Chapter Outline

# Target Recognition, Tracking, and Optical Flow Estimation

**Chapter 8 Multimodal Perception and Intelligent Decision-Making**

8.1 Image Processing Fundamentals

**8.2 Recognition, Tracking, Optical Flow**

8.3 Localization, Mapping, and Planning

8.4 Common Perception Sensors

8.5 Coordinate Transformations

8.6 Fusion and Control Decision-Making

8.7 Tools and Framework Interfaces

8.8 Experiments and Real-World Transfer

8.9 Exercises and Classroom Discussion

## 5 Units in This Section

**8.2.1 Object Detection and Instance Segmentation**
Textbook: Sec. 8.1.2.2

**8.2.2 Online Object Tracking and Association**
Textbook: Sec. 8.1.2.2

**8.2.3 Multi-View Perception and Pose Estimation**
Textbook: Sec. 8.1.2.2

**8.2.4 Tracking Modes and Visual Servoing**
Textbook: Sec. 8.1.2.2

**8.2.5 Optical Flow Estimation and Motion Analysis**
Textbook: Sec. 8.1.2.3

### 8.2.1 Object Detection and Instance Segmentation: Output Types

# From Classes and Boundaries to Identity and Pose

| Task | Output | Question Answered |
|---|---|---|
| **Classification** | Class | What kind of object is it? |
| **Detection** | Class and 2D box | Where is it in the image? |
| **Instance segmentation** | Per-object pixel mask | Which pixels are its own? |
| **Tracking** | ID, trajectory, state | Is it still the same object? |
| **Pose estimation** | Joints or rigid-body pose | What configuration or orientation? |

**Persistent identity needs temporal association, not a single-frame class.**

### 8.2.1 Object Detection and Instance Segmentation: Convolutional Neural Networks

## Convolutions Extract Features; Detection Heads Locate Objects




- **Convolutional neural network (CNN):** shared kernels extract local patterns; multiple layers build hierarchical features.
- **VGG (Visual Geometry Group):** stacks small convolutional kernels; a classification or detection head uses the backbone features.
- **Residual network (ResNet):** skip connections preserve the input so deep blocks can learn a residual.

$$\mathbf{y} = \mathbf{x} + F(\mathbf{x})$$

### 8.2.1 Object Detection and Instance Segmentation: Region-Based and Single-Stage Detection

# Two-Stage Proposals and Single-Stage YOLO

| Model | Proposal / Detection Path | Features and Results |
|---|---|---|
| **Region-based CNN (R-CNN)** | External region proposals | Extract features for each region |
| **Fast R-CNN** | External region proposals | Share whole-image features; classify and refine boxes |
| **Faster R-CNN** | Region proposal network (RPN) | Share the backbone for proposals and detection |
| **YOLO** | Single-stage detection in one whole-image forward pass | Predict classes and boxes directly; then filter outputs |

YOLO: You Only Look Once. Boxes/classes feed tracking; one frame has no persistent ID.

### 8.2.1 Object Detection and Instance Segmentation: Self-Attention and Set Prediction

# Patches Gather Context; Queries Predict Object Sets

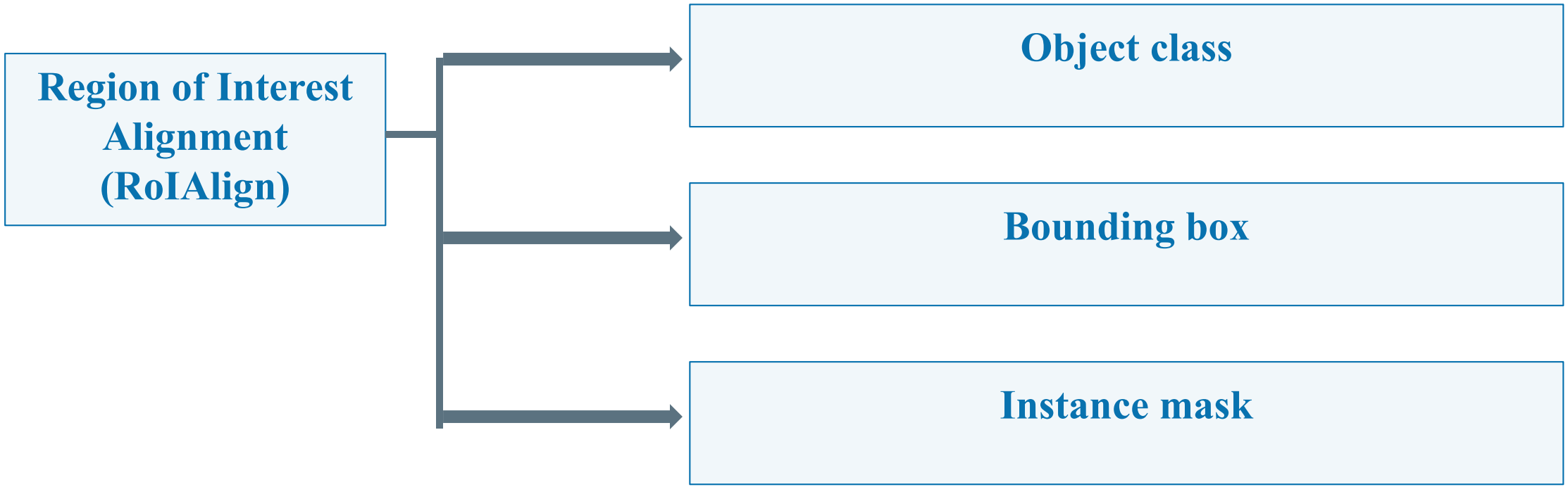
**Vision Transformer (ViT):** image patches form a sequence that gathers global context.
**Detection Transformer (DETR):** object queries directly predict a set of objects.




DETR uses one-to-one set matching during training; extra queries predict “no object.”

### 8.2.1 Object Detection and Instance Segmentation: Mask Branches

# Mask R-CNN Adds a Mask Branch for Each Object




Mask Scoring R-CNN estimates mask quality; class confidence and boundary quality are different measures.

### 8.2.1 Object Detection and Instance Segmentation: Prototype Masks

# YOLACT Shares Prototypes to Build Object Masks

**Shared prototypes** → **Combine object coefficients** → **Crop to the box**

$$M_i = \mathrm{crop}_{B_i}\left(\sigma\left(\sum_{j=1}^{k} c_{ij} P_j\right)\right)$$

Symbols: $M_i$: object mask; $P_j$: shared prototype; $c_{ij}$: combination coefficient.
$B_i$: object box; $\sigma$: maps to 0–1; $k$: number of prototypes.

- YOLACT++ adds improvements such as deformable convolutions. Sharing reduces repeated work; resolution still limits boundaries.

### 8.2.1 Object Detection and Instance Segmentation: Centers and Interframe Offsets

# Current Centers Point Back to Previous Objects

CenterTrack combines center-based detection with interframe association; MOT means multi-object tracking.

**Two frames + prior heatmap** → **Centers + boxes + offsets** → **Point back; match greedily**

$$\boldsymbol{\delta}_{t\to t-1} = \mathbf{c}_{t-1} - \mathbf{c}_t$$

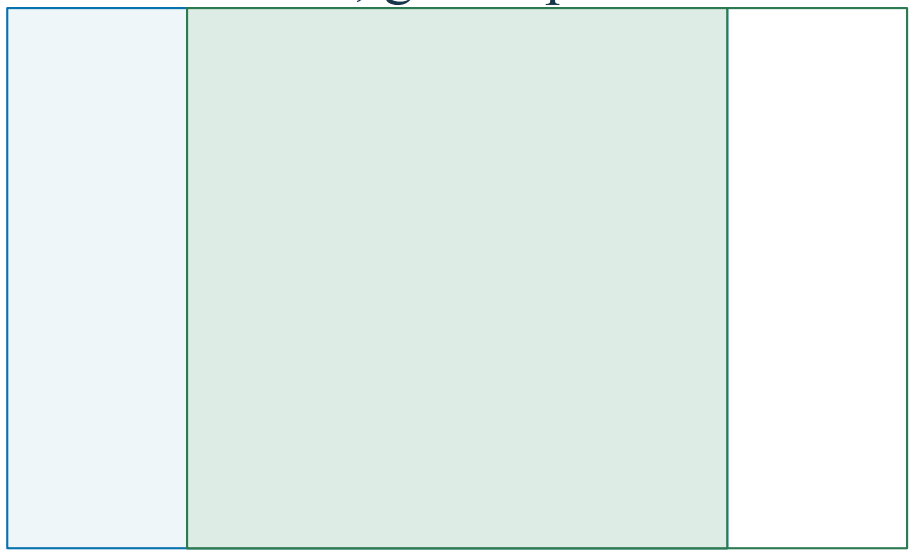
$$\hat{\mathbf{c}}_{t-1} = \mathbf{c}_t + \boldsymbol{\delta}_{t\to t-1}$$

Define the offset direction before locating prior centers. Keeping short tracks is not long-term appearance re-identification.

### 8.2.1 Object Detection and Instance Segmentation: A Box-Overlap Example

# Intersection over Union Quantifies Box Misalignment

Blue: reference; green: prediction



Both boxes: 80 × 60 px
Horizontal offset: 20 px

$$\mathrm{IoU} = \frac{|B \cap G|}{|B \cup G|}$$

$$\mathrm{IoU} = \frac{60 \times 60}{2 \times 80 \times 60 - 60 \times 60} = 0.60$$

Intersection over union (IoU) measures region overlap.
Here the threshold is 0.5; 0.60 passes the overlap test.

### 8.2.1 Object Detection and Instance Segmentation: Checking Detection Outputs

# Check Box Positions, False Detections, and Misses




Fig. 8.3(b) Ring detection

| Offset / px | IoU | ≥ 0.5 |
|---|---|---|
| 0 | 1.00 | Pass |
| 20 | 0.60 | Pass |
| 40 | 0.33 | Fail |

Check each: class, position, misses.
One frame gives no persistent ID.

Chapter Outline

# Target Recognition, Tracking, and Optical Flow Estimation

**Chapter 8 Multimodal Perception and Intelligent Decision-Making**


8.1 Image Processing Fundamentals

**8.2 Recognition, Tracking, Optical Flow**

8.3 Localization, Mapping, and Planning

8.4 Common Perception Sensors

8.5 Coordinate Transformations

8.6 Fusion and Control Decision-Making

8.7 Tools and Framework Interfaces

8.8 Experiments and Real-World Transfer

8.9 Exercises and Classroom Discussion


## 5 Units in This Section


**8.2.1 Object Detection and Instance Segmentation**
Textbook: Sec. 8.1.2.2

**8.2.2 Online Object Tracking and Association**
Textbook: Sec. 8.1.2.2

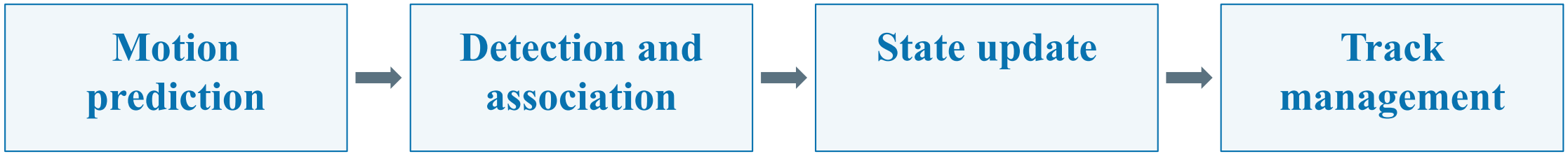
**8.2.3 Multi-View Perception and Pose Estimation**
Textbook: Sec. 8.1.2.2

**8.2.4 Tracking Modes and Visual Servoing**
Textbook: Sec. 8.1.2.2

**8.2.5 Optical Flow Estimation and Motion Analysis**
Textbook: Sec. 8.1.2.3

## 8.2.2 Online Object Tracking and Association: Predict, Associate, and Update

# Persistent Identity Needs State Prediction and Association




- Simple Online and Realtime Tracking (SORT) uses Kalman filtering and Hungarian assignment to maintain multiple tracks.
- Track states—new, confirmed, temporarily lost, and deleted—determine how long identities are retained.
- Crossings, occlusion, and camera motion can make prediction regions overlap and cause identity switches.

### 8.2.2 Online Object Tracking and Association: An Association Example

# One Assignment Determines Both Track Identities

$$\hat{u}_t = u_{t-1} + \dot{u}_{t-1}\Delta t$$

Constructed inputs (px): predictions A = 140, B = 120; detections D1 = 121, D2 = 139.

| Center distance / px | D1: 121 | D2: 139 |
|---|---|---|
| **A: 140** | 19 | 1 |
| **B: 120** | 1 | 19 |

A→D2 and B→D1 cost 2 px in total; the reverse costs 38 px. The gate here is 25 px.

### 8.2.2 Online Object Tracking and Association: Appearance and Matching Cascades

# Gate by Motion, Then Distinguish Appearance

**Motion gating** → **Appearance distance** → **Matching cascade**

$$d_{\text{app}} = 1 - \frac{\mathbf{r}_i^T \mathbf{r}_j}{\|\mathbf{r}_i\| \|\mathbf{r}_j\|}$$

- **Deep SORT adds appearance embeddings:** a network maps an object image to a feature vector. Smaller cosine distance means more similar appearance.
- **Cascade order:** match recently observed tracks first, then tracks with longer gaps since their last update.

### 8.2.2 Online Object Tracking and Association: Joint Detection and Embedding

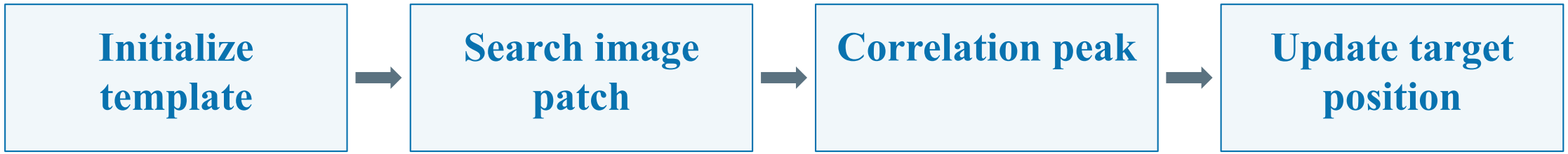
# A Shared Backbone Avoids Repeated Feature Extraction

| Framework | Association Cues | Computation |
|---|---|---|
| **Deep SORT** | Motion gating + appearance embedding | Extract appearance after detection |
| **JDE** | Motion + jointly trained embedding | Share the detection and embedding backbone |
| **CenterTrack** | Centers and interframe offsets | Input two frames and the prior heatmap together |

Joint Detection and Embedding (JDE) shares features; identity still needs association.

### 8.2.2 Online Object Tracking and Association: Kernelized Correlation Filtering

# Correlation Peaks Locate the Target




- **Kernelized Correlation Filter (KCF):** locate one target using the correlation response between a template and a search region.
- **Fast Fourier transform (FFT):** compute circular correlation in the frequency domain, exploiting sample structure to reduce cost.
- **Failure and recovery:** scale changes, occlusion, or similar backgrounds can cause drift; detection can reinitialize the template.

### 8.2.2 Online Object Tracking and Association: Continuous Tracking Results

# Crossings and Misses Test Identity Continuity

| Frame | A center / px | B observation / px | Identity Interpretation |
|---|---|---|---|
| 1 | 80 | 180 | Initialize A and B |
| 2 | 110 | 150 | Update each original identity |
| 3 | 140 | Missed; predict 120 | Retain B's prediction; no observation |
| 4 | 170 | 90 | Reassociate B; no identity swap |

**Prediction bridges short gaps; new observations must confirm identity.**

Chapter Outline

# Target Recognition, Tracking, and Optical Flow Estimation

**Chapter 8 Multimodal Perception and Intelligent Decision-Making**


8.1 Image Processing Fundamentals

**8.2 Recognition, Tracking, Optical Flow**

8.3 Localization, Mapping, and Planning

8.4 Common Perception Sensors

8.5 Coordinate Transformations

8.6 Fusion and Control Decision-Making

8.7 Tools and Framework Interfaces

8.8 Experiments and Real-World Transfer

8.9 Exercises and Classroom Discussion


## 5 Units in This Section


**8.2.1 Object Detection and Instance Segmentation**
Textbook: Sec. 8.1.2.2

**8.2.2 Online Object Tracking and Association**
Textbook: Sec. 8.1.2.2

**8.2.3 Multi-View Perception and Pose Estimation**
Textbook: Sec. 8.1.2.2

**8.2.4 Tracking Modes and Visual Servoing**
Textbook: Sec. 8.1.2.2

**8.2.5 Optical Flow Estimation and Motion Analysis**
Textbook: Sec. 8.1.2.3

### 8.2.3 Multi-View Perception and Pose Estimation: Pose Outputs

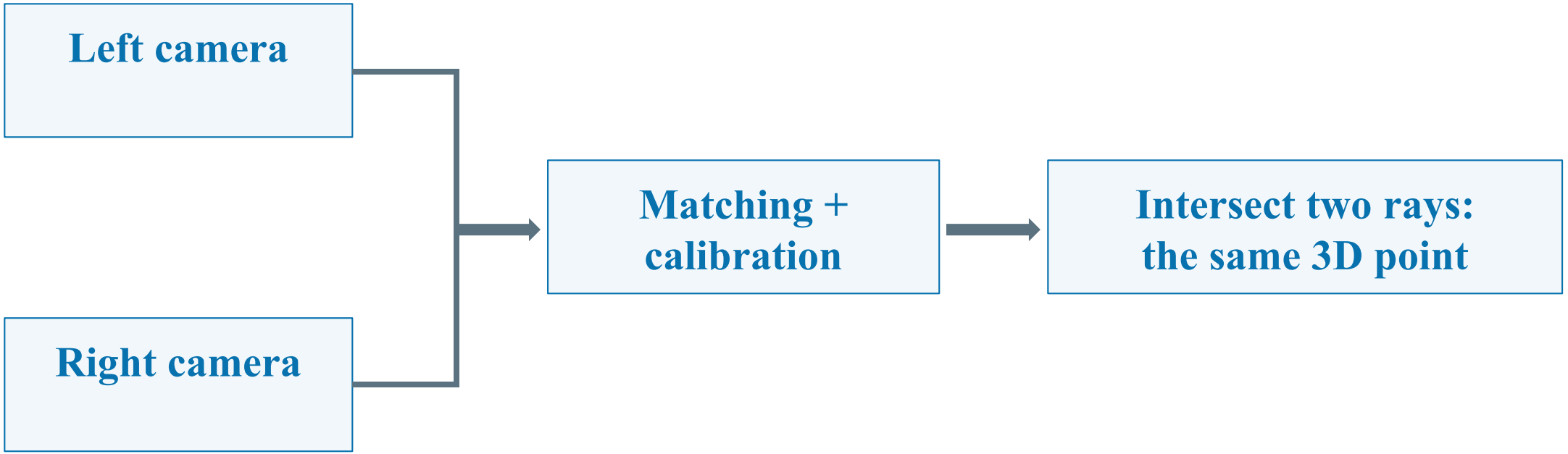
# Human Joints and Rigid-Body 6DoF Are Different Outputs

| Object | Output | Typical Method / Conditions |
|---|---|---|
| **Human skeleton** | Joint positions and connections | Shotton: depth image → 3D joints |
| **Human image** | Joint positions in the image | High-Resolution Network (HRNet): retains fine spatial features |
| **Rigid object** | 3D translation + 3D rotation | Known model, keypoints, or depth geometry |

Rigid-body six degrees of freedom (6DoF) = 3 translations + 3 rotations.

### 8.2.3 Multi-View Perception and Pose Estimation: Multi-View Geometry

# Match the Same Point Before Intersecting the Rays




Perspective-n-Point (PnP): known 3D points and their 2D projections determine the relative camera–object pose.

### 8.2.3 Multi-View Perception and Pose Estimation: A Triangulation Example

# A 40 px Disparity Gives 3 m Depth

- **Geometry:** rectified stereo, a horizontal baseline, and synchronized views of the same point. Depth is along the camera optical axis.
- **Inputs:** focal length 600 px; baseline 0.20 m; disparity 40 px.

$$Z = \frac{f_x B}{d}, \qquad d = u_L - u_R$$

$$Z = \frac{600 \times 0.20}{40} = 3.00 \text{ m}$$

### 8.2.3 Multi-View Perception and Pose Estimation: Reprojection and Bundle Adjustment

# Refine Camera Poses and 3D Points Together

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \pi(R_i\mathbf{P}_j + \mathbf{t}_i)$$

$i$: camera index; $j$: point index; $\mathbf{z}_{ij}$: pixel observation; $\mathbf{r}_{ij}$: reprojection residual; $\pi$: projection.

$$\min_{\{R_i,\mathbf{t}_i,\mathbf{P}_j\}} \sum_{i,j} \|\mathbf{r}_{ij}\|^2$$

- **Symbols:** R is a 3 × 3 rotation matrix; t is a 3 × 1 translation; a 3D point is 3 × 1. Transform into the camera frame, then project to pixels.
- **Conditions:** fix the reference and scale; reject mismatches; constrain each 3D point with observations from multiple views.

### 8.2.3 Multi-View Perception and Pose Estimation: Learning and Multimodal Fusion

# Different Observations Jointly Constrain Pose

| Observation / Method | Constraint Provided | Conditions |
|---|---|---|
| **HRNet / depth-based human pose** | Joint or body-part positions | Output semantics match the human model |
| **Cross-view Transformer** | Feature relationships across views | View matches, training distribution, synchronization |
| **Vision + LiDAR + IMU** | Pixels, range, and short-term motion | Consistent extrinsics, timing, and error weights |

LiDAR: light detection and ranging (range point clouds). IMU: inertial measurement unit. Downweight low-confidence observations.

### 8.2.3 Multi-View Perception and Pose Estimation: Interpreting Error Results

# Distant Depth Is More Sensitive to Disparity Error

| Given Disparity | Measured Too Low | Measured Too High |
|---|---|---|
| **40 px → 3.00 m** | 38 px → 3.16 m | 42 px → 2.86 m |
| **10 px → 12.00 m** | 8 px → 15.00 m | 12 px → 10.00 m |

- Conditions: focal length 600 px, baseline 0.20 m; only disparity changes.
- Farther objects have smaller disparities, so the same pixel error produces a larger error in meters.

Chapter Outline

# Target Recognition, Tracking, and Optical Flow Estimation

**Chapter 8 Multimodal Perception and Intelligent Decision-Making**


8.1 Image Processing Fundamentals

**8.2 Recognition, Tracking, Optical Flow**

8.3 Localization, Mapping, and Planning

8.4 Common Perception Sensors

8.5 Coordinate Transformations

8.6 Fusion and Control Decision-Making

8.7 Tools and Framework Interfaces

8.8 Experiments and Real-World Transfer

8.9 Exercises and Classroom Discussion


## 5 Units in This Section


**8.2.1 Object Detection and Instance Segmentation**
Textbook: Sec. 8.1.2.2

**8.2.2 Online Object Tracking and Association**
Textbook: Sec. 8.1.2.2

**8.2.3 Multi-View Perception and Pose Estimation**
Textbook: Sec. 8.1.2.2

**8.2.4 Tracking Modes and Visual Servoing**
Textbook: Sec. 8.1.2.2

**8.2.5 Optical Flow Estimation and Motion Analysis**
Textbook: Sec. 8.1.2.3

### 8.2.4 Tracking Modes and Visual Servoing: Tracking Modes

## Fixed Cameras, Gimbals, and Vehicles Change Different Things

| Mode | What Changes Actively? | Effect on Tracking |
|---|---|---|
| **Fixed-camera observation** | Sensor pose is not actively adjusted | Detect and annotate continuously |
| **Dynamic gimbal tracking** | Camera viewing direction | Keep the target in view or centered |
| **Vehicle visual feedback** | Vehicle position and orientation | Change viewpoint, heading, and relative distance |

"Static" describes the observing platform; the target can still move.

## 8.2.4 Tracking Modes and Visual Servoing: Visual Servoing Information Paths

# Errors Come from 3D Pose or Image Features

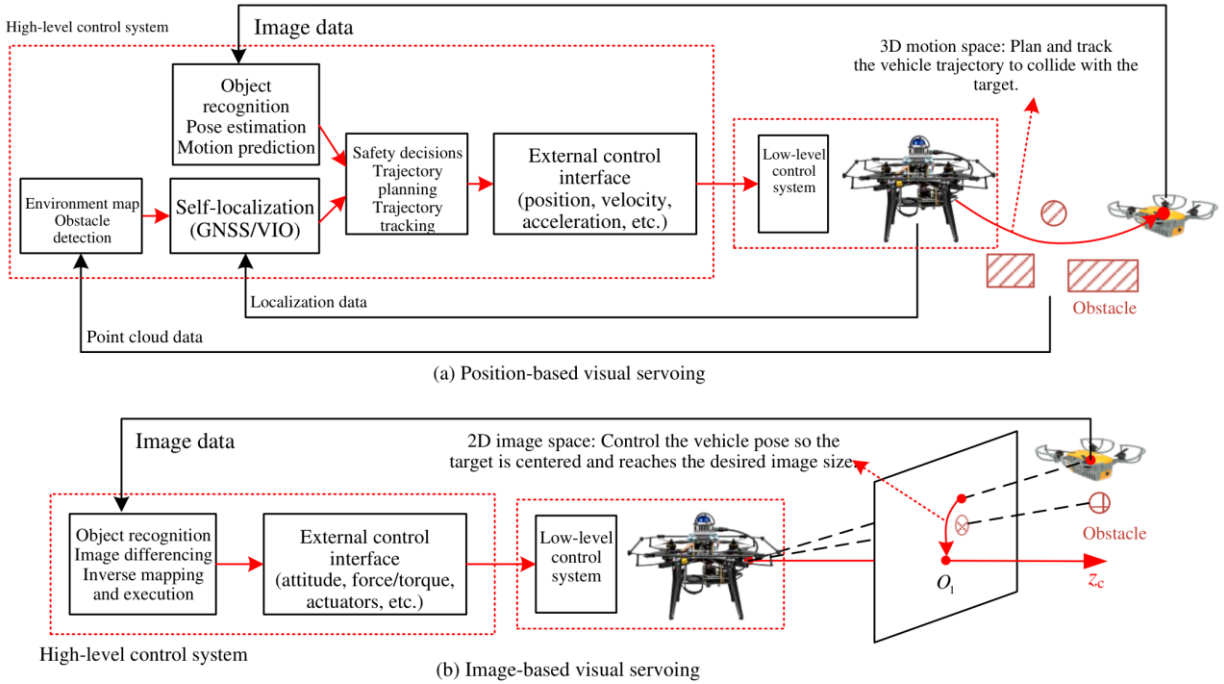


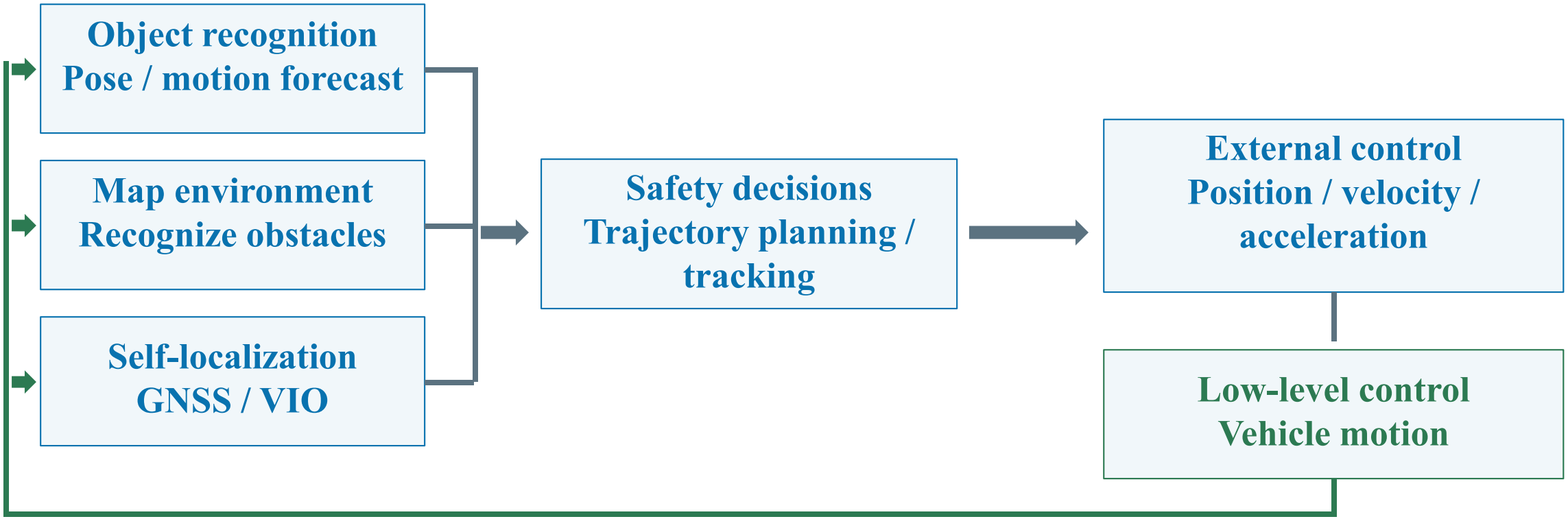

Fig. 8.2 Visual servoing control methods

(a) Uses 3D pose error. (b) Uses image-feature error.

### 8.2.4 Tracking Modes and Visual Servoing: The Pose-Error Control Chain

## 3D States Pass Through Planning to Position Interfaces

GNSS: global navigation satellite system; VIO: visual-inertial odometry.

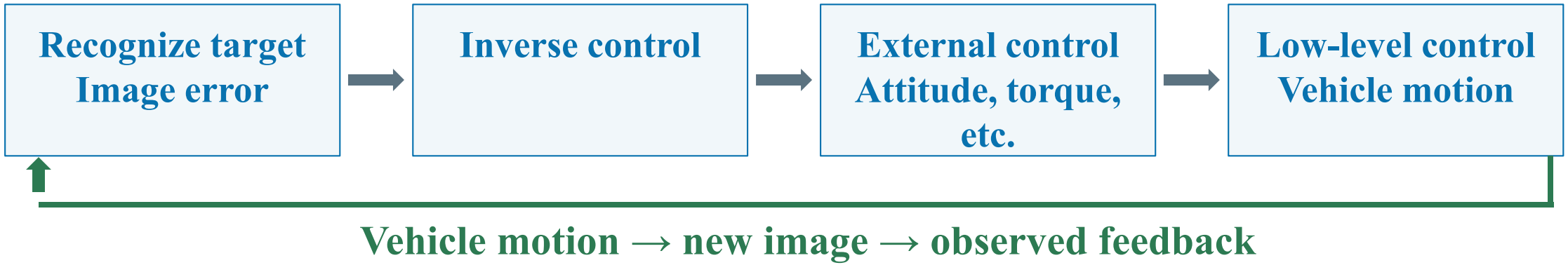



Feedback: images, localization, point clouds; (a) adds mapping and trajectory planning.

## 8.2.4 Tracking Modes and Visual Servoing: The Image-Error Control Chain

# Inverse Control Maps Image Error to Motion Commands

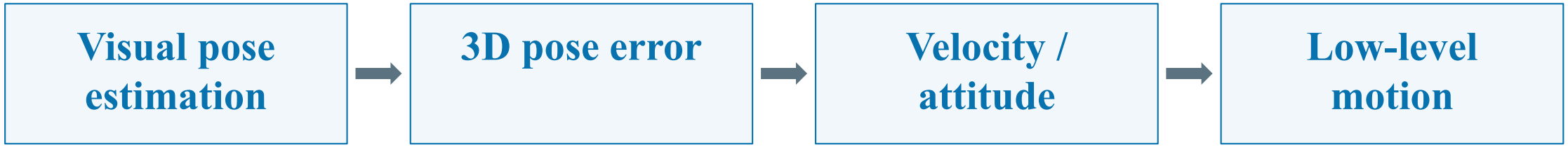
Fig. 8.2(b): subtract desired image features from current features; a complete 3D map is not required first.




- **Image objective:** keep the projection at the desired position and scale. The error can contain multiple features.
- The design selects attitude, torque, or velocity interfaces; image-based visual servoing is defined by its error features.

### 8.2.4 Tracking Modes and Visual Servoing: Position-Based Visual Servoing

# Relative Pose Error Guides 3D Motion




$$\mathbf{e}_p = \mathbf{s}_p - \mathbf{s}_p^*$$

- **Position-based visual servoing (PBVS):** error features include 3D translation and rotation relative to the target.
- A planner can be added for obstacle avoidance; basic pose feedback does not require a global map.

### 8.2.4 Tracking Modes and Visual Servoing: Image-Based Visual Servoing

# The Image Jacobian Connects Feature Changes to Velocity

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^*, \qquad \dot{\mathbf{s}} = \mathbf{L}(\mathbf{s}, Z)\mathbf{v}_c$$

$\mathbf{s}, \mathbf{s}^*$: current and desired features; $\mathbf{e}$: error; $Z$: depth; $\mathbf{v}_c$: camera velocity.

$$\mathbf{v}_c = -\lambda \hat{\mathbf{L}}^{+} \mathbf{e}$$

$\hat{\mathbf{L}}^{+}$: pseudoinverse of the estimated interaction matrix; $\lambda$: positive gain.

- **Image-based visual servoing (IBVS):** image features define the error; the interaction matrix relates their rates of change to camera velocity.
- **Conditions:** feature units depend on the choice; camera velocity includes translation and rotation; gain is positive; local geometry is valid.

### 8.2.4 Tracking Modes and Visual Servoing: One Lateral Control Step

# A 40 px Rightward Error Gives 0.2 m/s Rightward Motion

Simplification: stationary target at fixed depth 2 m; camera axes right, down, forward; horizontal translation only.

$$\dot{u} = -\frac{f_x}{Z} v_{c,x}, \qquad v_{c,x} = \frac{\lambda Z}{f_x} e_u$$

$$v_{c,x} = \frac{1 \times 2}{400} \times 40 = 0.20 \ \text{m/s}$$

$$e_1 = 40 - \frac{400}{2} \times 0.20 \times 0.10 = 36 \ \text{px}$$

Focal length 400 px; gain 1/s; interval 0.10 s; positive velocity points right.

### 8.2.4 Tracking Modes and Visual Servoing: Closed-Loop Iteration Results

# Error Falls by a Factor of 0.9; Velocity Falls with It

$$e_{k+1} = (1 - \lambda \Delta t) e_k = 0.9 e_k$$

| Update Step | Image Error / px | Camera Rightward Velocity / m/s |
|---|---|---|
| 0 | 40.00 | 0.200 |
| 1 | 36.00 | 0.180 |
| 10 | 13.95 | 0.0697 |
| 20 | 4.86 | 0.0243 |

Offline iteration: fixed depth, stationary target, no delay, and no velocity saturation.

### 8.2.4 Tracking Modes and Visual Servoing: End-to-End and Hybrid Strategies

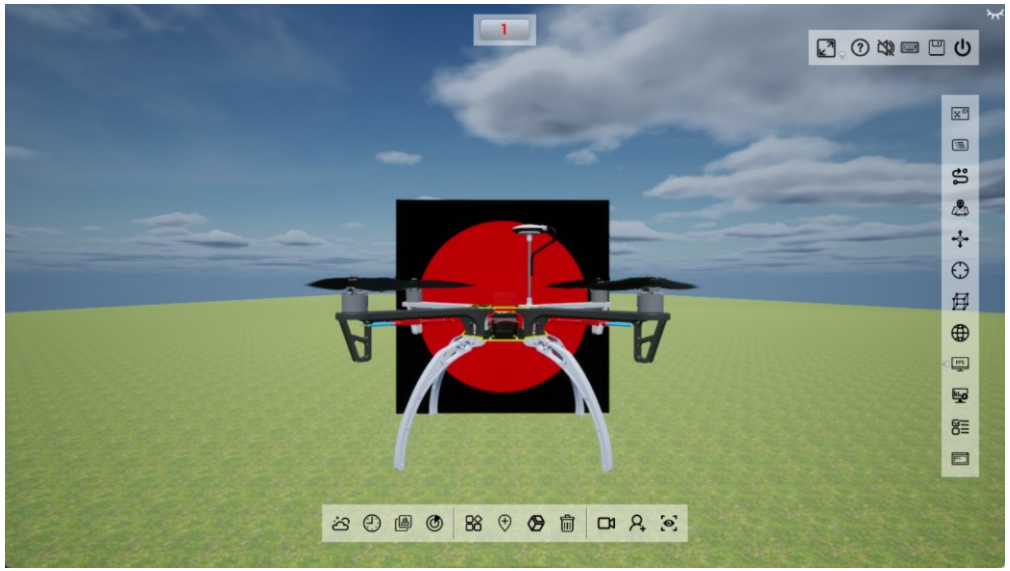
## Learned Mappings and Geometric Feedback Have Conditions

| Approach | Error or Mapping | Key Conditions |
| --- | --- | --- |
| **PBVS** | Relative 3D pose error | Target model, calibration, and depth |
| **IBVS** | Image-feature error | Visible features and local interaction matrix |
| **End-to-end** | Map images directly to actions | Training distribution, latency, and output constraints |

Hybrid: reach a suitable observation area, then switch to image feedback based on confidence, distance, and limits.

## 8.2.4 Tracking Modes and Visual Servoing: Circle-Following Source Code

# Center and Size Errors Generate Velocity References



Circle-following example scene

Aligned front camera; vehicle nearly level
Center right/down → right/down velocity
Box height < initial → forward velocity
Horizontal follows heading; vertical stays North-East-Down (NED) down.
P only; 15 px center and size deadbands

Lost target: skip sending. Esc exits.
Stopping streaming, stopping motion, and landing are separate actions.

Student entry: [installation directory]/RflySimAPIs/8.RflySimVision/1.BasicExps/1-VisionCtrlDemos/e6_Circle-follow/circle_follow_autoCtrl.py

Chapter Outline

# Target Recognition, Tracking, and Optical Flow Estimation

**Chapter 8 Multimodal Perception and Intelligent Decision-Making**


8.1 Image Processing Fundamentals

**8.2 Recognition, Tracking, Optical Flow**

8.3 Localization, Mapping, and Planning

8.4 Common Perception Sensors

8.5 Coordinate Transformations

8.6 Fusion and Control Decision-Making

8.7 Tools and Framework Interfaces

8.8 Experiments and Real-World Transfer

8.9 Exercises and Classroom Discussion


## 5 Units in This Section

**8.2.1 Object Detection and Instance Segmentation**
Textbook: Sec. 8.1.2.2

**8.2.2 Online Object Tracking and Association**
Textbook: Sec. 8.1.2.2

**8.2.3 Multi-View Perception and Pose Estimation**
Textbook: Sec. 8.1.2.2

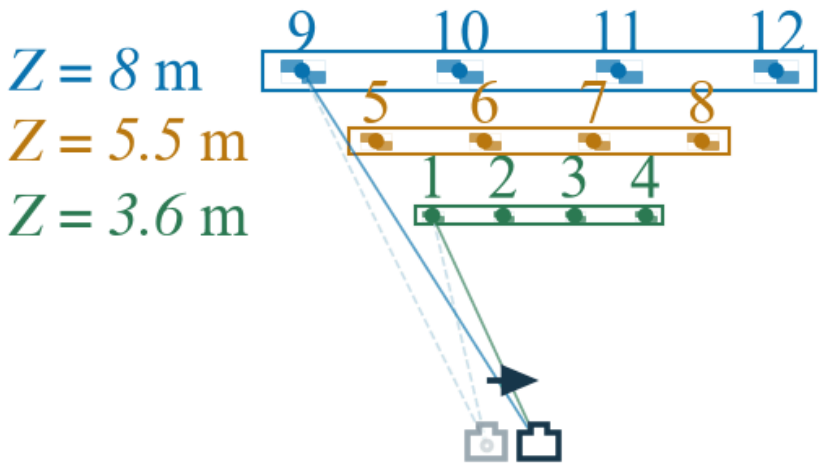
**8.2.4 Tracking Modes and Visual Servoing**
Textbook: Sec. 8.1.2.2

**8.2.5 Optical Flow Estimation and Motion Analysis**
Textbook: Sec. 8.1.2.3

### 8.2.5 Optical Flow Estimation and Motion Analysis: Image Motion

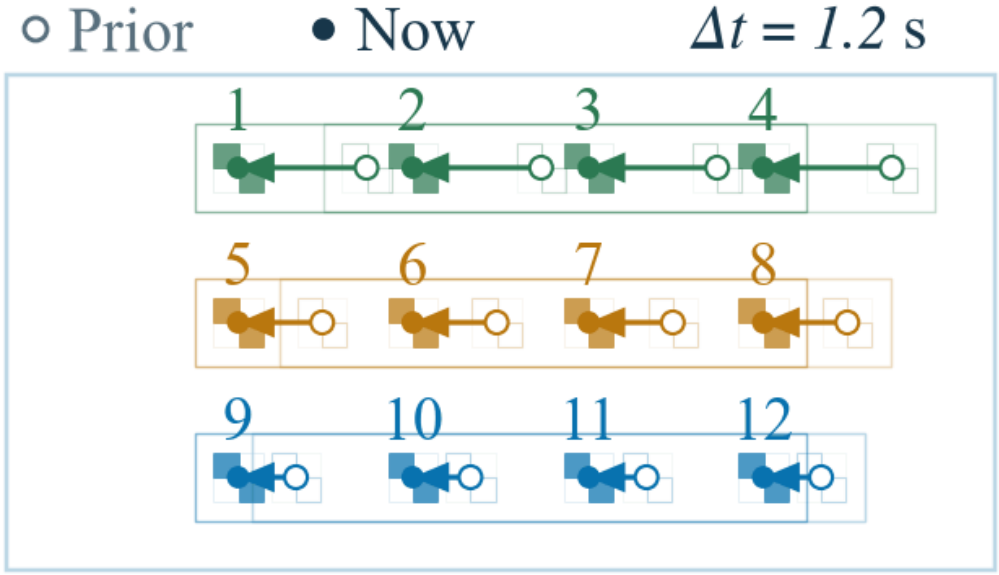
# Optical Flow Gives Pixel Motion; Speed Needs Scale

**Camera shift: opposite image motion, larger for near points**




Camera shift right

Mean $\bar{v}_x = +0.594$ m/s



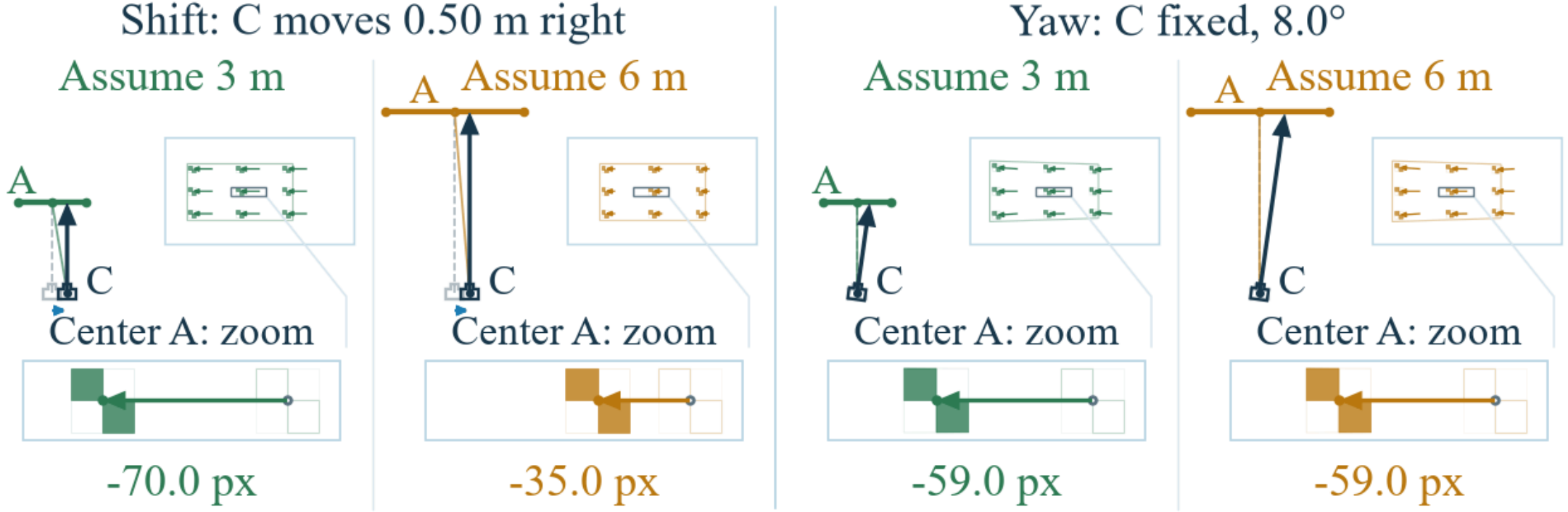

1: $\Delta u = -83.16$ px

Static scene, pure lateral camera translation: image points move oppositely; nearer points move farther.

### 8.2.5 Optical Flow Estimation and Motion Analysis: Sources of Camera-Induced Motion

# Translation and Rotation Both Produce Optical Flow

Points on one initial ray: translation depends on depth; pure rotation does not.




Projection comparison: optical center shifts 0.50 m right versus an 8° rotation about that center.

### 8.2.5 Optical Flow Estimation and Motion Analysis: Brightness Constraints and Observability

## One Pixel Cannot Determine 2D Motion Alone

$$I(u, v, t) \approx I(u + \Delta u, v + \Delta v, t + \Delta t)$$

$$I_u \Delta u + I_v \Delta v + I_t \Delta t \approx 0$$

$I$: brightness; $I_u, I_v, I_t$: horizontal, vertical, and temporal partial derivatives; $\Delta$: the corresponding increment.

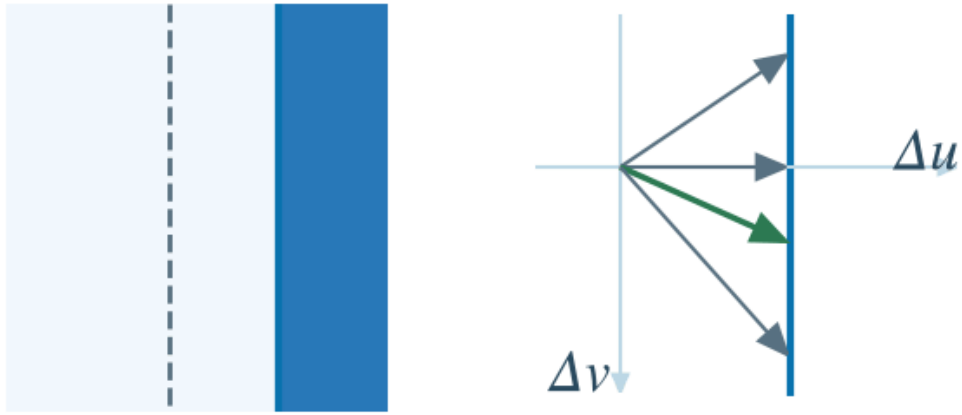
- **Brightness constancy and small motion:** corresponding image structure, little lighting change, and displacement small enough for a first-order approximation.
- **Aperture problem:** a single edge direction leaves motion along the edge ambiguous. Corners and neighborhood constraints supply more information.

### 8.2.5 Optical Flow Estimation and Motion Analysis: The Aperture Problem

## Edges Leave Tangential Ambiguity; Corners Constrain 2D Motion

Vertical edge: horizontal motion only. Independent corner gradients fix 2D motion.

Edge: tangent is ambiguous

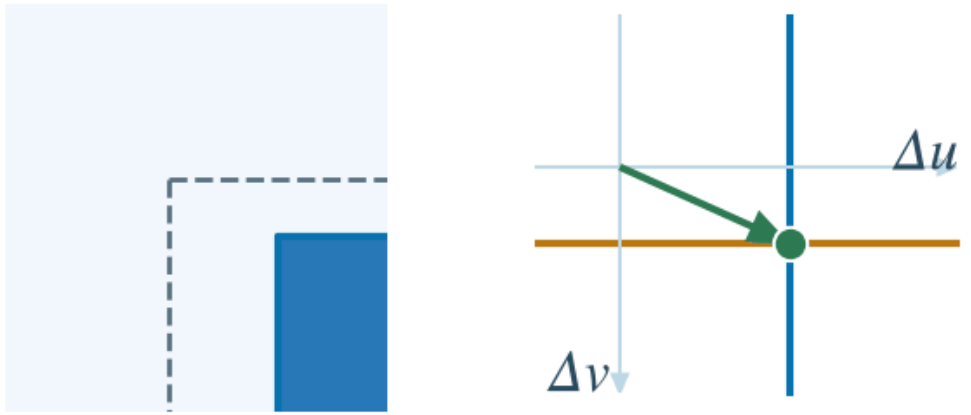



Zoomed patch

Flow constraints

Same horizontal shift, many tangents

Corner: two constraints




Zoomed patch

Flow constraints

Intersection (2.0, 1.0) px

Displacement (2, 1) px, matching the local worked example.

### 8.2.5 Optical Flow Estimation and Motion Analysis: Dense and Sparse Methods

# Different Methods Supply Different Motion Constraints

| Method | Added Constraint | Typical Output / Limits |
|---|---|---|
| **Horn–Schunck** | Brightness constancy + global smoothness | Dense; may smooth motion boundaries |
| **Farneback** | Local quadratic polynomial approximation | Dense; depends on local texture and scale |
| **Lucas–Kanade** | Neighboring points share local motion | Often used for sparse feature tracking |

Dense: all pixels; sparse: features. Both face lighting changes, occlusion, and large motion.

### 8.2.5 Optical Flow Estimation and Motion Analysis: A Local Worked Example

# Three Gradient Constraints Give a (2, 1) px Displacement

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

$$(A^T A)\boldsymbol{\delta} = A^T \mathbf{b}, \qquad \boldsymbol{\delta} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \text{ px}$$

The gradients are constructed inputs. Two independent directions are needed to determine 2D displacement.

### 8.2.5 Optical Flow Estimation and Motion Analysis: Learned Optical Flow

# From Learned Matching to Iterative Correlation Updates

| Network | Core Structure | Output and Conditions |
|---|---|---|
| **FlowNet** | Learn paired-image features and matching | Supervised training for a 2D flow field |
| **PWC-Net** | Pyramid + warping + cost volume | Estimate residual motion at multiple scales |
| **RAFT** | All-pairs correlation + recurrent updates | Refine flow iteratively; compute depends on configuration |

Occlusion, training distribution, and resolution still matter. Learned flow does not automatically have metric scale.

### 8.2.5 Optical Flow Estimation and Motion Analysis: A Depth-to-Velocity Example

# A −4 px Ground Displacement Gives 0.2 m/s

Conditions: downward view of level ground; depth 2 m; focal length 400 px; interval 0.10 s; camera rotation removed.
Image right is positive; ground-point displacement is −4 px.

$$v_{c,x} = -\frac{Z}{f_x}\frac{\Delta u}{\Delta t}$$

$$v_{c,x} = -\frac{2}{400}\frac{-4}{0.10} = 0.20 \ \text{m/s}$$

Depth restores scale; angular velocity compensates rotation. Extrinsics then map camera velocity to body velocity.

### 8.2.5 Optical Flow Estimation and Motion Analysis: Velocity and Integration Results

# Metric Velocity Still Integrates into Position Drift

| Interval End / s | Image Motion / px | Velocity / m/s | Total Motion / m |
|---|---|---|---|
| **0.10** | −4 | +0.20 | 0.02 |
| **0.20** | −4 | +0.20 | 0.04 |
| **0.30** | −4 | +0.20 | 0.06 |

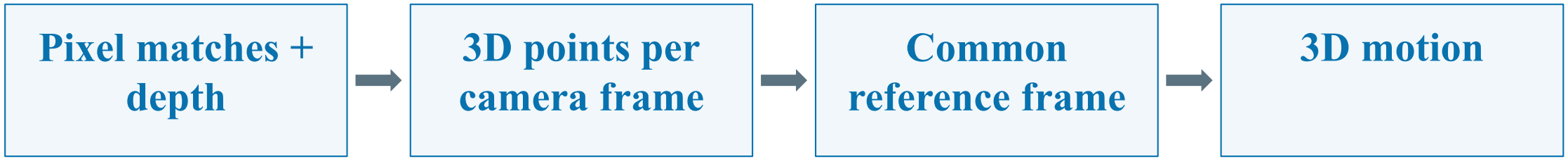
Separate case: constant velocity bias 0.01 m/s for 60 s. Find the position bias.

$$\Delta x_{\text{bias}} = 0.01 \times 60 = 0.60 \text{ m}$$

**Integration accumulates a small velocity bias into long-term position drift.**

### 8.2.5 Optical Flow Estimation and Motion Analysis: Recovering 3D Motion

# Back-Project with Depth, Then Compensate Camera Motion




$$\mathbf{P}_t^w = R_{c,t}^w (Z_t K^{-1} \tilde{\mathbf{p}}_t) + \mathbf{t}_{c,t}^w$$

$w$: fixed reference frame; $R$ and $\mathbf{t}$: rotation and translation from the camera frame to $w$.

$$\mathbf{v}^w = \frac{\mathbf{P}_{t+1}^w - \mathbf{P}_t^w}{\Delta t}$$

R: 3 × 3; t and P: 3 × 1; homogeneous pixels: 3 × 1. Back-projection uses intrinsics and optical-axis depth.

Exercises and Discussion

# What Is Missing from Detection, Identity, and Pose?

- **Q1:** Two objects cross. Their boxes remain intact, but identifiers A and B swap. Which stage failed? How can it improve?

- **Q2:** HRNet outputs human joint positions. Can these directly represent one rigid body's six-degree-of-freedom pose?

- **Q3:** A stationary target is 40 px right of center. The camera translates left. Does image error grow or shrink, and why?

Reference Points

# Check the Output Before Choosing Association and Feedback

- **A1:** The identity switch occurs in association. Add motion gating, appearance cues, and short-term track retention; recheck identity continuity.

- **A2:** Human joints describe a skeletal configuration. Rigid pose describes 3D translation and rotation and needs a target model and geometric constraints.

- **A3:** Moving the camera left shifts the stationary target farther right, increasing error. In this simplified case, move the camera right.

Exercises and Discussion

# Use Depth and Time to Convert Flow to Velocity

Problem: stationary level ground, downward view. Depth 1.5 m; focal length 300 px; interval 0.05 s.
Rotation-compensated horizontal displacement: −3 px. Find rightward camera velocity.

$$v_{c,x} = -\frac{1.5}{300}\frac{-3}{0.05} = 0.30 \text{ m/s}$$

- **Answer:** 0.30 m/s to the right; interval displacement is 0.015 m. Displacement and velocity have different units.
- **Changed condition:** if rotation causes all the flow, compensate rotation first; translational velocity should be near zero.

Section Summary

# From Object Observations to States for Decision-Making

- **Object outputs:** classes, boxes, instance masks, persistent identities, and poses each solve different problems.
- **Geometry and feedback:** joints are not rigid-body poses; position-based and image-based servoing use different error features.
- **Motion scale:** optical flow needs depth, time, and camera-motion compensation to recover metric velocity.
- **Common errors:** a fixed platform can observe moving targets; short-term prediction is not a new observation; integration can still drift.