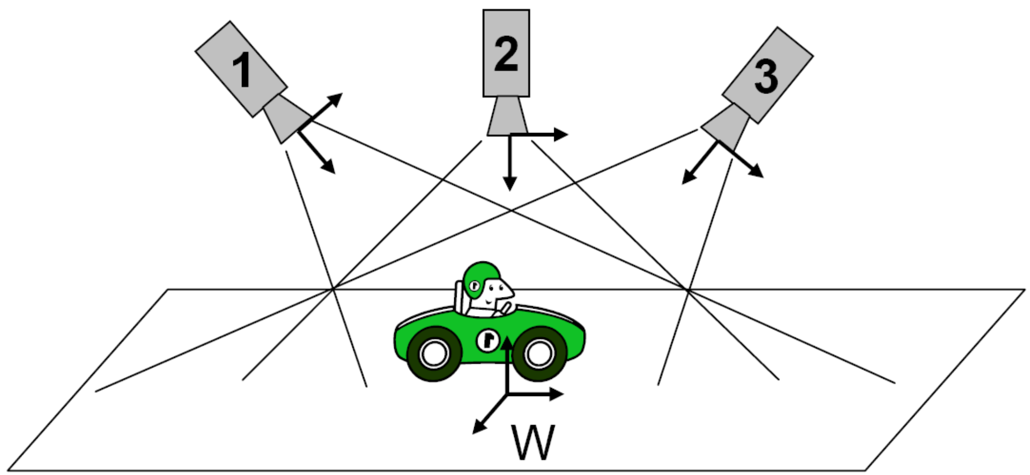
- Need to define a proper fusion scheme, and parameters
- Higher computational effort → slower frame rate

Data fusion – multi-camera

Complimentary setup

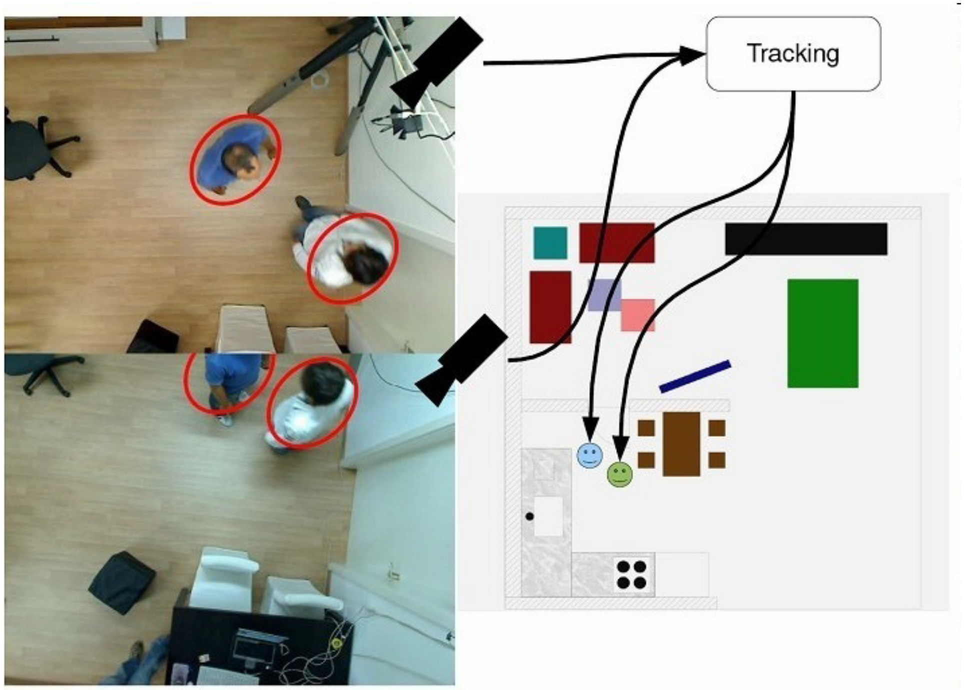


Redundant setup

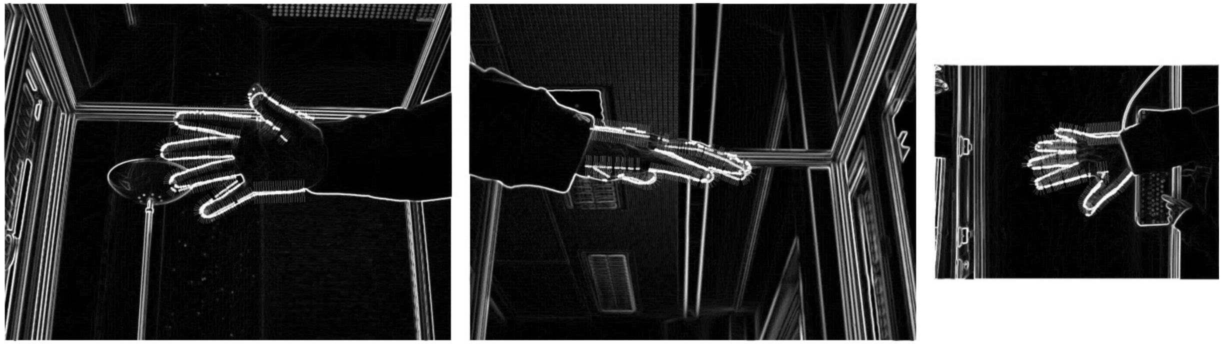


Data fusion – multi-camera

Complimentary setup:
Indoor people tracking

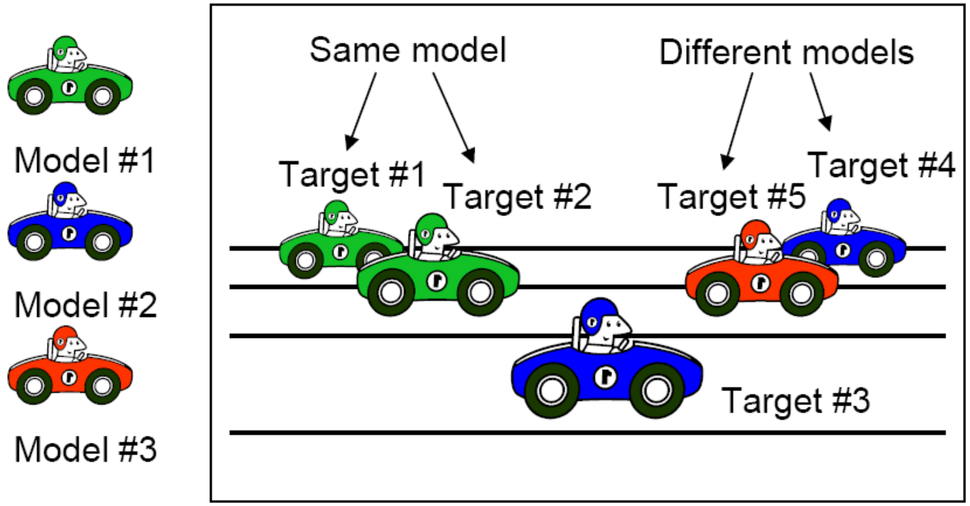


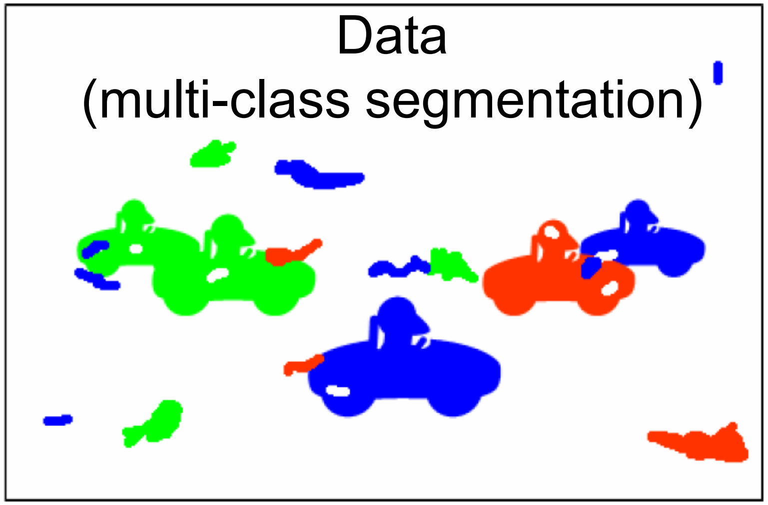
Redundant setup:
3D hand tracking



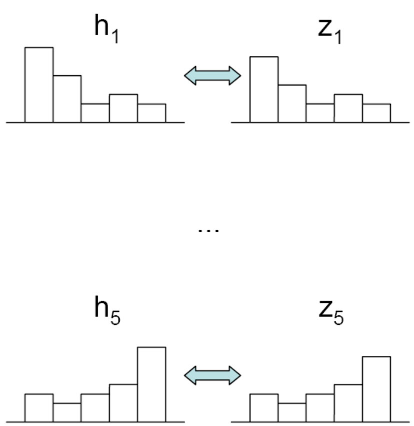
Multi-target – occlusion handling

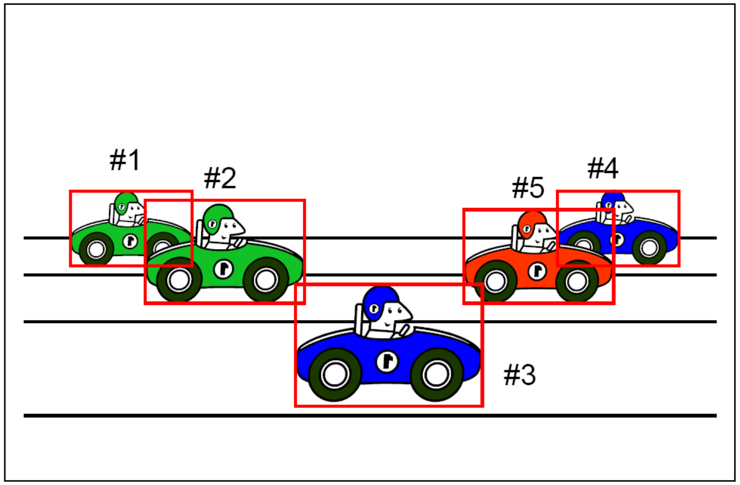
Pixel-level





Feature-level

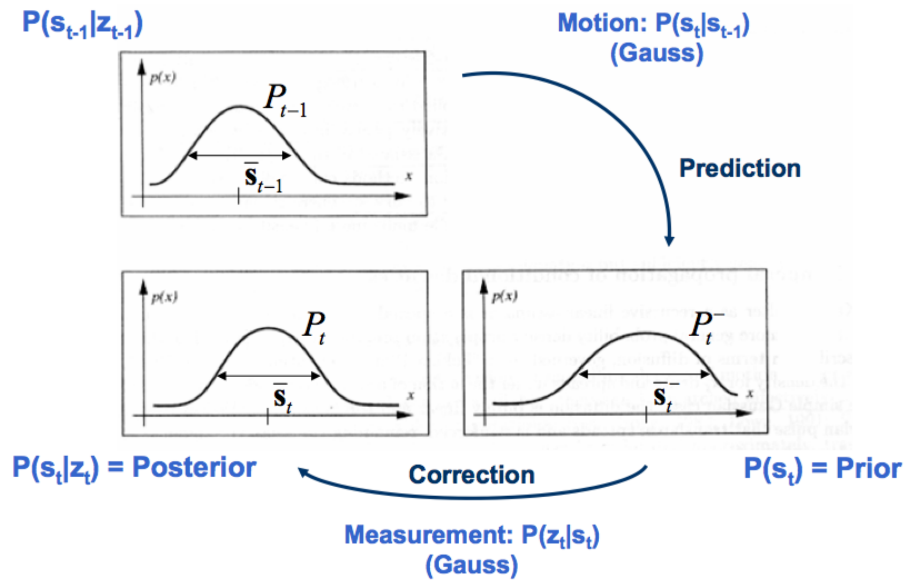




Bayesian Tracking: prediction - correction

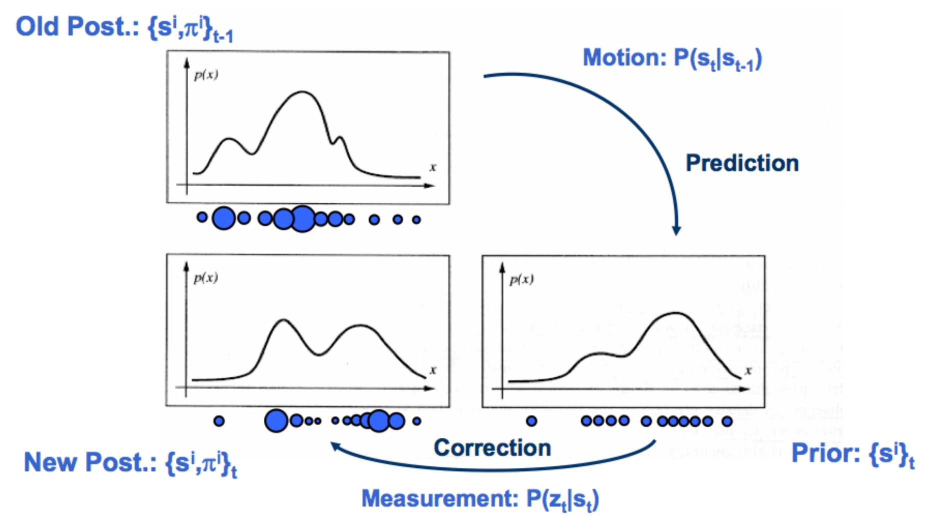
Gaussian filters

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



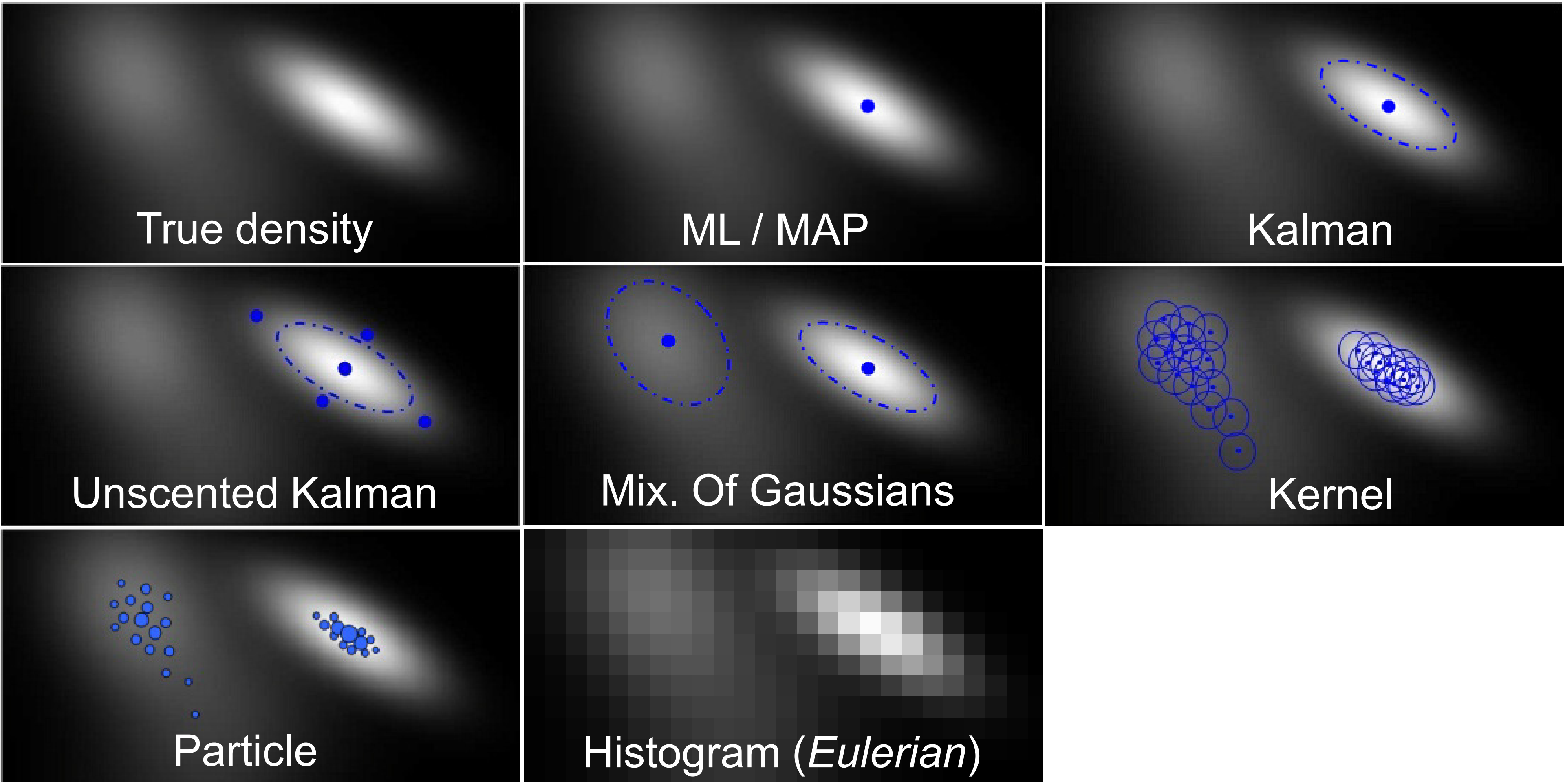
Monte-Carlo filters

- S-I-R particle filter
- MCMC particle filter

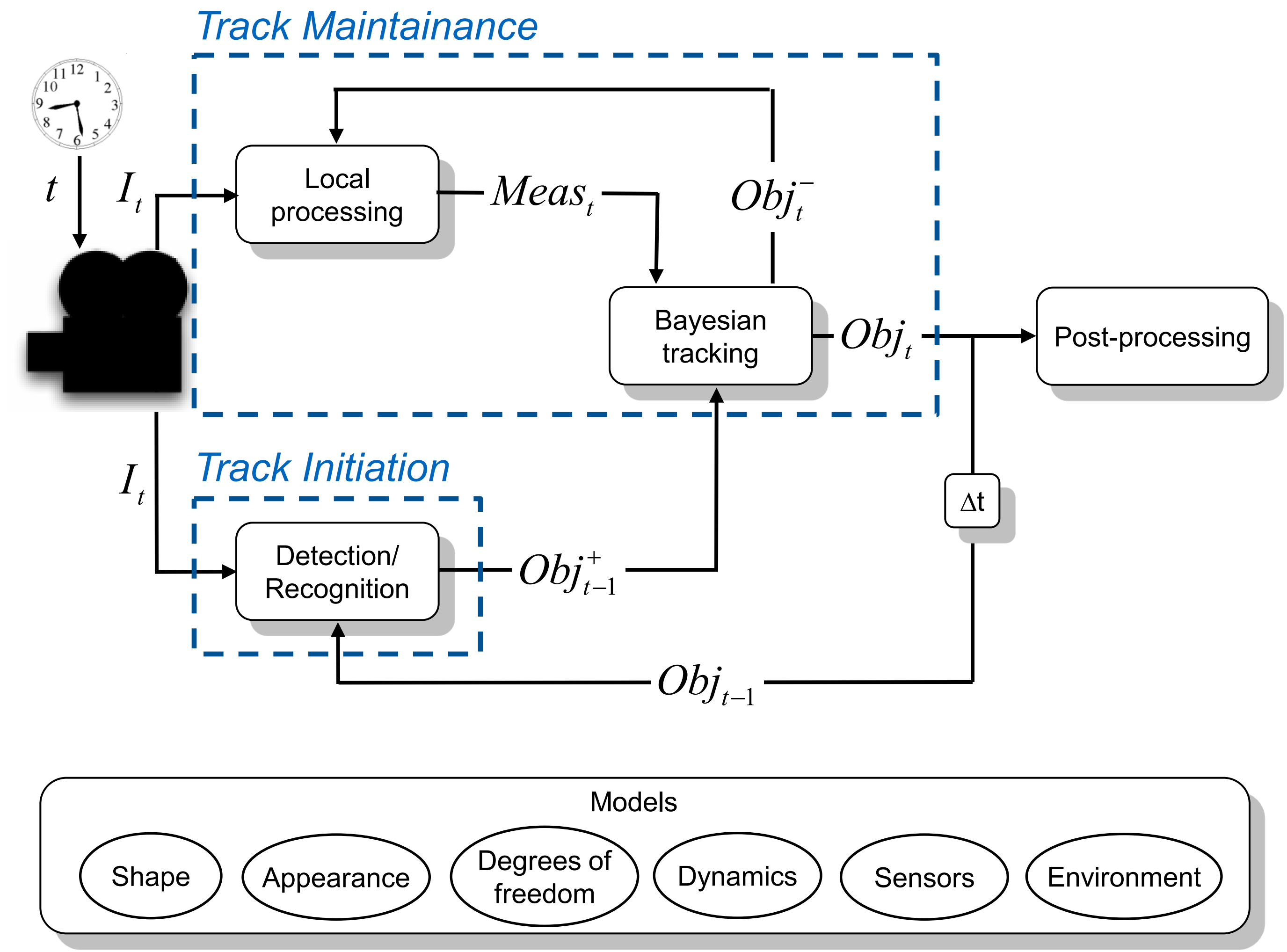


Representing the target distribution

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

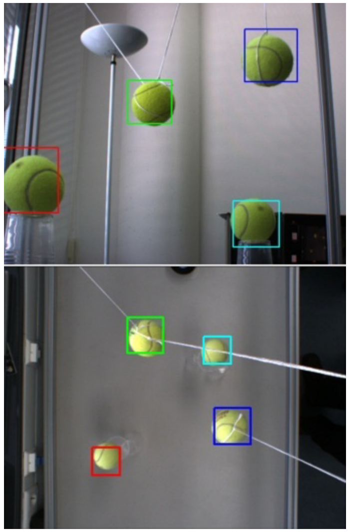


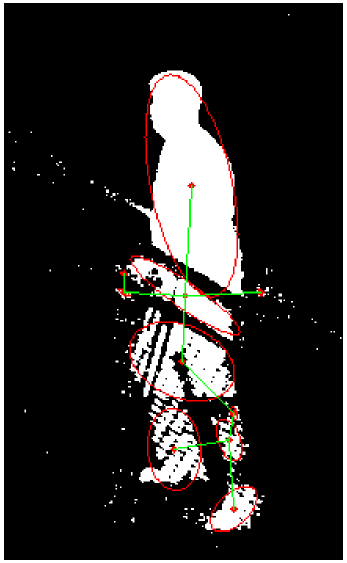
Flow diagram of OpenTL-based applications

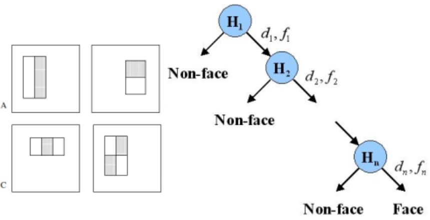


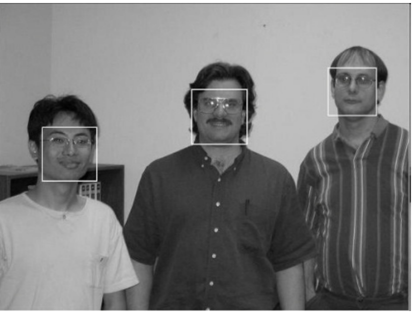
Object detection (examples)

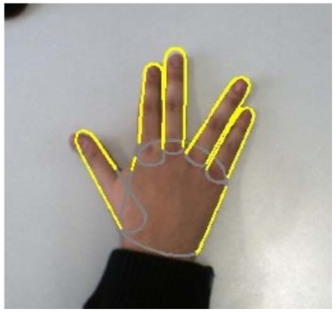
- General-purpose Monte-Carlo sampling in state-space
- People detection based on foreground blobs clustering
- Invariant keypoints matching (for textured objects)
- Marker detection based on intensity edges
- Hand detection based on color and edge lines
- Face detection based on a trained classifier (with Haar features)

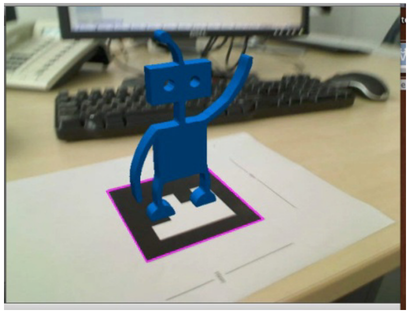


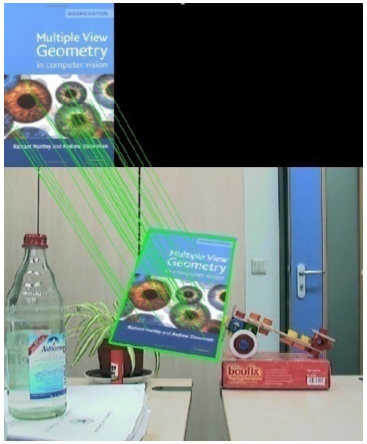




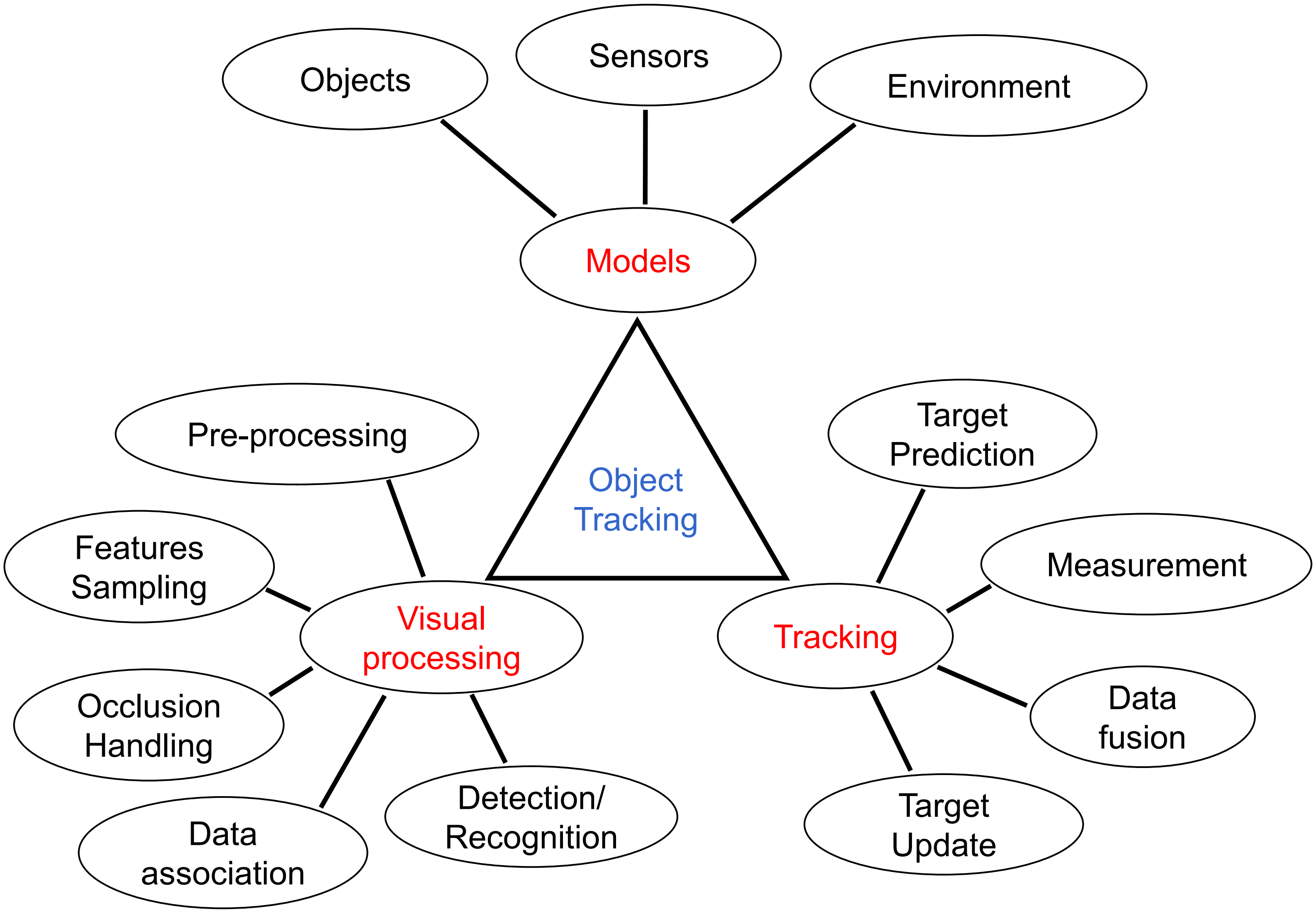




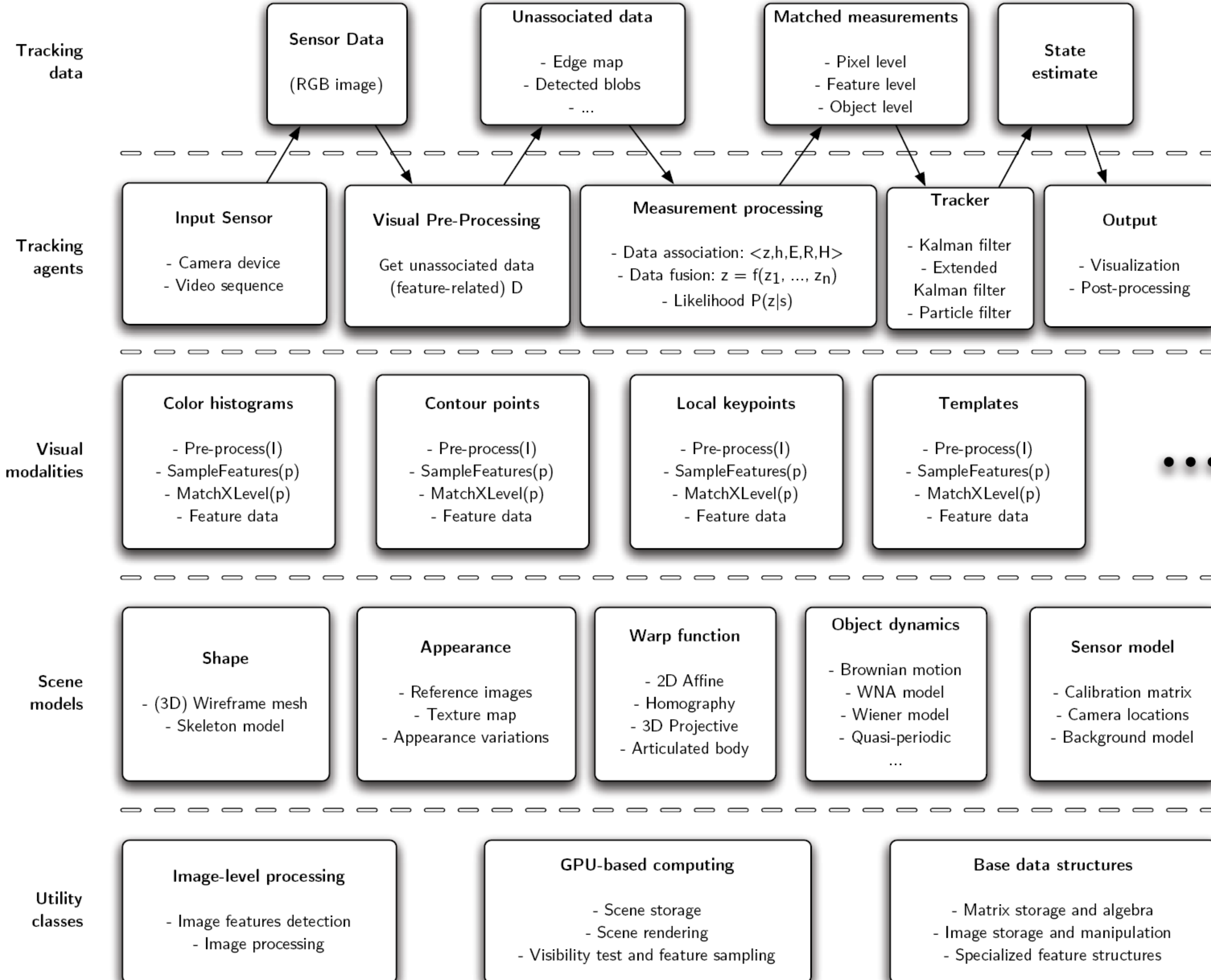




Resume: model-based object tracking



Building applications with OpenTL



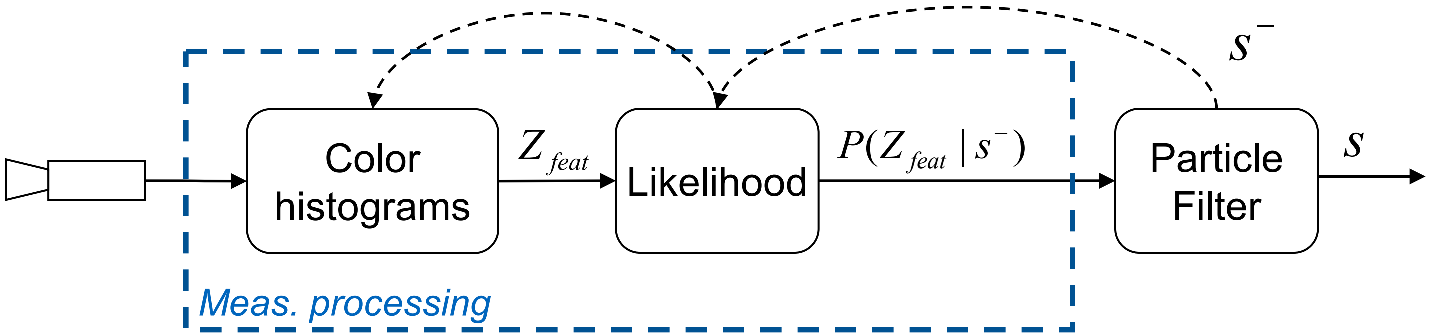
Tracking data
Sensor Data
(RGB image)
Unassociated data
- Edge map
- Detected blobs
- ...
Matched measurements
- Pixel level
- Feature level
- Object level
State estimate
Tracking agents
Input Sensor
- Camera device
- Video sequence
Visual Pre-Processing
Get unassociated data (feature-related) D
Measurement processing
- Data association: <z,h,E,R,H>
- Data fusion: $z = f(z_1, ..., z_n)$
- Likelihood $P(z|s)$
Tracker
- Kalman filter
- Extended Kalman filter
- Particle filter
Output
- Visualization
- Post-processing
Visual modalities
Color histograms
- Pre-process(I)
- SampleFeatures(p)
- MatchXLevel(p)
- Feature data
Contour points
- Pre-process(I)
- SampleFeatures(p)
- MatchXLevel(p)
- Feature data
Local keypoints
- Pre-process(I)
- SampleFeatures(p)
- MatchXLevel(p)
- Feature data
Templates
- Pre-process(I)
- SampleFeatures(p)
- MatchXLevel(p)
- Feature data
Scene models
Shape
- (3D) Wireframe mesh
- Skeleton model
Appearance
- Reference images
- Texture map
- Appearance variations
Warp function
- 2D Affine
- Homography
- 3D Projective
- Articulated body
Object dynamics
- Brownian motion
- WNA model
- Wiener model
- Quasi-periodic
...
Sensor model
- Calibration matrix
- Camera locations
- Background model
Utility classes
Image-level processing
- Image features detection
- Image processing
GPU-based computing
- Scene storage
- Scene rendering
- Visibility test and feature sampling
Base data structures
- Matrix storage and algebra
- Image storage and manipulation
- Specialized feature structures

Features of OpenTL

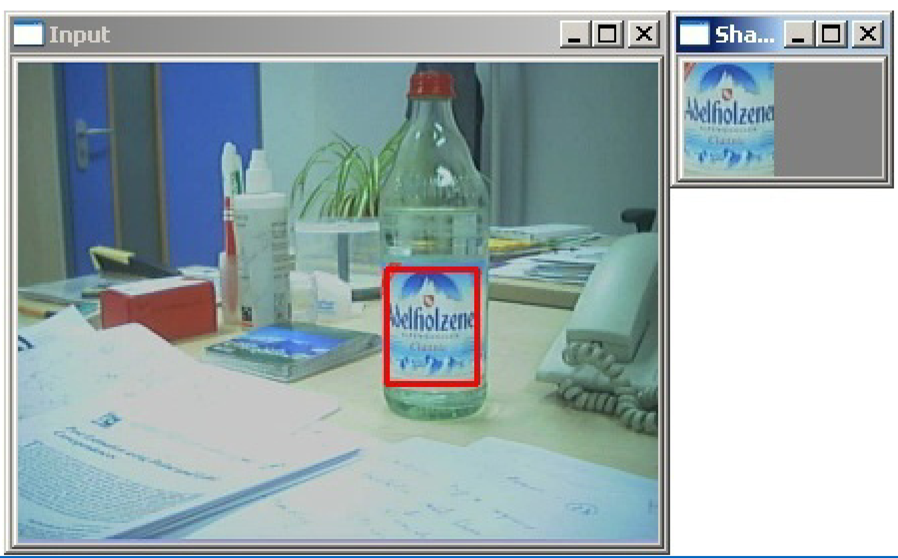
- Modularized, object-oriented software architecture
- Common abstractions for layers
- Real-time performance
- Different Bayesian filters
- A large variety of visual modalities, with multiple processing levels
- Robust improvements (multi-hypotheses, data fusion, …)
- Generic sensor abstraction
- Support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

Application: planar object tracking

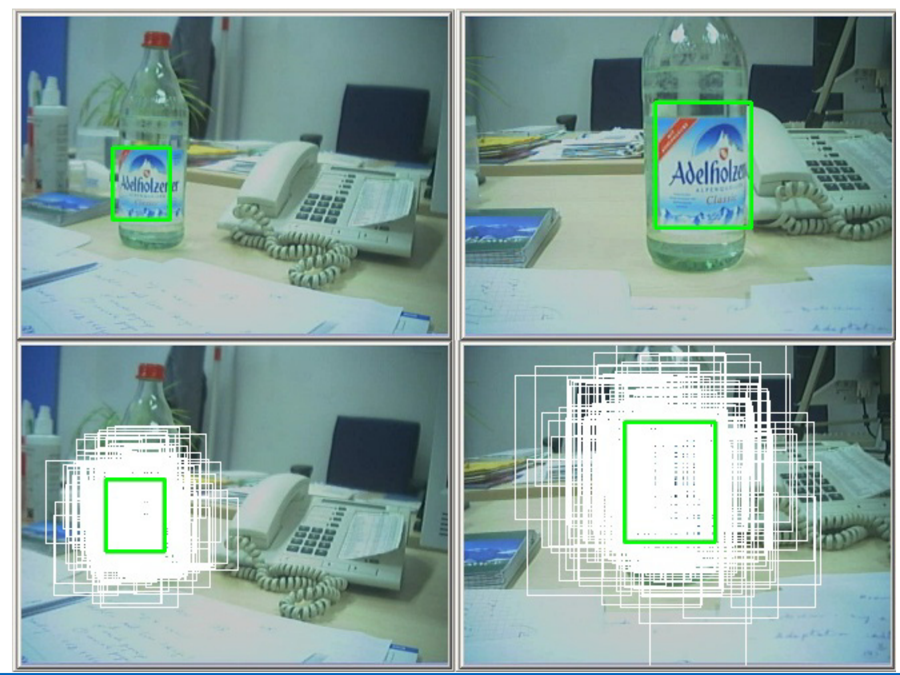
Single-target, single-camera, color-based

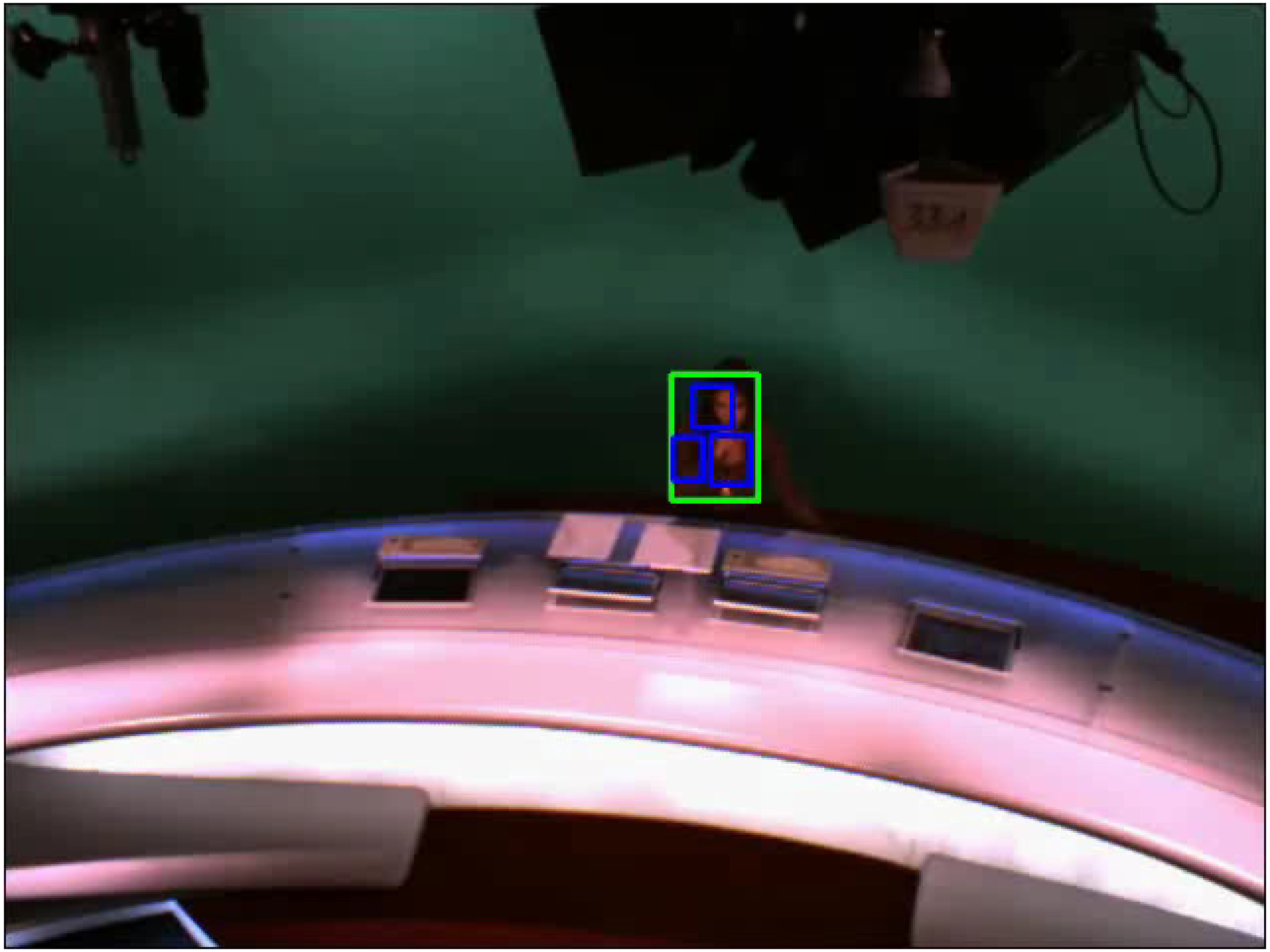


Modeling



Tracking

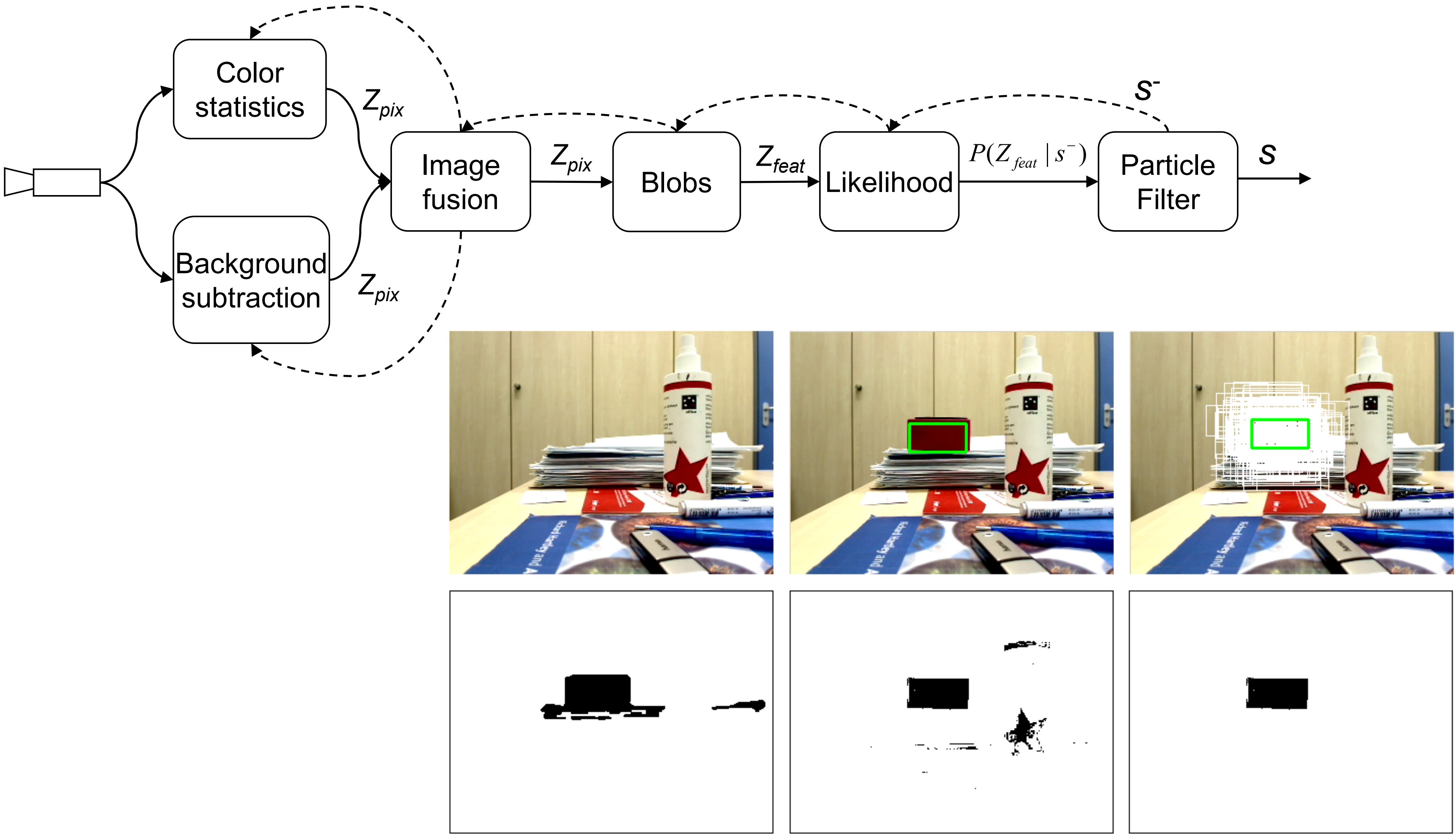





promod

Application: planar object tracking

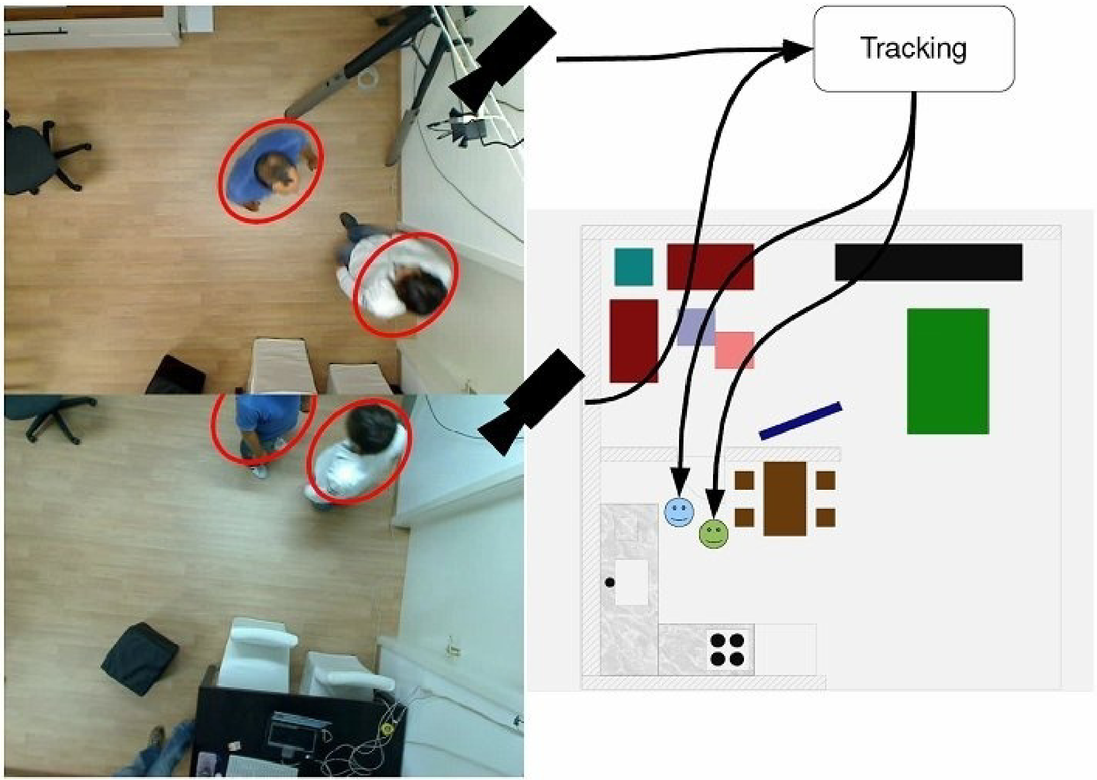
Fusion color+background (pixel-level), and blob detection



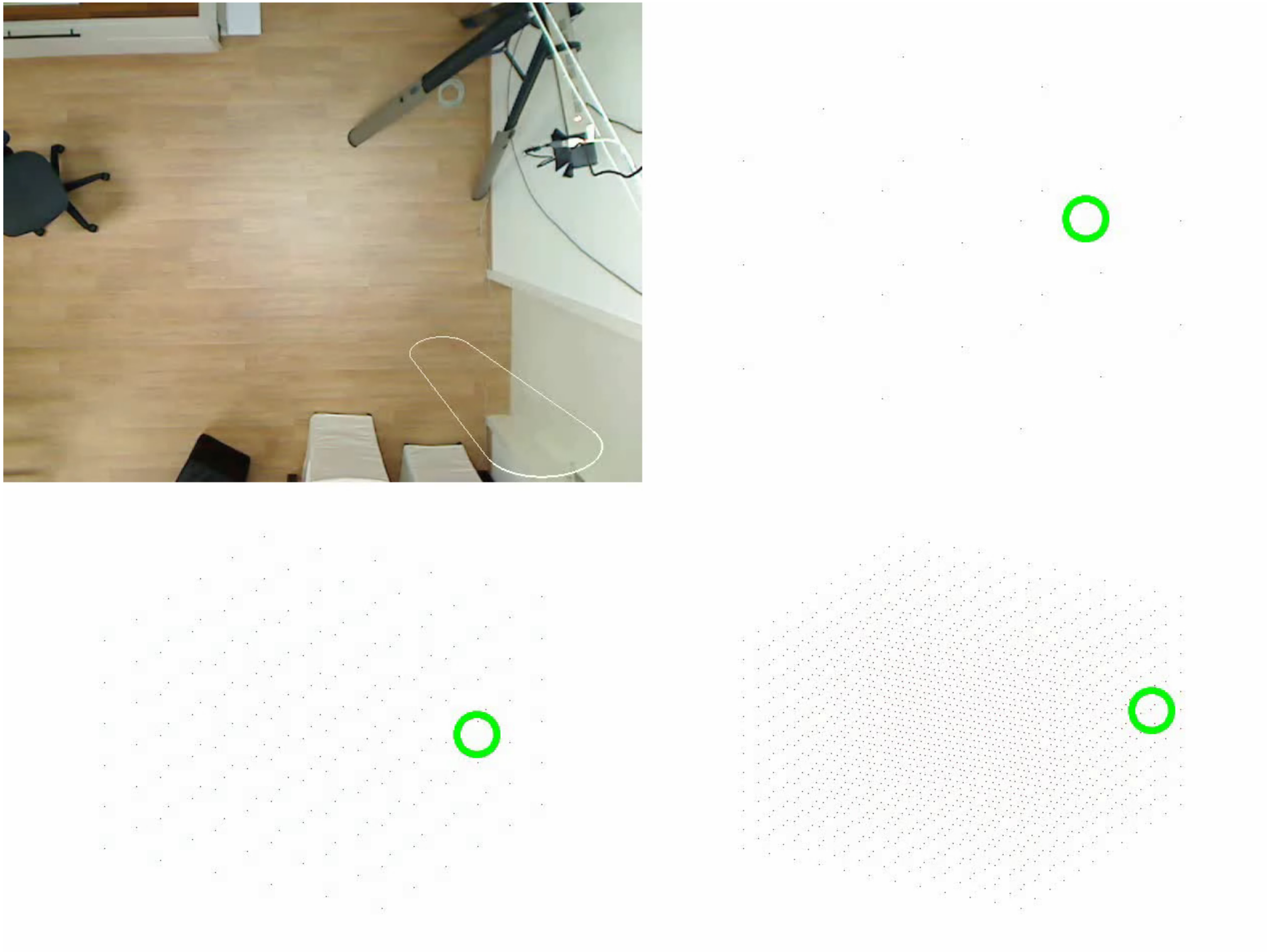
People tracking with distributed cameras

CoTeSys – ITrackU

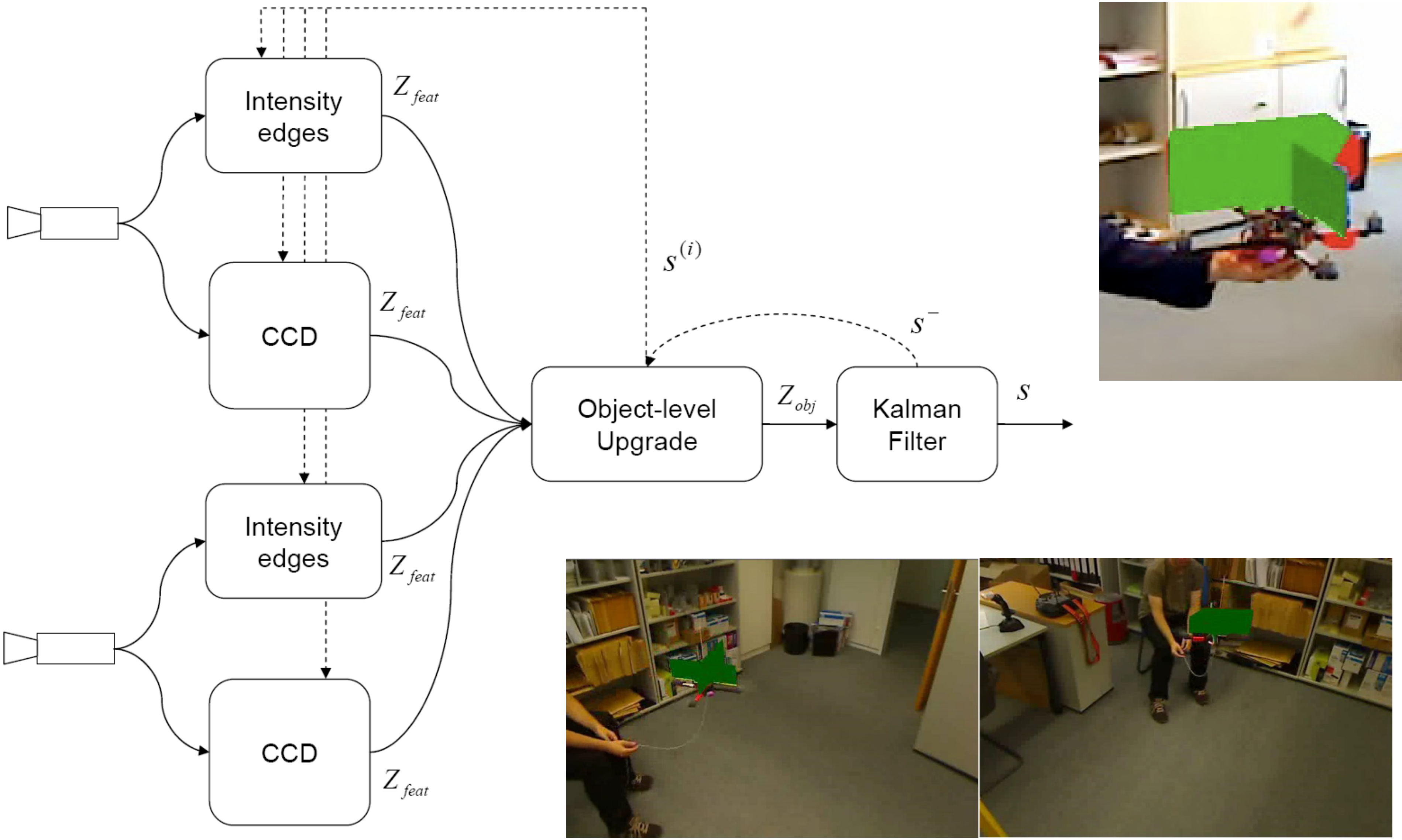
- Distributed, multi-camera setup
- People detection and tracking in real-time
- 3D localization
- Goal:
  Human-Robot Interaction (*coffee-break* scenario)



Hierarchical grid-based localization

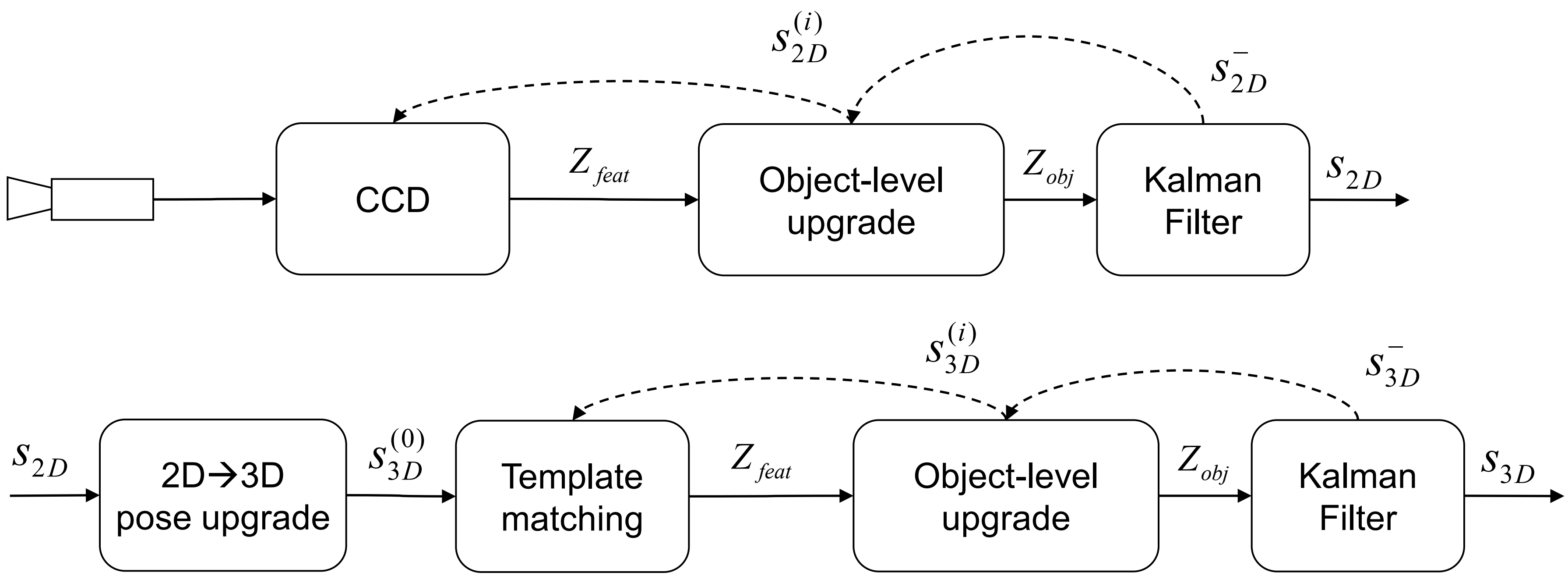


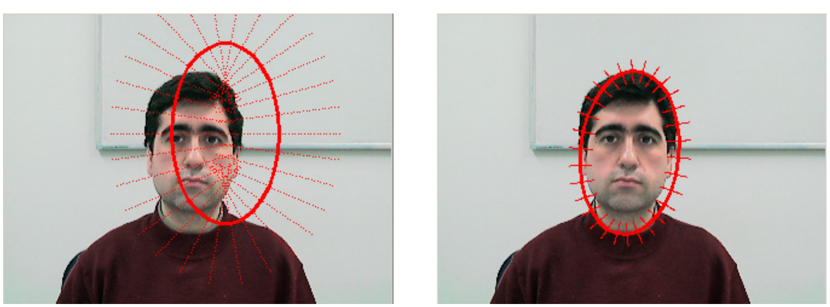
Stereo tracking of a quadcopter



Application: face tracking

Cascade of two pipelines, with 3D pose upgrade

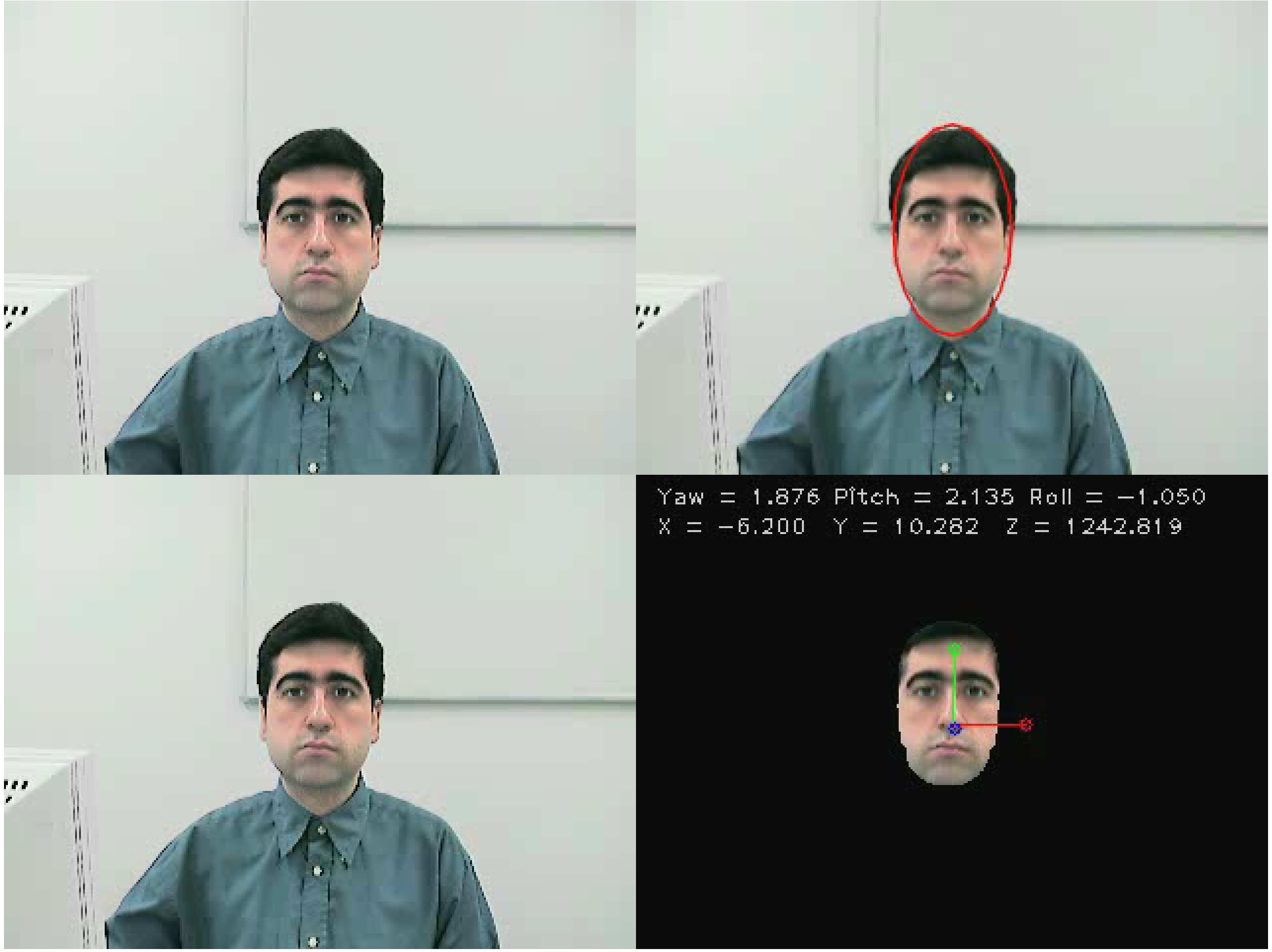




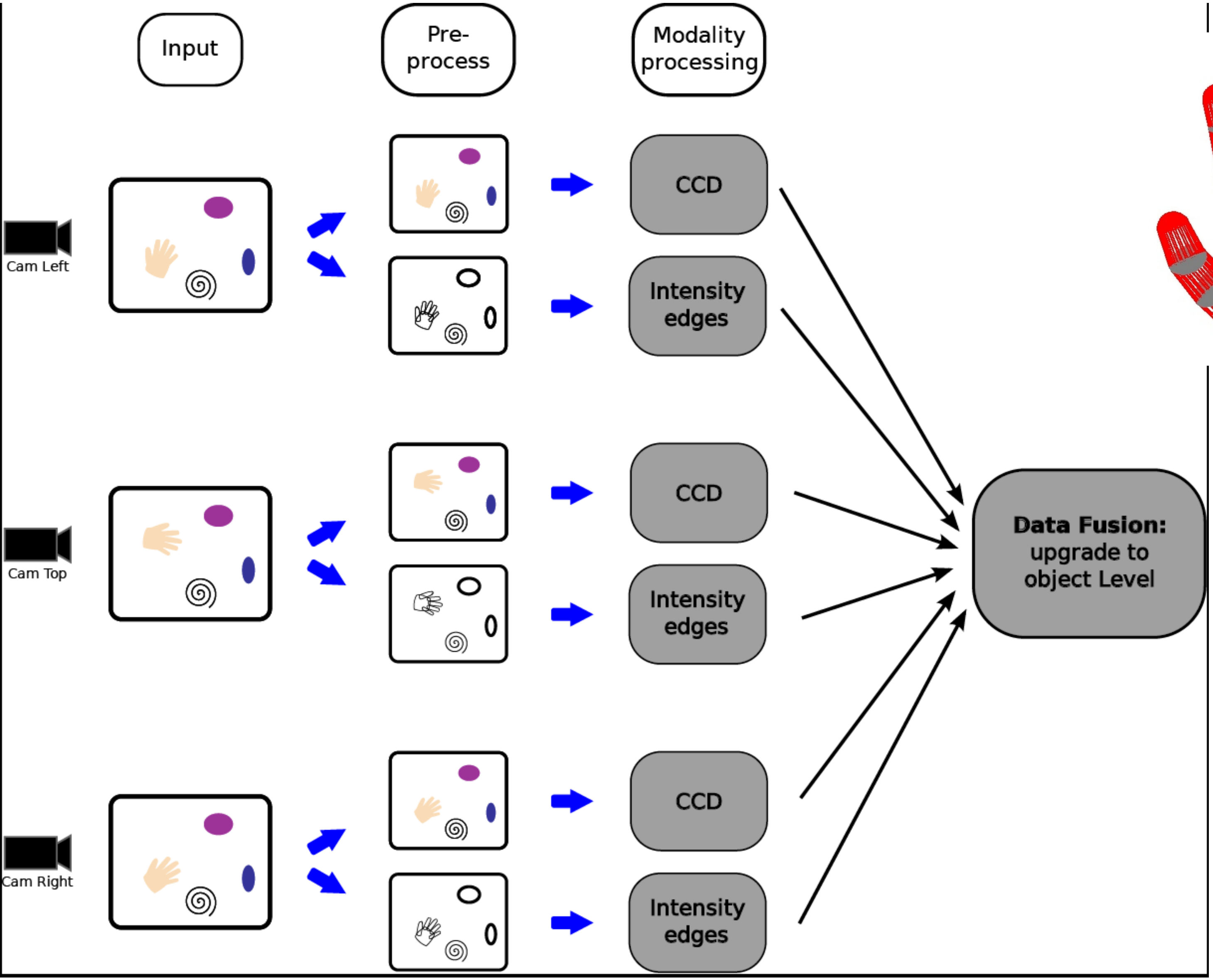
2D tracking

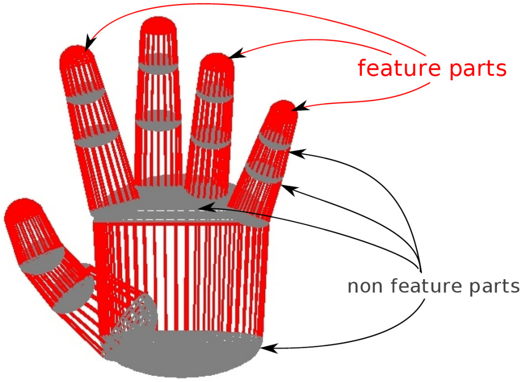
3D model

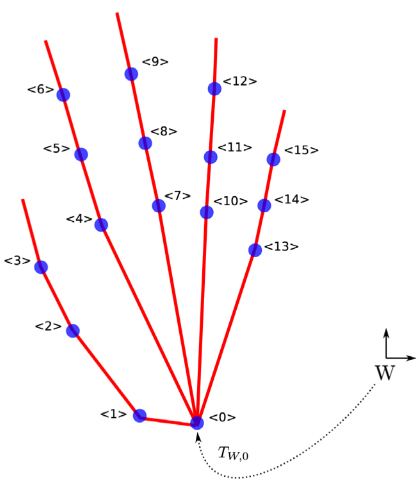
3D tracking

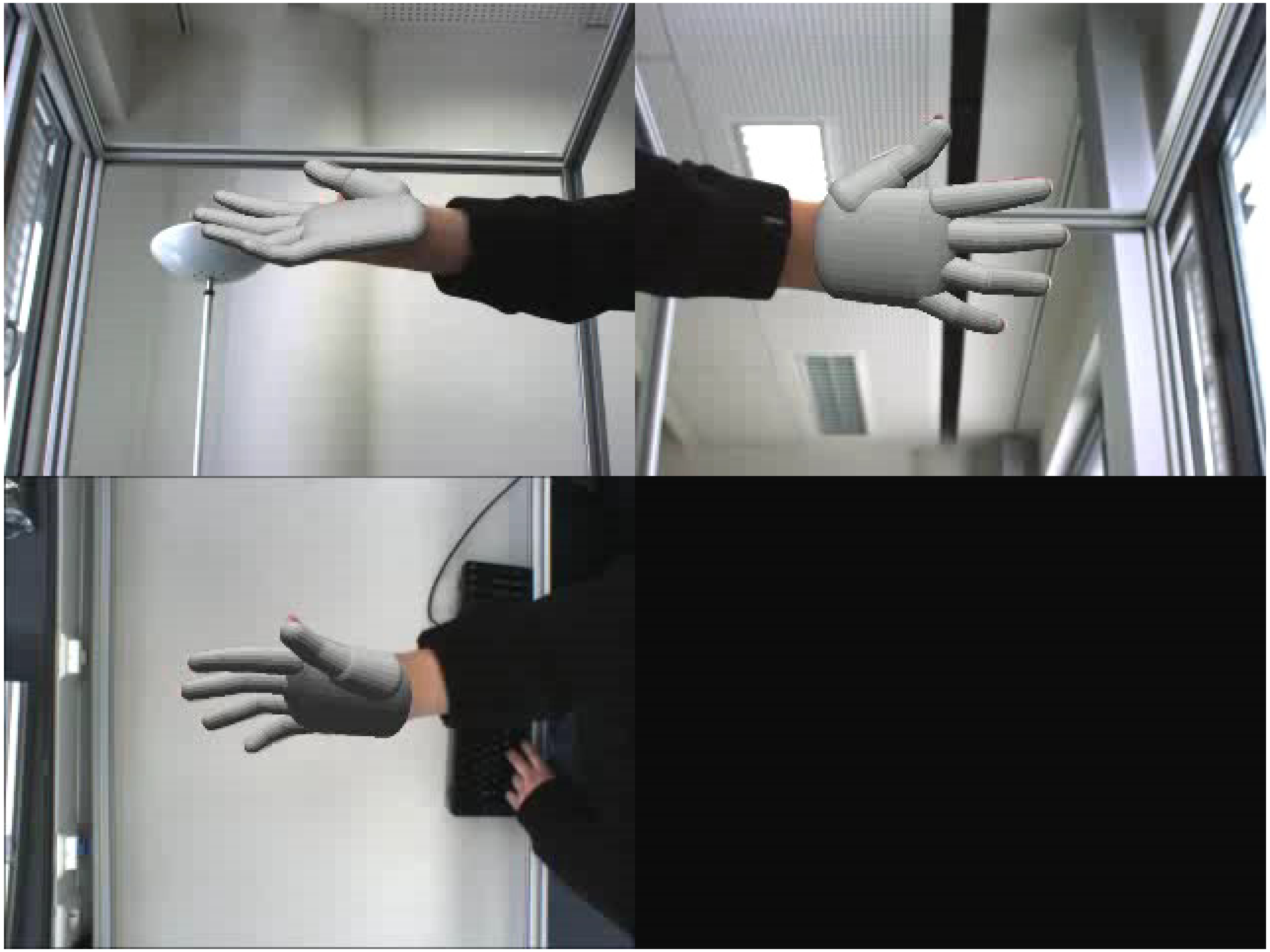

Yaw = 1.876 Pitch = 2.135 Roll = -1.050
X = -6.200 Y = 10.282 Z = 1242.819

Application: 3D hand tracking





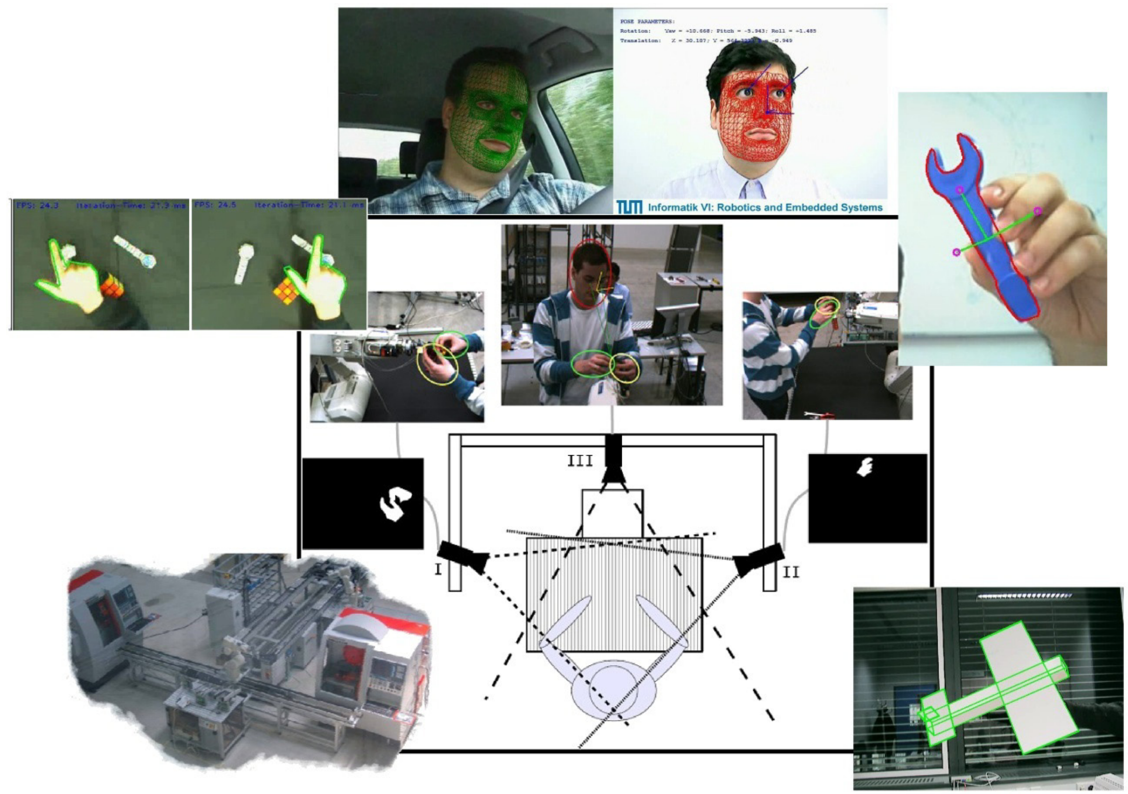




Our Tracking Projects

www.opentl.org

- 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