Chapter Outline

RflySim Swarm Framework and Interfaces

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

RL = reinforcement learning

10.7.1 Simulation Framework and Model Trade-Offs

Textbook: Sec. 10.2.1

10.7.2 Interfaces and the Control Loop

Textbook: Sec. 10.2.3; algorithm development: Sec. 10.2.2

10.7.3 Deployment and Data Organization

Textbook: Secs. 10.2.1–10.2.2

10.7.4 Transfer to Real Vehicles

Textbook: Secs. 10.2.2–10.2.3

10.7.1 Simulation Framework and Model Trade-Offs: Overall Framework

RflySim Swarm Simulation Framework

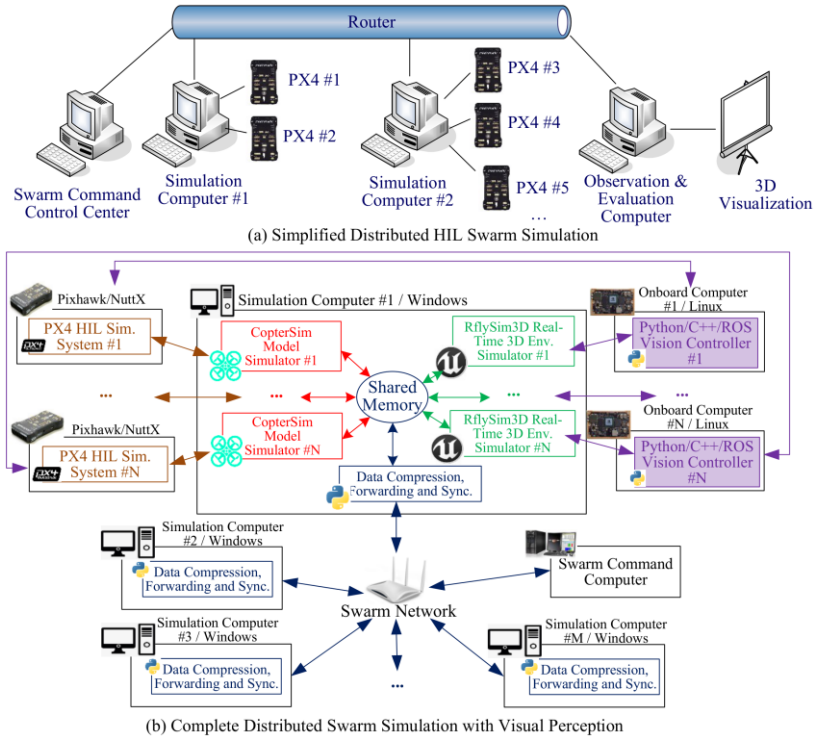


Fig. 10.8 RflySim Swarm Simulation Framework

Loop: Algorithms, states, commands. Hardware-in-the-loop (HITL; textbook HIL).

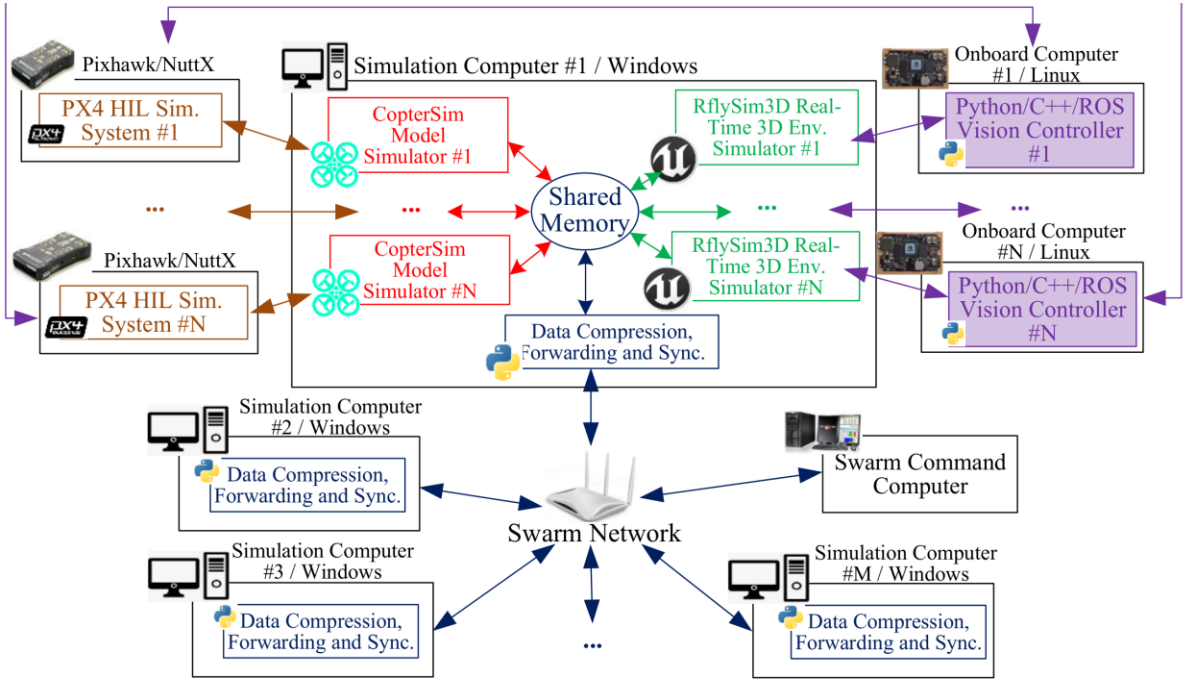
Difference: Generate and process raw images and point clouds?

Read the figure: Displayed scenes and algorithm observations differ.

10.7 RflySim Framework and Interfaces

10.7.1 Simulation Framework and Model Trade-Offs: Visual Chain Detail

Perception and Control Signals in Fig. 10.8(b)



(b) Complete Distributed Swarm Simulation with Visual Perception

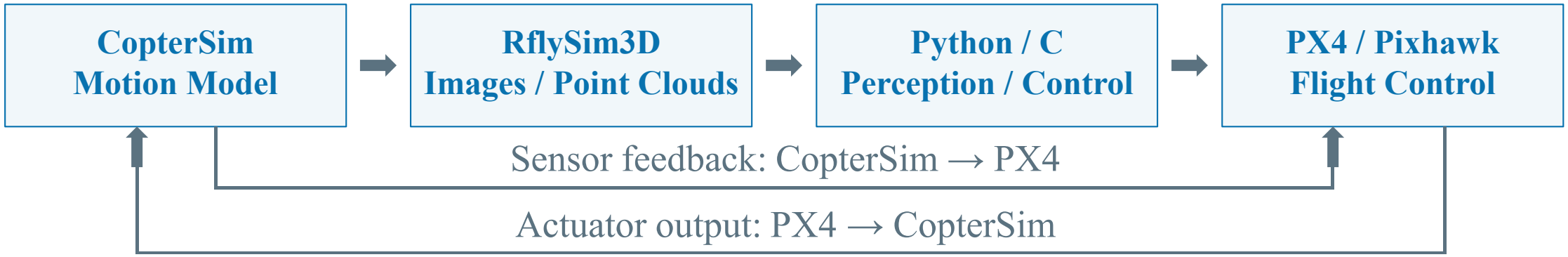
Fig. 10.8(b), detail: Visual processing on one host (from textbook Fig. 10.8)

CopterSim: motion. RflySim3D: observations. Linux algorithms: targets. PX4: execution.

10.7.1 Simulation Framework and Model Trade-Offs: Components and Signal Directions

Model, Perception, and Control in Fig. 10.8

Complete perception: Model and scene → onboard algorithms → flight control.



On one host: CopterSim and RflySim3D exchange states through shared memory.

Across hosts: Aggregation, compression, forwarding, and synchronization programs connect simulation hosts and swarm command over the network.

Simplified perception: Read pose, map, and target results directly, bypassing raw image and point-cloud processing.

10.7.1 Simulation Framework and Model Trade-Offs: Model Choices

Perception and Dynamics Are Separate Choices

Dimension	Choice and Retained Content	Question It Can Address
Perception pipeline	Direct pose, or images / point clouds processed by perception	Do observation errors affect decisions?
Motion model	Point mass / full dynamics / PX4 control loop	Do execution dynamics and flight modes matter?
HITL	Real autopilot or onboard computer in the loop	Do timing, interfaces, and load meet requirements?
Compute scale	Record node and sensor counts, rates, and compute resources.	Can this configuration maintain the required cycle?

10.7.1 Simulation Framework and Model Trade-Offs: Architecture Distinctions

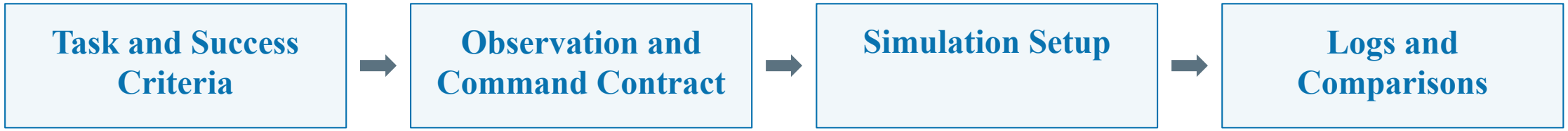
Control, Deployment, and Model Fidelity

Example	Deployment and Model	Control Relationship
8 vehicles on 2 hosts	Each host simulates 4 vehicles.	One controller computes all 8 commands: centralized control.
4 programs, chain following	4 independent programs, possibly on one host	Each uses its predecessor's state: distributed control.
1 PX4 + 4 point masses	Nodes use different model fidelities.	Inspect the control programs' information dependencies.

Distributed simulation does not imply distributed control. Mixed models do not imply a hybrid control architecture.

10.7.1 Simulation Framework and Model Trade-Offs: Development Goals

Task Requirements Guide the Simulation Setup



Formation task: Input a common reference and offsets. Record formation error, minimum separation, and velocity limiting.

Perception task: Input images or point clouds. Record recognition and localization errors, plus state freshness.

Resource task: Record cycle time, deadline-miss ratio, and real-time factor. Measure node capacity for the chosen configuration.

Chapter Outline

RflySim Swarm Framework and Interfaces

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.7.1 Simulation Framework and Model Trade-Offs

Textbook: Sec. 10.2.1

10.7.2 Interfaces and the Control Loop

Textbook: Sec. 10.2.3; algorithm development: Sec. 10.2.2

10.7.3 Deployment and Data Organization

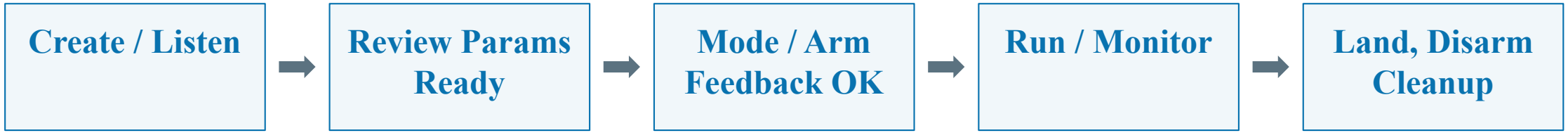
Textbook: Secs. 10.2.1–10.2.2

10.7.4 Transfer to Real Vehicles

Textbook: Secs. 10.2.2–10.2.3

10.7.2 Interfaces and the Control Loop: Creation, Reception, and Control

Python Lifecycle and Control Side Effects



Direct PX4 Micro Air Vehicle Link (MAVLink): This version's initOffboard writes failsafe parameters and can request arming. Review first.

Cleanup: If still armed, stopRun requests forced disarming before stopping reception.

Order: Confirm landing and disarming, then call endOffboard and stopRun. Review parameter policy separately for real vehicles.

10.7.2 Interfaces and the Control Loop: Code Example

Objects and Targets in the Four-Vehicle MAVLink Example

Directory: 0.ApiExps/e2.PyRflySwarmAPIExps/1.MAVLinkFull4Swarm/
Control file: MAVLinkFull4Swarm.py

```
mav = PX4MavCtrl.PX4MavCtrler(1)
mav.InitMavLoop()
mav.initOffboard()
mav.SendPosNED(0, 0, -1.7, 0)
```

One object per ID (1–4). Local north-east-down (NED), negative z up. Targets: (0,0,-1.7), (2,2,-1.7), (-2,-2,-1.7), (0,0,-3) m.

initOffboard can request arming and change failsafe parameters. Review each effect before real-vehicle deployment.

10.7.2 Interfaces and the Control Loop: Coordinate Contract

Align Origins Before Comparing Relative Positions

$$p_i^G = R_i p_i^L + o_i$$

Common and local positions and origin offsets are in m. Rotation is dimensionless. Below, local axes are aligned, so the rotation is the identity.

Vehicle	Local Position / m	Origin Offset / m	Common Position / m
1	(0,0,-2)	(0,0,0)	(0,0,-2)
2	(0,0,-2)	(4,0,0)	(4,0,-2)

Teaching result: Identical local coordinates still give 4 m separation in the common frame.

10.7.2 Interfaces and the Control Loop: Communication Modules

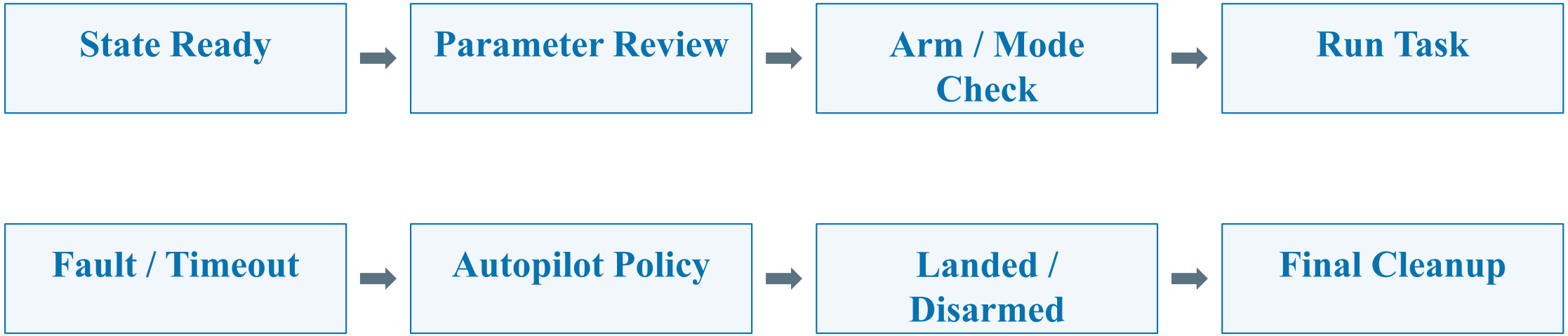
Simulink Modules and Their Data Sources

Module	Input / Output	Checks
RflyUdpFast	User Datagram Protocol (UDP) structures: states and controls	Full / Simple format, IDs, transmit and receive ports
RflyUdpMavlink	MAVLink state streams and control targets	Flight mode, target level, fields, frames
RflyVrpnRecv	Rigid-body pose and validity feedback	Rigid-body name, server, time, frames

The model still implements control logic. A connected interface establishes only the data link.

10.7.2 Interfaces and the Control Loop: Readiness and Exit

External Control Requires State Gating



Entry: Review parameters and protection policy. Confirm valid pose, ID, control mode, and arming feedback.

Exit: Confirm landing and disarming before cleanup. stopRun requests forced disarming if still armed.

10.7.2 Interfaces and the Control Loop: Namespaces and Publish–Subscribe

ROS 2 and DDS Messaging Across Nodes

ROS 2 = Robot Operating System 2. DDS = Data Distribution Service.

Per-Vehicle State Topics → **Independent Control Node** → **Mode and Target Topics** → **Flight Controller Bridge**

Entry: 0.ApiExps/e10.ROS2Swarm/
SITL4Run.bat, WinWSL_StartuXRCE.bat, control package px4_swarm_control.zip.

Save other simulations. SITL4Run.bat requests admin rights and stops CopterSim.
Cleanup also closes QGroundControl and RflySim3D.

Mapping: Match each vehicle's namespace, target system ID, coordinate frame, and timestamp.

Require compatible ROS 2 / PX4 messages and bridge versions. Verify subscriptions, quality of service (QoS), and mode feedback before motion.

Chapter Outline

RflySim Swarm Framework and Interfaces

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.7.1 Simulation Framework and Model Trade-Offs

Textbook: Sec. 10.2.1

10.7.2 Interfaces and the Control Loop

Textbook: Sec. 10.2.3; algorithm development: Sec. 10.2.2

10.7.3 Deployment and Data Organization

Textbook: Secs. 10.2.1–10.2.2

10.7.4 Transfer to Real Vehicles

Textbook: Secs. 10.2.2–10.2.3

10.7.3 Deployment and Data Organization: Distributed Simulation

Two Hosts Simulate Eight Vehicles, One Computes Control



Two-way exchange: The local network carries states and commands. One model sends all 8 targets by ID.

Deployment: Run the control model on host A or B. A third host is unnecessary.

10.7.3 Deployment and Data Organization: Hosts, IDs, and Ports

Eight-Vehicle Local Network: Setup and Entry Files

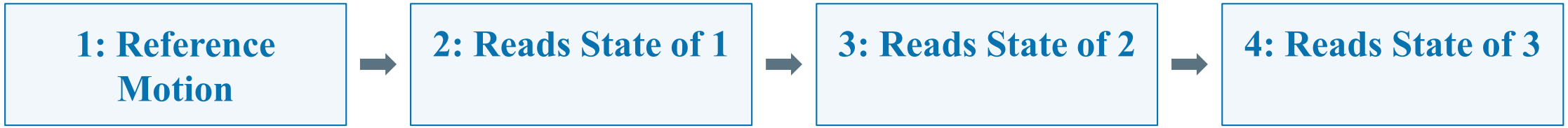
Root: [installation directory]/RflySimAPIs/10.RflySimSwarm/
Subdirectory: 2.AdvExps/e2.MatDISCtrlUAVsSim/1.BroadNetSwarm_Mat/

Role	Entry File	Check After Launch
Host A	SITLRunUdpSimple1_4.bat	States for IDs 1–4 update independently.
Host B	SITLRunUdpSimple5_8.bat	IDs 5–8 have no duplicates or cross-wiring.
Controller	RflyUdpUltraSimpleEight_Dist.slx	Enable control after all 8 state channels are ready.

Check Readme.pdf, versions on both hosts, and port mappings before following its steps.

10.7.3 Deployment and Data Organization: Distributed Control Comparison

Control Information in Four-Program Chain Following



Full edition or higher only:
3.CustExps/e2.PyDISCtrlUAVsSim/e1_FourUAVSwarmTrack/

Environment: SITLRun4MavlinkFull.bat. Control: UavPythonRunALL.bat.
Separate programs: UAV1Ctrl.py through UAV4Ctrl.py.

Observe each following error, state age, and response to predecessor loss. Multiple processes alone do not prove convergence.

10.7.3 Deployment and Data Organization: Mixed Models

One PX4 Leader and Four Point-Mass Followers



Entry: 2.AdvExps/e4_PX4+MPSwarm/
UAVCtrlSILRun.bat → Python38Run.bat → HybridSwarmCtrl.py

Mechanism: Leader feedback drives follower targets. Following offsets are 3 m. Initial placement spacing is 2 m.

Validation: Compare target and actual relative positions. Log delay, tracking error, minimum separation, and shutdown behavior.

10.7.3 Deployment and Data Organization: Identity, Time, and Validity

A Data Contract Across Hosts

Information	Record	Failure Symptom
Identity	Vehicle ID, host, ports, message source	Cross-wired states or commands to the wrong vehicle
Space	Axes, common origin, units, attitude convention	Reversed directions, shifted or rotated formation
Time	Generation, receipt, and use times, clock error bound	Stale states, incorrect network-delay estimates
Validity	Localization flag, mode feedback, sequence, timeout policy	Old values remain in control.

Chapter Outline

RflySim Swarm Framework and Interfaces

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.7.1 Simulation Framework and Model Trade-Offs

Textbook: Sec. 10.2.1

10.7.2 Interfaces and the Control Loop

Textbook: Sec. 10.2.3; algorithm development: Sec. 10.2.2

10.7.3 Deployment and Data Organization

Textbook: Secs. 10.2.1–10.2.2

10.7.4 Transfer to Real Vehicles

Textbook: Secs. 10.2.2–10.2.3

10.7.4 Transfer to Real Vehicles: Localization and Control

Indoor Motion Capture Network Topology

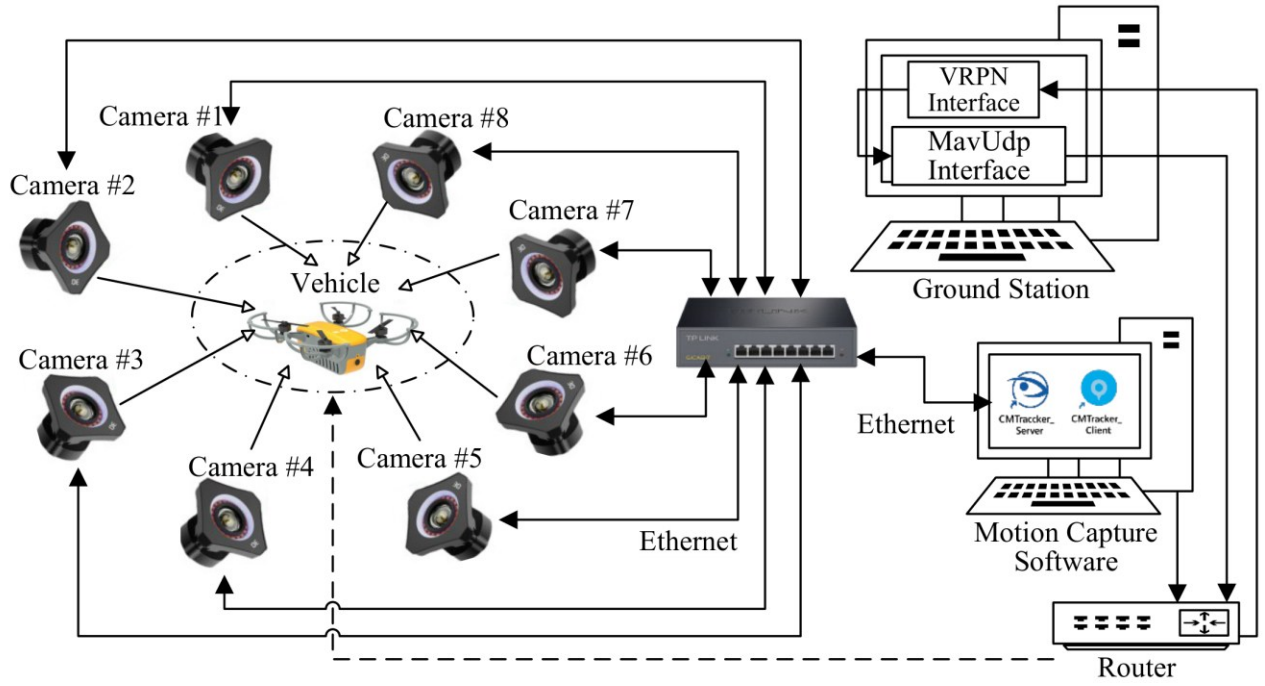


Fig. 10.9 Indoor Motion Capture Platform Network Topology

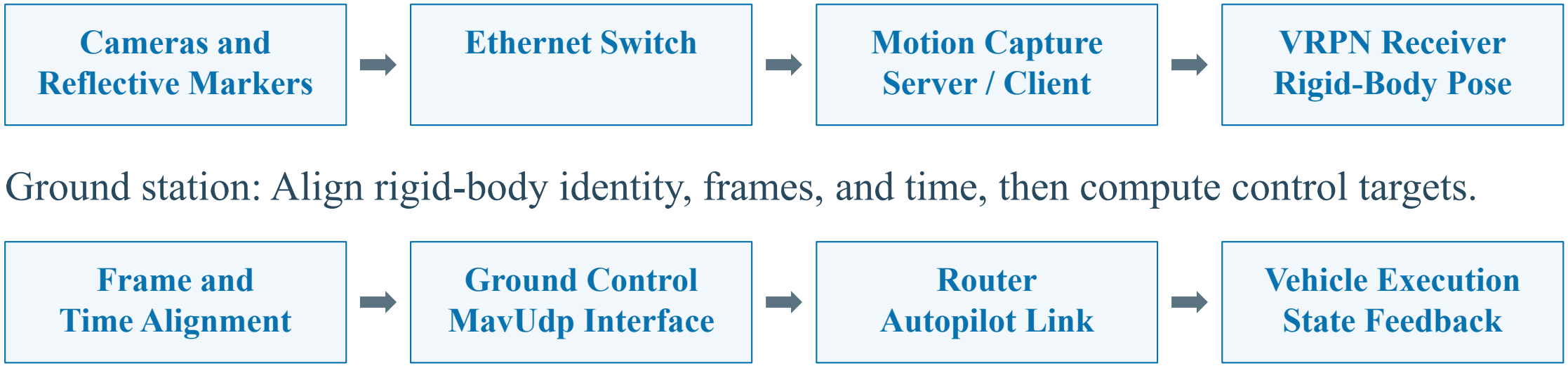
Localization: Reflective markers, cameras, and rigid-body pose.

Data: Virtual-Reality Peripheral Network (VRPN) or NatNet publishes poses.

Control: The program sends targets through the flight-controller interface.

10.7.4 Transfer to Real Vehicles: VRPN and MavUdp

Fig. 10.9: Motion Capture to Control Commands



Match rigid-body names to vehicle IDs. Use static and single-axis tests to check units, directions, validity, and timestamps.

Entry: 0.ApiExps/e1.MatRflySwarmAPIExps/5.RflyVrpnRecvAPIExps/ RflyVrpnRecvAPIExp.slx. Read Readme.pdf in the same directory first.

10.7.4 Transfer to Real Vehicles: Indoor Frames and NED

A Coordinate Transform in One Calculation

Assume motion capture uses east-north-up (ENU).
Control uses north-east-down (NED), with the same origin.

$$p^{NED} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} p^{ENU}$$

$$p^{ENU} = (2,3,1)^T \text{ m} \quad \Rightarrow \quad p^{NED} = (3,2,-1)^T \text{ m}$$

Meaning: 2 m east, 3 m north and 1 m above the shared origin. Add translation if origins differ. Check attitude and velocity conventions too.

10.7.4 Transfer to Real Vehicles: Transfer Stages

Staged Validation from Simulation to Real Vehicles

SITL = software-in-the-loop (SIL in the textbook). HITL = hardware-in-the-loop (HIL).

Stage	Replaced or Added	Evidence Before Advancing
SITL	Software autopilot and virtual vehicles	Correct frames, modes, tracking, and fault branches
HITL	Real autopilot / onboard computer	Timing, cycle time, and link-loss actions meet requirements.
Indoor vehicles	Motion capture or other real localization, real actuators	Static alignment, single-vehicle and small-swarm loops pass.
Outdoor expansion	GNSS-fused pose, real links and site	Tolerates localization degradation, link range, and scale changes.

GNSS = global navigation satellite system

10.7.4 Transfer to Real Vehicles: Observation and Execution Limits

Algorithm Transfer Requires Interface Revalidation

Observations: Replace truth with motion capture, GNSS fusion, or visual estimates. Record quality, update rate, and validity.

Execution: Match target level to flight mode, command authority, and saturation limits.

Runtime: For Simulink / Python → C++ / ROS 2 deployment, check cycle time, dependencies, message types, and shutdown handling.

Equivalence: Compare target outputs for the same input sequence, then use real feedback to check closed-loop errors and failure branches.

Exercises and Discussion

Connecting a Six-Vehicle Motion Capture Experiment

Scenario: 6 vehicles initially run on 2 simulation hosts with 1 centralized controller. Indoor transfer then uses ENU poses from motion capture.

Question 1: What is the control architecture? Does simulation deployment change its information dependencies?

Question 2: How do rigid-body names and ENU poses map to flight-controller IDs and NED targets?

Question 3: After all vehicles take off, which metrics establish formation performance and successful transfer?

Reference Points

Reference Answers and Verifiable Relationships

Question	Reference Answer	Evidence to Check
1 Architecture	Centralized control. Distributed simulation deployment.	Who receives all states and computes all commands?
2 Interfaces	One rigid-body name per ID. Convert ENU to NED, align origins and time.	Static and single-axis positions, message IDs, mode feedback
3 Validation	Tracking and formation meet criteria. Safe transitions and handled timeouts.	Trajectories, minimum separation, state age, anomaly logs

Section Summary

Four Checks for Control Loops and Transfer

Configuration: Simplified / complete describes perception. Point mass / PX4 describes model fidelity.

Interfaces: Python, Simulink, and ROS 2 must maintain consistent identity, frames, time, and modes.

Deployment: Information and computation dependencies define the control architecture. Multiple hosts and mixed models describe other dimensions.

Transfer: Replace localization, execution, and runtime in stages. Use errors and fault responses to verify the control loop.