Chapter Outline

A Learning Path Through the Course, Platform, and Examples

Chapter 1 Introduction

1.1 Concepts and Applications
1.2 History and Trends
1.3 Components and Architecture
1.4 Coordinate Systems and Control Principles
1.5 Assembly and Debugging
1.6 RflySim Toolchain
1.7 Learning Path and Experiments
1.8 Exercises and Discussion

5 Units in This Section

1.7.1 Objective and Structure

1.7.2 Unified Launcher and Resources

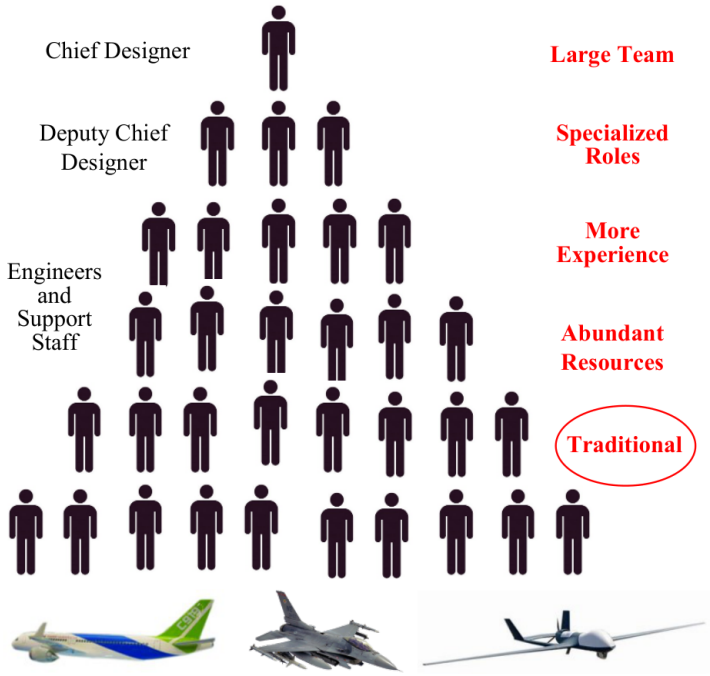
1.7.3 Experimental Methods and Reproduction

1.7.4 Custom Development and AI Assistance

1.7.5 Research and Individual Learning Paths

1.7.1 Objective and Structure: Objectives · Training Engineers

Changing Team Structures Call for System Integration Skills



(a) Large Aerospace Vehicle Design



(b) Micro- and Small-Aircraft Development

Fig. 1.22 Comparison of Demand Trends for Aerospace Engineers

- **Team structure:** large projects divide roles;
small projects often combine several roles.
- **Skills gap:** chief technology officers (CTOs)
and full-stack engineers who coordinate component selection, algorithms, and tests.
- **Training goal:** connect course knowledge,
then specialize in control, perception, or collaboration.

Small teams need coordination across modules; system foundations support specialization.

1.7.1 Objective and Structure: Objectives · Target Skills

System Understanding and Specialized Skills Support Development

System Perspective

Vehicle · Perception · Control · Mission

Specialist Depth

Connect system knowledge horizontally; develop specialist depth vertically

- **Understand the system:** explain observations, algorithms, execution, and feedback.
- **Reproduce experiments:** identify the entry point, requirements, and expected behavior.
- **Develop solutions:** explain how a change affects the result.

Understand the whole system, then build transferable skills through one area of specialization.

1.7.1 Objective and Structure: Objectives · Supporting Resources

Textbook, Platform, and Examples Serve Different Learning Needs

Textbook and Slides

Learn concepts and methods
Understand models and conditions

Unified Launcher

Find tools and resources
Organize the runtime environment

Companion Examples

Connect code with behavior
Reproduce and modify experiments

Learning Question	Start With
Why use this approach?	Textbook, slides, and related chapters
Where do I begin?	Unified launcher, chapter index, and example instructions
How is it implemented?	Example source code and interface documentation

Textbooks explain principles; platforms supply environments; examples link knowledge to code.

1.7 Learning Path and Experiments

1.7.1 Objective and Structure: Chapter Structure

System Development in Four Parts and Ten Chapters

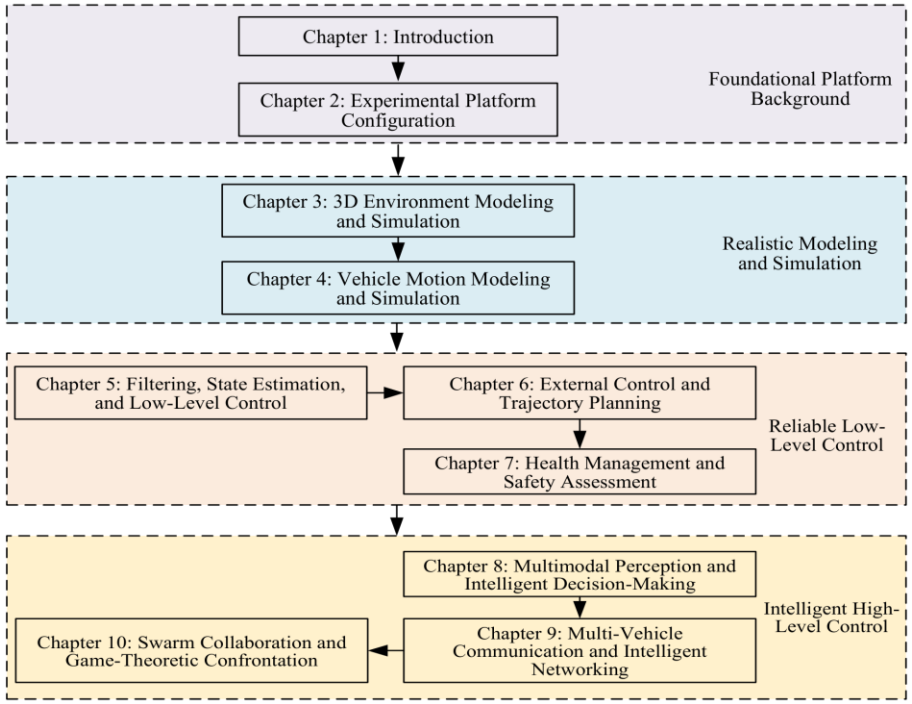


Fig. 1.23 Chapter Structure of the Book

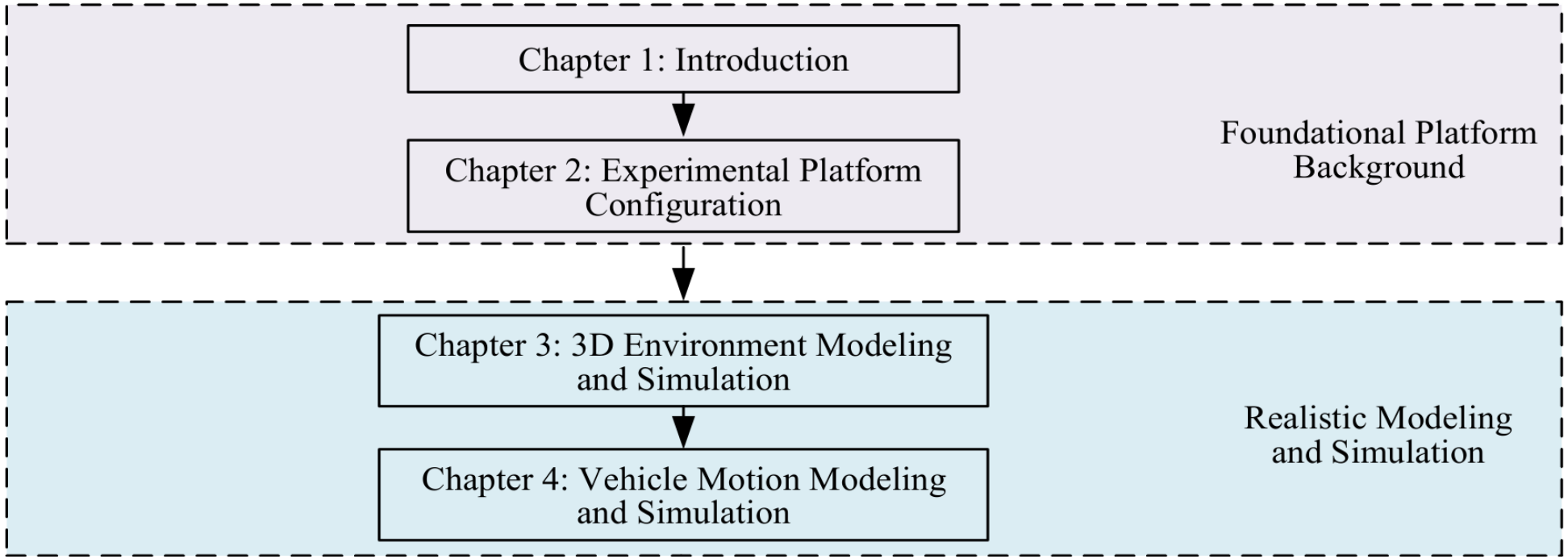
- **Platforms and models:** Chapters 1–4 introduce systems, runtime environments, and simulation objects.
- **Individual vehicles:** Chapters 5–7 develop estimation, control, planning, and safety assessment.
- **Complex tasks:** Chapters 8–10 connect perception and decision-making, networking, and swarm collaboration.

Platforms and models support individual vehicles, perception, decisions, and swarm tasks.

1.7.1 Objective and Structure: Chapter Distribution · Platform and Models

Chapters 1–4: Build the Environment and Simulation Objects

Fig. 1.23 (upper): Foundational Platform Background; Realistic Modeling and Simulation

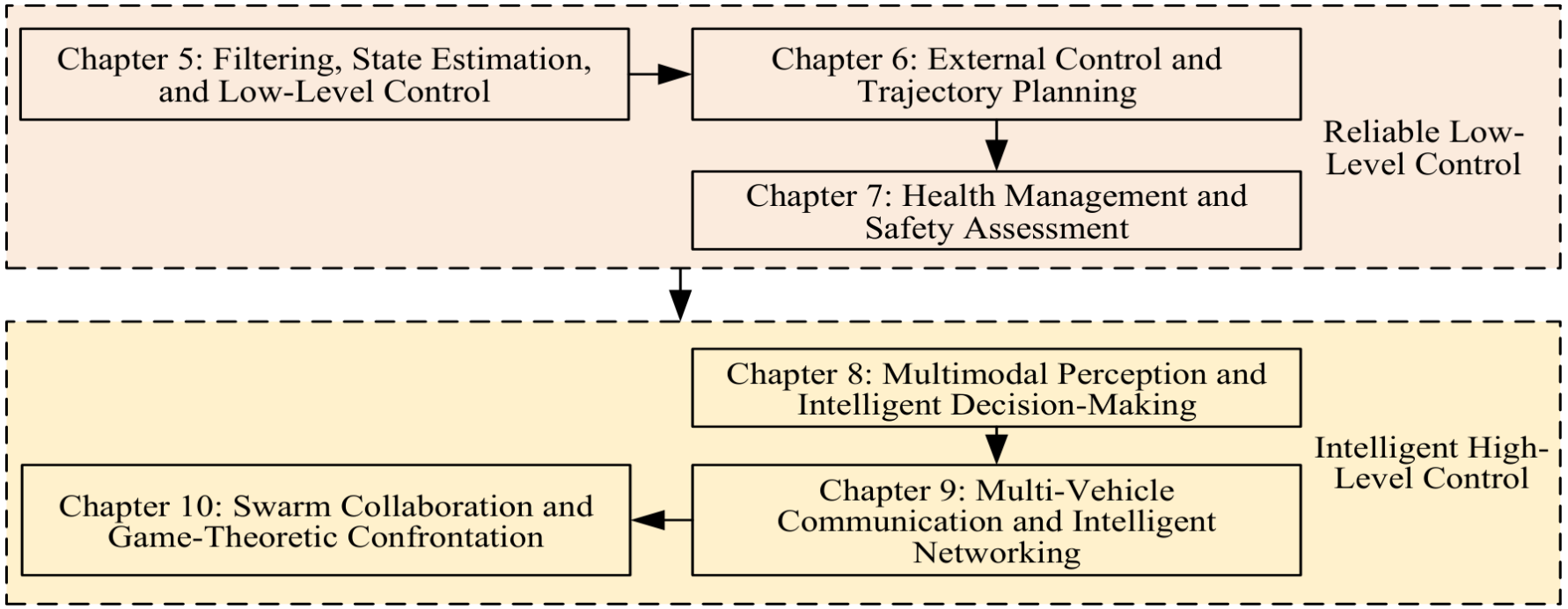


Environment models give observations, motion models responses, for algorithm design and tests.

1.7.1 Objective and Structure: Chapter Distribution · Control and Intelligence

Chapters 5–10: From Reliable Execution to Intelligent Tasks

Fig. 1.23 (lower): Reliable Low-Level Control; Intelligent High-Level Control



Vehicles execute motion; perception and communication extend mission decisions to swarms.

1.7.1 Objective and Structure: Choosing a Direction

Three Learning Paths Share Foundations and Differ in Focus

Shared foundation: system understanding and the experimental environment (Chapters 1–2)

Learning Direction	Main Chapters	First Result to Check
Modeling and Control	Chapters 4, 5, and 6; use Chapter 7 for assessment	Explain how a parameter change affects motion
Perception and Decision-Making	Chapters 3 and 8; Chapter 6 for motion control	Connect image input, detection, and motion commands
Communication and Collaboration	Chapters 9 and 10; fill in individual-vehicle foundations as needed	Explain vehicle states, communication, and collaboration

Shared foundations support different specialties; choose chapters to match your task.

1.7.1 Objective and Structure: Where to Begin

Let the First Task Identify the Knowledge You Need

- **Shared starting point:** understand components, coordinates, and signals; prepare an environment that can run basic simulation.
- **Red-ball tracking:** an onboard camera finds a red ball and guides the aircraft toward it. Reuse the scene and vehicle model, then learn the Python, image coordinates, and velocity commands you need.
- **Study as needed:** Chapter 8 for image processing; Chapters 4–6 for motion; Chapters 9 and 10 for multiple vehicles.
- **Chapter roles:** this section explains how to select and study examples; Chapter 2 covers installation, startup, and interface operations.

Learn the foundations your task needs, then use the results to decide what to study next.

Chapter Outline

A Learning Path Through the Course, Platform, and Examples

Chapter 1 Introduction

1.1 Concepts and Applications
1.2 History and Trends
1.3 Components and Architecture
1.4 Coordinate Systems and Control Principles
1.5 Assembly and Debugging
1.6 RflySim Toolchain
1.7 Learning Path and Experiments
1.8 Exercises and Discussion

5 Units in This Section

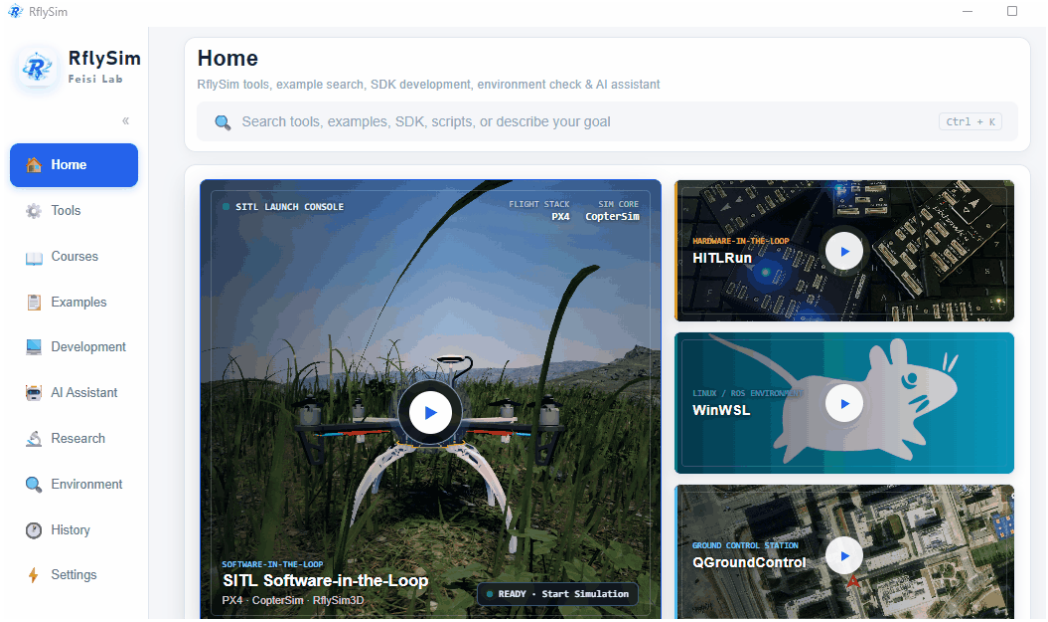
1.7.1 Objective and Structure
1.7.2 Unified Launcher and Resources
1.7.3 Experimental Methods and Reproduction
1.7.4 Custom Development and AI Assistance
1.7.5 Research and Individual Learning Paths

1.7.2 Unified Launcher and Resources: Overview

One Launcher Connects Six Types of Learning Resources

Double-click the RflySim desktop icon.

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



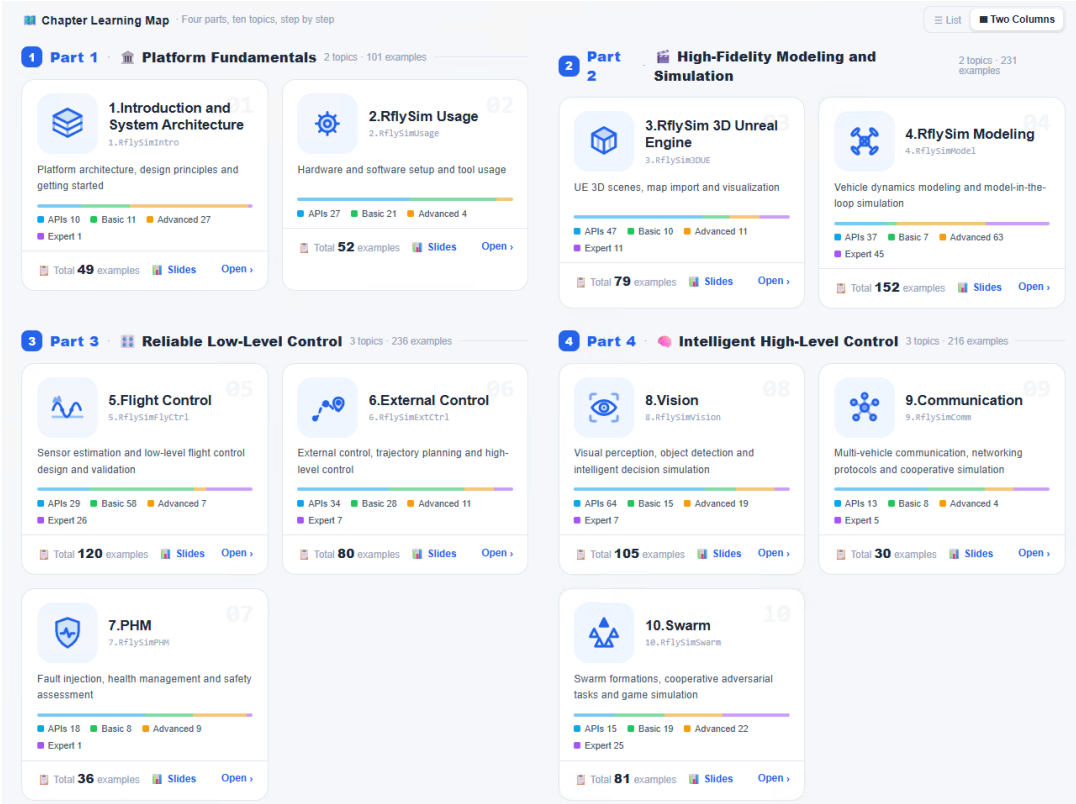
Home	Search and quick launch
Tools	Open development environments
Examples	Choose experiments by chapter
Custom Dev.	Look up interface manuals
AI Assistant	Ask about concepts and methods
Research	Find papers and source code

Choose an entry for your task: course material, tools, examples, and research resources connect.

1.7 Learning Path and Experiments

1.7.2 Unified Launcher and Resources: Examples

The Examples Page Maps Chapters to Experiments



- **Learning map:** four modules match the four parts of the book; each card represents a chapter.
- **Difficulty counts:** cards show example counts by level. Begin with interface and basic examples.
- **Connect to theory:** read Readme.pdf, then explain the behavior using the relevant chapter.

Figure: chapter learning map in Examples

Every experiment links a course chapter to observable behavior.

1.7.2 Unified Launcher and Resources: Chapters and Directories

How Textbook Chapters Map to Example Directories

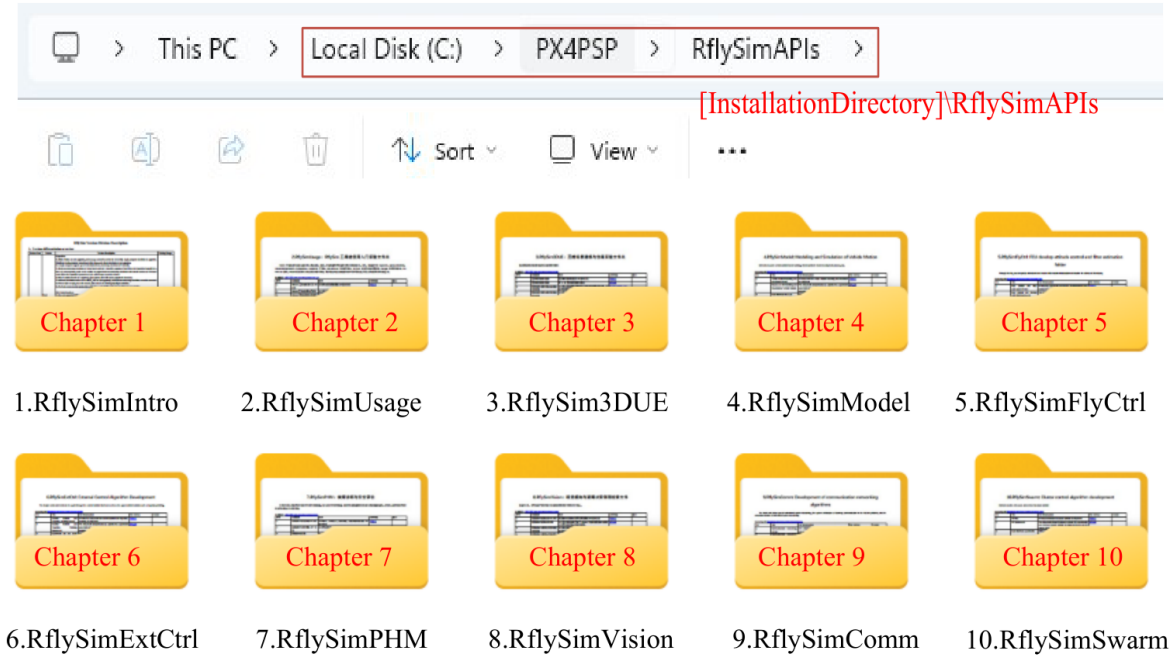


Fig. 1.24 Chapters and Their Example Directories

- **Example root:** [InstallationDirectory]\RflySimAPIs; the default installation is PX4PSP on drive C:.
- **Chapter mapping:** ten numbered directories match the ten chapters. Choose a topic, then an experiment.
- **Connected entry points:** search in Examples; find documents, code, and models in local directories.
- **Before running:** read the instructions for dependencies, startup, and expected behavior.

Chapters locate knowledge; directories locate files; instructions make experiments reproducible.

1.7.2 Unified Launcher and Resources: Foundations and Control

Chapters 1–5: Platform, Models, and Low-Level Control

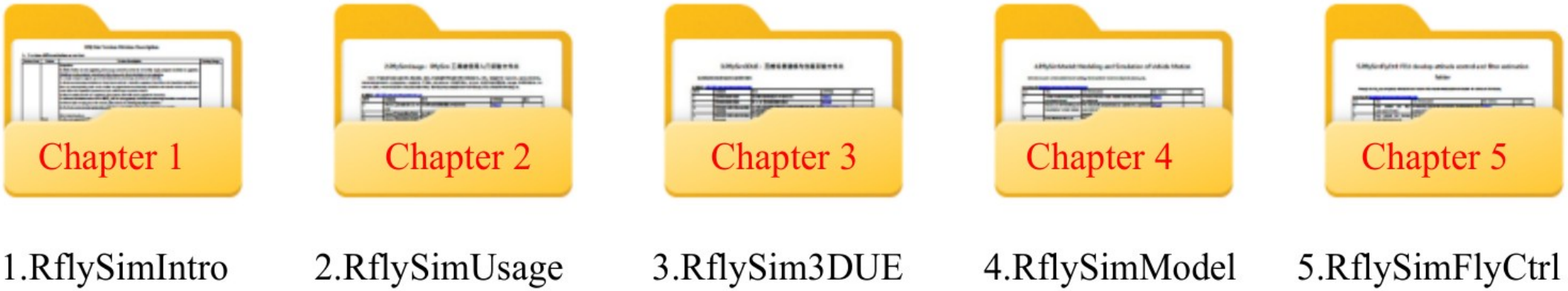


Fig. 1.24 (top row): example directories for Chapters 1–5

- Chapters 1–4: build the runtime environment, 3D scenes, and vehicle motion models.
- Chapter 5: use these models for state estimation and low-level feedback control.

Start with working models and an environment; use examples to learn estimation and feedback.

1.7.2 Unified Launcher and Resources: Missions and Swarms

Chapters 6–10: Mission Capabilities and Multi-Vehicle Collaboration

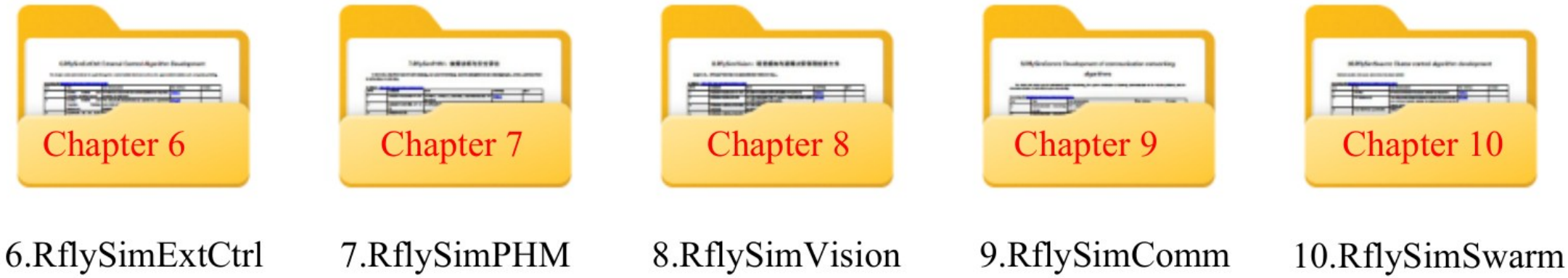


Fig. 1.24 (bottom row): example directories for Chapters 6–10

- Chapters 6–7: individual-vehicle trajectory execution, health management, and safety assessment.
- Chapters 8–10: combine perception and decision-making, networking, and swarm collaboration.

Complex tasks combine motion control, perception and communication; folders locate examples.

1.7.2 Unified Launcher and Resources: Folder Structure

Each Chapter: Three Directory Types and Four Documents

1.RflySimIntro (shared core structure)

0.ApiExps	Interface tutorials
1.BasicExps	Resources and basic tests
2.AdvExps	Advanced development
Intro.pdf	Chapter introduction
Index.pdf	Chapter example index
API.pdf	Chapter interface guide
PPT.pdf	Chapter teaching slides

Table 1.2 Internal Structure of an Example Folder

- **Interface learning:** 0.ApiExps groups tutorials and minimal examples by tool or interface.
- **Basic use:** 1.BasicExps contains introductory material and basic experiments for principles and standard workflows.
- **Advanced development:** 2.AdvExps combines modules or requires more software and hardware. Build foundations first.

Three core folder types are shared; extensions vary by version. Choose a path with Intro.pdf.

1.7.2 Unified Launcher and Resources: Choosing an Example

Choose a Starting Example by Task, Background, and Dependencies

Selection Factor	What to Check	Example Starting Points
Task Goal	The behavior you want to observe or change	Image acquisition, velocity control, target tracking
Prior Knowledge	Choose a starting level for your background	0.ApiExps → 1.BasicExps → 2.AdvExps
Development Method	The language or modeling tool used by the example	Python code or a Simulink model
Operating Requirements	Required software, models, modes, and equipment	Check dependencies and expected results in the instructions

Start with a clear goal, sufficient foundations, and achievable operating requirements.

1.7.2 Unified Launcher and Resources: Resource Roles

Instructions, Interface Manuals, and Source Code Work Together

Resource	Question Answered	When to Read It
Intro.pdf / PPT.pdf	Chapter topics and underlying principles	Starting a chapter or explaining behavior
Examples / Index.pdf	Relevant experiments and their entry points	Choosing an experiment
Example Readme.pdf	Dependencies, startup sequence, and expected results	Before the first run
API.pdf / Interface Manual	Inputs, outputs, units, and calling conventions	Reading code or modifying interface calls
Source Code / Models	How data and algorithms form the workflow	After reproduction, section by section

1.7 Learning Path and Experiments

1.7.2 Unified Launcher and Resources: Home and Tools

Home Returns to Experiments; Tools Opens Development Environments

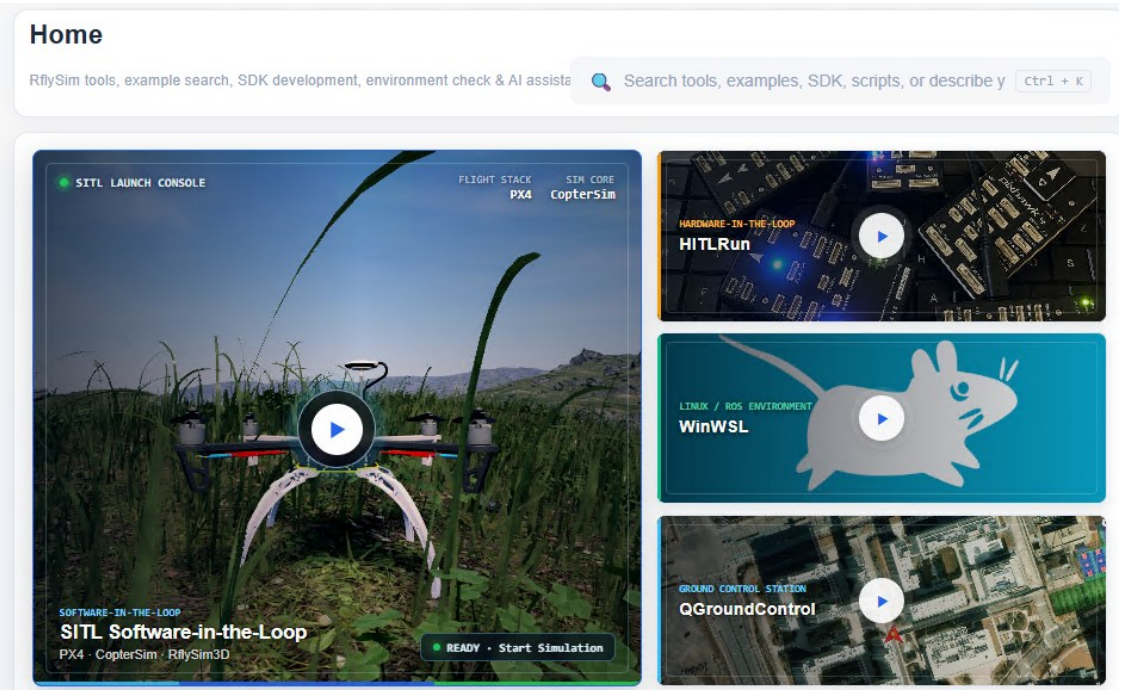


Figure: Home search and four launch entries

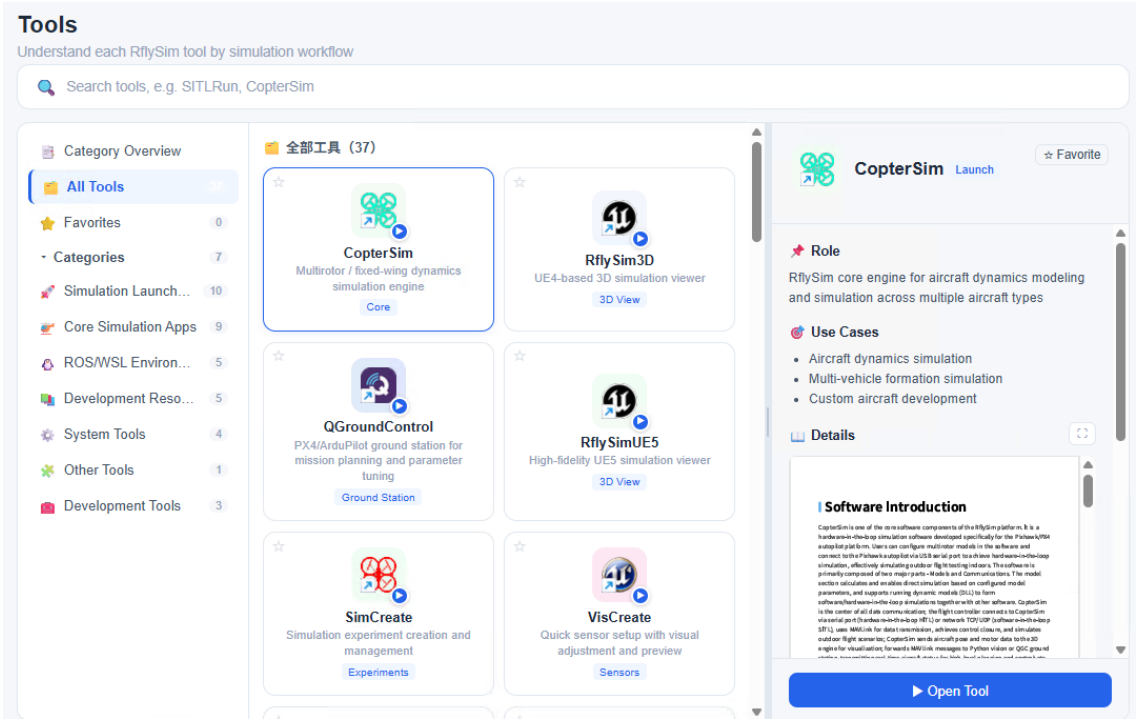


Figure: Tools groups software by simulation workflow

Home speeds up repeat runs; Tools helps combine development tools for a task.

Chapter Outline

A Learning Path Through the Course, Platform, and Examples

Chapter 1 Introduction

1.1 Concepts and Applications
1.2 History and Trends
1.3 Components and Architecture
1.4 Coordinate Systems and Control Principles
1.5 Assembly and Debugging
1.6 RflySim Toolchain
1.7 Learning Path and Experiments
1.8 Exercises and Discussion

5 Units in This Section

1.7.1 Objective and Structure
1.7.2 Unified Launcher and Resources
1.7.3 Experimental Methods and Reproduction
1.7.4 Custom Development and AI Assistance
1.7.5 Research and Individual Learning Paths

1.7.3 Experimental Methods and Reproduction: Method

Set a Goal, Then Reproduce, Explain, and Modify

Define the Goal
Goal and expected behavior

➡

Run the Defaults
Keep the original results

➡

Trace the Data Flow
Input, processing, output

➡

Change One Factor
Predict, then compare

- **Reproduction record:** save the example name, environment versions, key settings, and observed behavior.
- **Explanation:** connect changes on screen to data, algorithms, and control outputs.
- **Comparison:** compare before and after under the same task conditions, then choose the next step.

Reproduce a baseline, explain principles, then change one factor to build understanding.

1.7.3 Experimental Methods and Reproduction: Visual Tracking

Red-Ball Tracking Connects Image Changes to Vehicle Motion

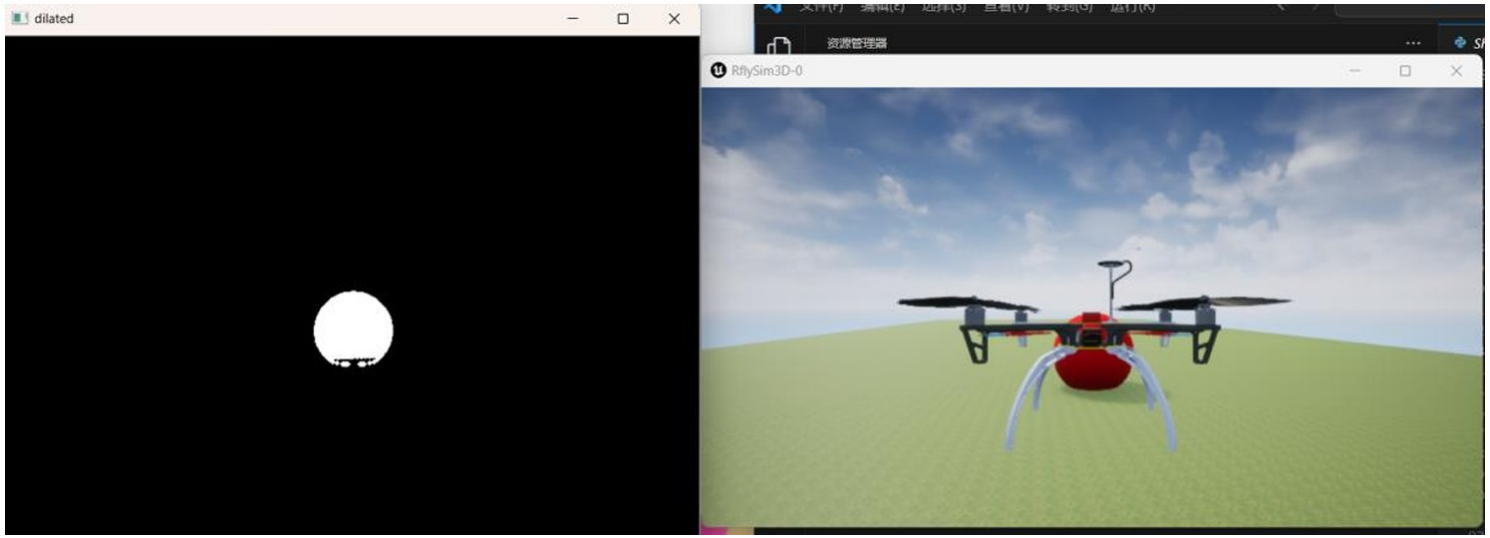


Figure: red pixels shown in white (left); 3D simulation (right)

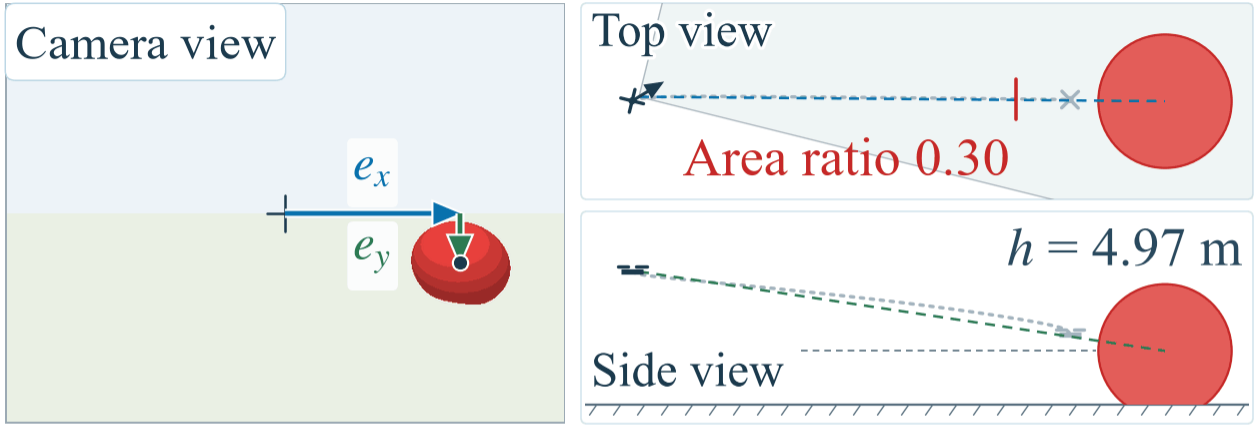
- **Find the target:** rotate clockwise in place while no red ball is visible.
- **Align with the target:** turn for horizontal offset; climb or descend for vertical offset.
- **Approach the target:** stop moving forward when the red-area ratio reaches the threshold.

Trace observations and control feedback signal by signal to explain tracking behavior.

1.7.3 Experimental Methods and Reproduction: Tracking Feedback Loop

Image Feedback Aligns the Vehicle and Brings It to a Gradual Stop

●●○ Detected: align and advance



$e_x = +200$ px → $\dot{\psi} = 0.00375\, e_x = +0.75$ rad/s

$e_y = +56$ px → $v_z = 0.004\, e_y = +0.22$ m/s

Area ratio $0.030 < 0.30$ → $v_x = 0.55$ m/s (desired 2)

Figure: red-ball tracking (offline numerical simulation)

- **Search:** while the ball is unseen, rotate in place until it is detected.
- **Align:** turn for horizontal offset; climb or descend for vertical offset while approaching the target.
- **Area-based stop:** at a ratio of 0.30, set desired forward speed to zero. Commands ramp down; response lag delays the stop.

Zero commands do not stop instantly; rate limits and vehicle response affect where motion stops.

1.7.3 Experimental Methods and Reproduction: Default Run

After Installation, Run Red-Ball Tracking Using Readme.pdf

Location: [InstallationDirectory]\RflySimAPIs\1.RflySimIntro\1.BasicExps\e7_RunPythonProject

1. Start Simulation	Double-click ShootBall3SITL.bat to start SITL simulation
2. Set the Path	Run ReLabPath.py in RflySimAPIs\RflySimSDK
3. Run the Program	Double-click Python38Run.bat; enter python ShootBall3.py
4. End Simulation	Press Enter in the ShootBall3SITL.bat command window to close all programs

- **Expected:** take off at about 5 s, fly left and backward at 15 s, start visual tracking at 25 s, then stop in front of the ball.
- **If blocked:** check Readme.pdf FAQs first, e.g., a missing red ball or failed image acquisition.

- Reproduction record: save parameter defaults, versions, and observed behavior for later explanation and comparison.

1.7.3 Experimental Methods and Reproduction: Input and Feedback

How Image Errors Become Motion Commands

Camera Image → **Target Features** → **Control Calculation** → **Motion Execution**

Raw color image → Center and area → Velocity and yaw rate → Autopilot and vehicle

Vehicle motion changes the next camera observation

Observation and Variable	Relationship in the Code
Target off the image center	Horizontal error controls yaw rate Vertical error controls climb/descent speed
Target region grows larger	Command zero forward velocity at the area-ratio threshold
No valid target detection	Turn to search; output zero commands if no image data is available

Detection extracts image features; control turns errors into motion, producing a new image.

1.7.3 Experimental Methods and Reproduction: One Calculation Step

Calculate Velocity Commands From One Image

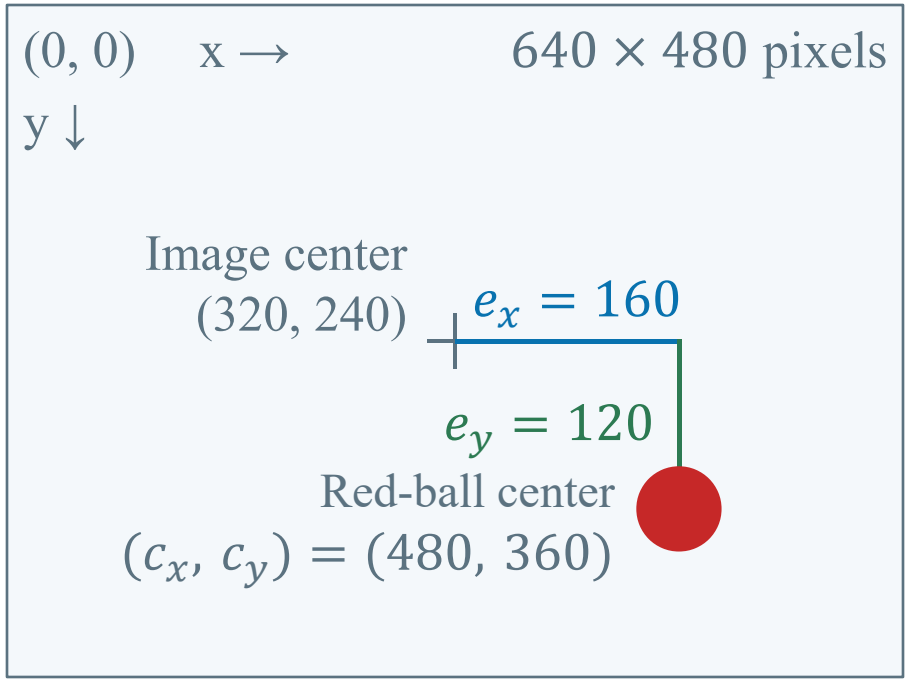


Figure: pixel errors in one image

- **Image errors:** $e_x = c_x - 320$, $e_y = c_y - 240$, in pixels; origin at top left, y positive downward.
- **Control law:** $\dot{\psi} = 0.00375\, e_x$, $v_z = 0.004\, e_y$; if red-area ratio < 0.3, $v_x = 2$ m/s; otherwise 0.
- Directions: in the body forward-right-down frame, $\dot{\psi}$ is yaw rate, positive right; v_z is down and v_x forward.
- **Example:** (480, 360) $\rightarrow e_x = 160$, $e_y = 120 \rightarrow$ $\dot{\psi} = 0.6$ rad/s right turn; $v_z = 0.48$ m/s descent.

Pixel errors times gains give velocity commands; repeating at 30 Hz gradually centers the ball.

1.7.3 Experimental Methods and Reproduction: Controlled Change

Test Your Understanding by Changing One Parameter

Comparison Element	Experiment Plan
Fixed Conditions	Same scene, ball position, camera configuration, and initial state
One Change	Area-ratio threshold max_prop: 0.30 → 0.15
Predicted Behavior	Forward command reaches zero earlier, at about 1.3–1.4 times the distance, not twice
Compare and Explain	Area is roughly inverse-square in distance; compare zero-command distance and final stopping position

Keep task conditions fixed and change one parameter to connect the result to a specific cause.

1.7.3 Experimental Methods and Reproduction: Study as Needed

Match Chapter 1 Resources to Gaps in Your Knowledge

Location: Examples → Chapter 1 (1.RflySimIntro)

Resource Group	What to Learn and When
0.ApiExps **Tool and Interface Tutorials**	Build programming, modeling, system, and hardware knowledge; use these tutorials when your task needs the tool.
1.BasicExps **Basic Resources and Introductory Practice**	Use the textbook series to learn principles; use programming, system-operation, and assembly tutorials to build foundations.
2.AdvExps **Environments and Topic Practice**	Prepare hardware and a dev. environment for the task; study network-port setup, log analysis, and AI assistance.

Links lead to the optional-study index; the main path continues with custom development.

Chapter 1 resources address different knowledge gaps; combine them around your task.

Chapter Outline

A Learning Path Through the Course, Platform, and Examples

Chapter 1 Introduction

1.1 Concepts and Applications
1.2 History and Trends
1.3 Components and Architecture
1.4 Coordinate Systems and Control Principles
1.5 Assembly and Debugging
1.6 RflySim Toolchain
1.7 Learning Path and Experiments
1.8 Exercises and Discussion

5 Units in This Section

1.7.1 Objective and Structure
1.7.2 Unified Launcher and Resources
1.7.3 Experimental Methods and Reproduction
1.7.4 Custom Development and AI Assistance
1.7.5 Research and Individual Learning Paths

1.7.4 Custom Development and AI Assistance: Interfaces and SDK

Custom Development Connects Algorithms to Platform Capabilities

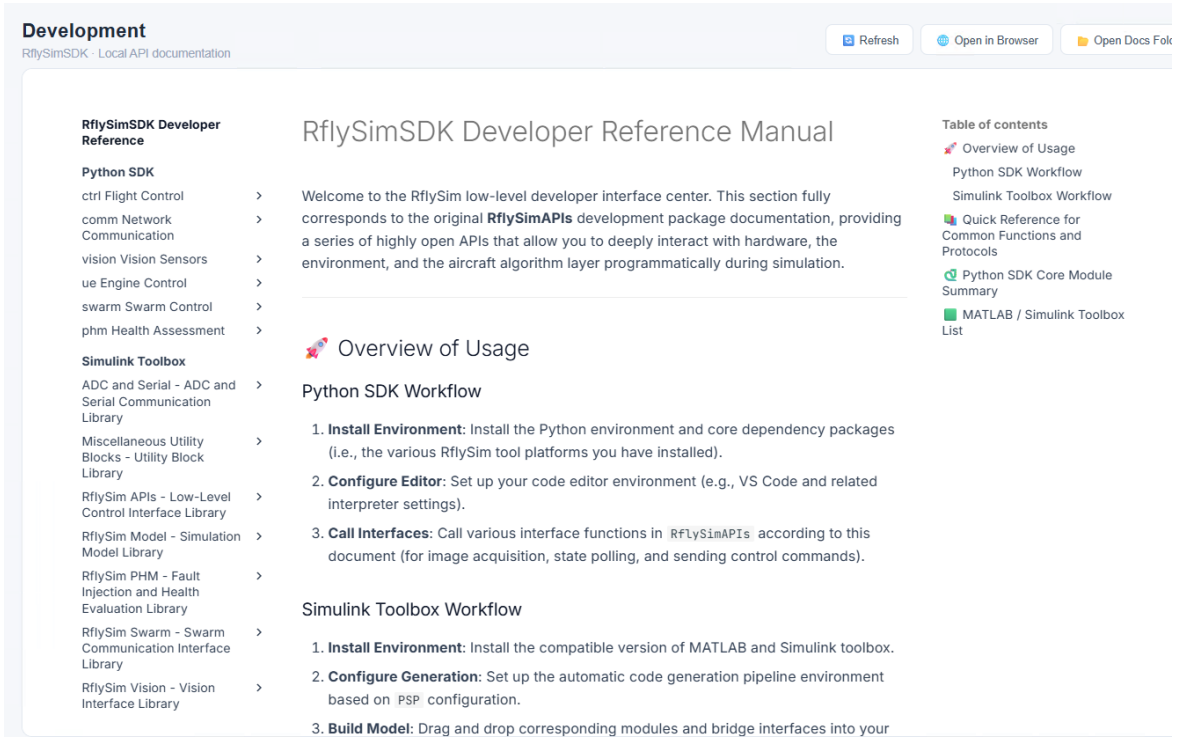


Figure: SDK reference manual under Custom Development

- **Application programming interface (API):** conventions for calling platform functions, e.g., SendVelFRD sends velocity commands.
- **Software development kit (SDK):** groups interfaces by function. Red-ball tracking uses ctrl, vision, and ue modules.
- Study inputs, outputs, units, coordinate frames, and update behavior.

Interfaces define data and calls; the SDK integrates algorithms into complete experiments.

1.7.4 Custom Development and AI Assistance: Where to Extend

Replace One Algorithm Stage in a Working Example

Keep the Baseline
Original example runs

➡

Replace a Module
Detection, planning, or control

➡

Match Interfaces
Data, units, and rates

➡

Add Conditions
Motion, backgrounds, disturbances

- **Perception:** replace detection on the same input images and compare recognition behavior.
- **Control:** retain observation interfaces and change how errors become motion commands.
- **Progressive verification:** finish SITL first, then proceed to HITL and real-vehicle tests as the task requires.

Keep the input-output chain working while replacing or extending algorithms step by step.

1.7.4 Custom Development and AI Assistance: AI Assistant

The AI Assistant Explains Concepts, Finds Resources, and Suggests Plans

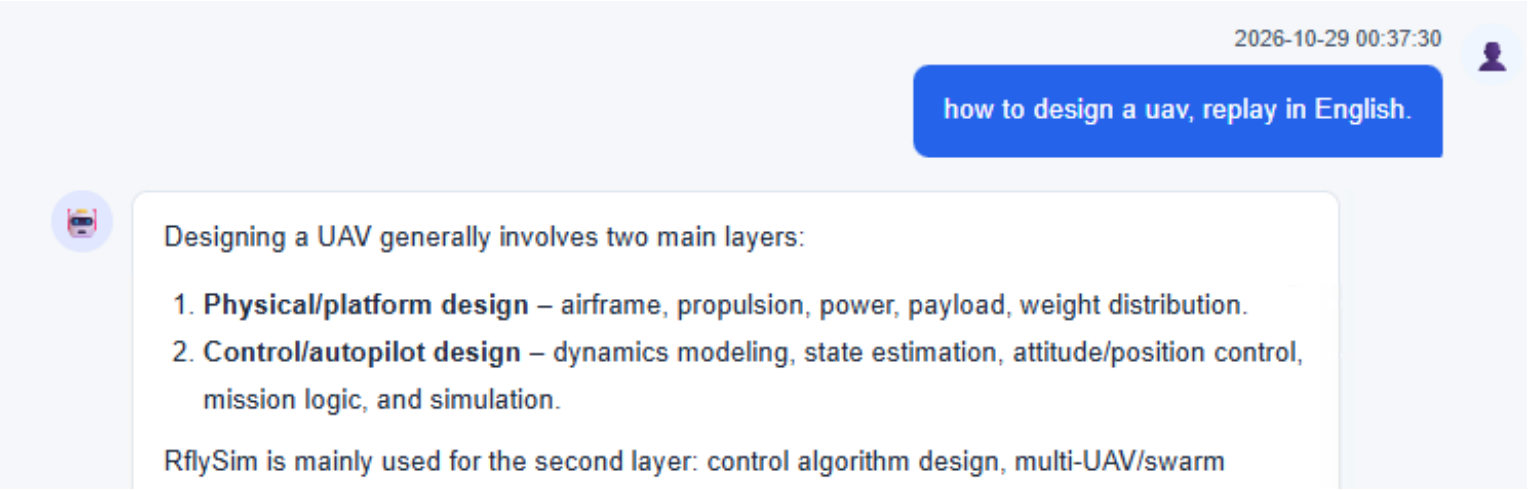


Figure: multi-vehicle example (3.CustExps needs Full Edition or above; Basic Edition supports this lesson)

Ask About Concepts

Explain concepts, interfaces, and code

Find Resources

Locate chapters, examples, and manuals

Suggest a Plan

Break down tasks and experiments

Tie questions to concrete tasks and resources so the assistant can support learning effectively.

1.7.4 Custom Development and AI Assistance: Ask and Verify

Specific Questions Lead to Actionable Next Steps

Example Question

"In e7_RunPythonProject, the onboard camera sees the red ball, but the aircraft keeps turning. How can I tell whether it is searching or correcting horizontal error?"

Add Context	→	Request Evidence	→	Check in a Run
Example, environment, behavior Code and error messages		Relevant document or code branch What to observe next		Is detection valid? Is horizontal error nonzero?

Give the goal, example location and behavior; check answers against code and experiments.

Chapter Outline

A Learning Path Through the Course, Platform, and Examples

Chapter 1 Introduction

1.1 Concepts and Applications
1.2 History and Trends
1.3 Components and Architecture
1.4 Coordinate Systems and Control Principles
1.5 Assembly and Debugging
1.6 RflySim Toolchain
1.7 Learning Path and Experiments
1.8 Exercises and Discussion

5 Units in This Section

1.7.1 Objective and Structure
1.7.2 Unified Launcher and Resources
1.7.3 Experimental Methods and Reproduction
1.7.4 Custom Development and AI Assistance
1.7.5 Research and Individual Learning Paths

1.7.5 Research and Individual Learning Paths: Papers and Code

Research Connects Methods With Runnable Implementations

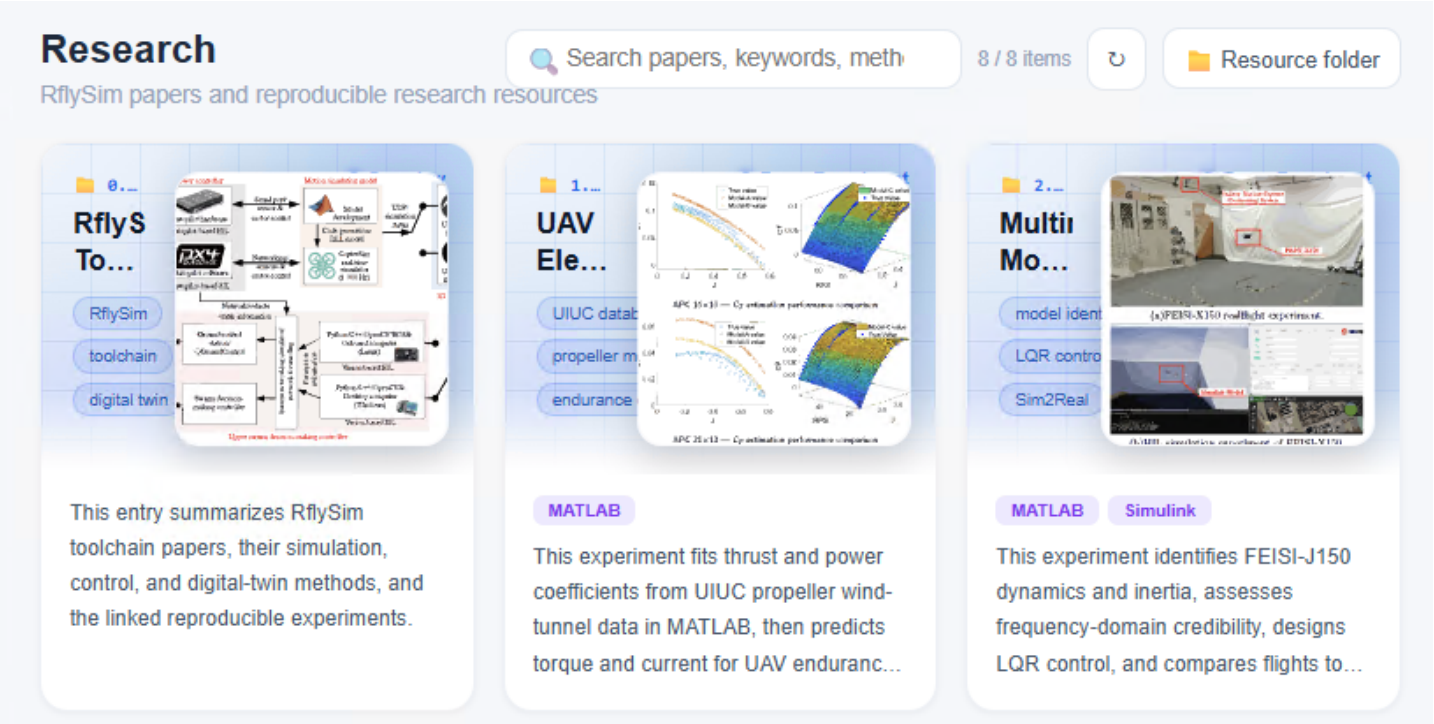


Figure: papers, method descriptions, and open-source projects

- **Find a question:** choose a direction by keyword, method, or tool.
- **Connect to implementation:** plan experiments using the paper, Readme.pdf, and code.
- **Build evidence:** record versions, conditions, results, and differences.

Start with a research question; use papers, code and experiments to learn implementation needs.

1.7.5 Research and Individual Learning Paths: Research Case

Digital Twin: Begin Reproduction With Existing Data

Research → 2.UAV-DT-Control-Sim2Real → 3_Sim2RealCompare/compare_acceleration.m

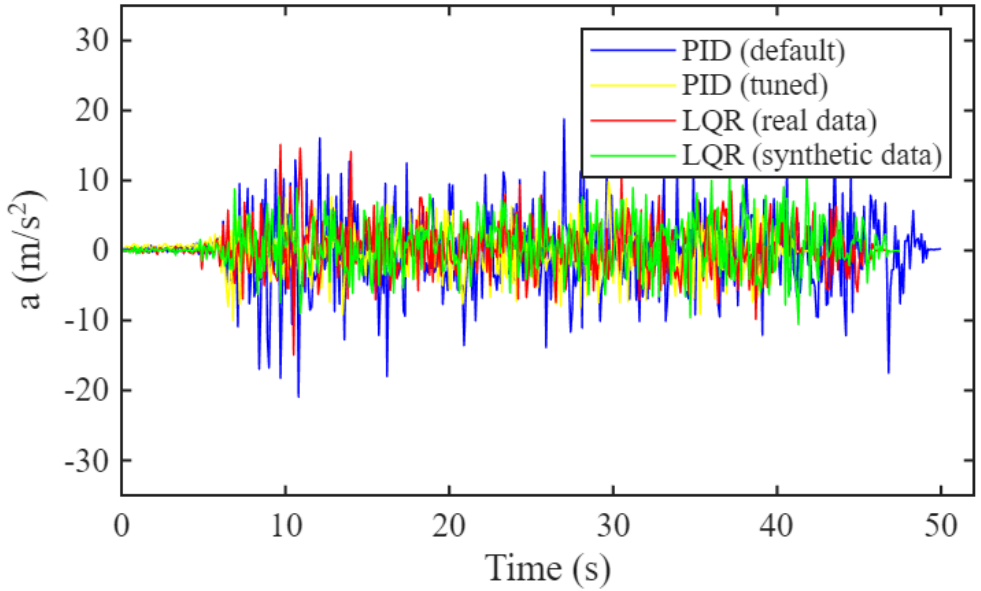


Figure: x-acceleration with default/tuned PID (proportional–integral–derivative) and real/synthetic-data LQR (linear quadratic regulator)

- **Research question:** can digital-twin data (simulation approximating real-vehicle dynamics) replace real-vehicle data for controller design?
- **Supplied data:** first replot records; check time windows and sign processing.
- **Interpretation:** compare acceleration fluctuations with position-tracking error.

Understand research reproduction through the question, model, algorithm, and evidence.

1.7.5 Research and Individual Learning Paths: Advanced Tasks

Advanced Study: Visual Ring Traversal and Multi-Vehicle Simulation

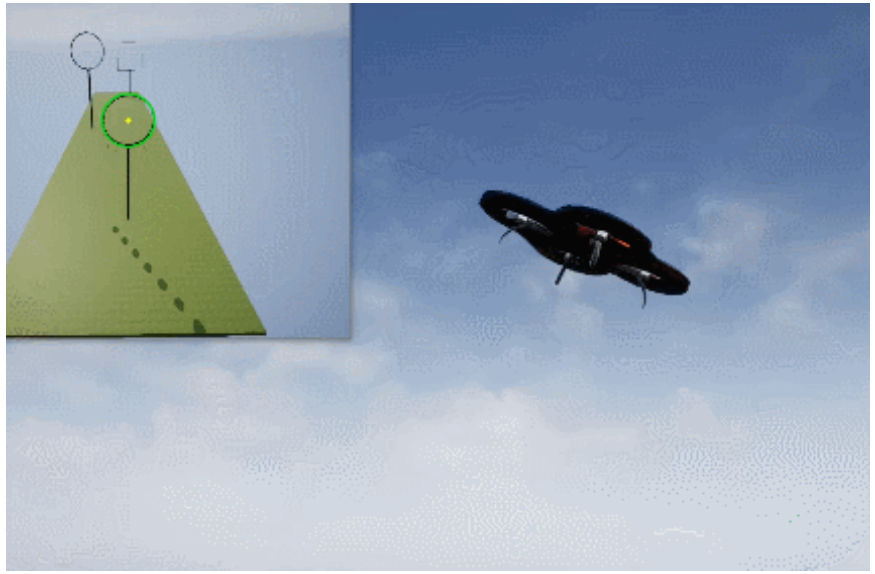


Figure: visual ring traversal (historical demonstration)

Main links: Chapters 6 and 8

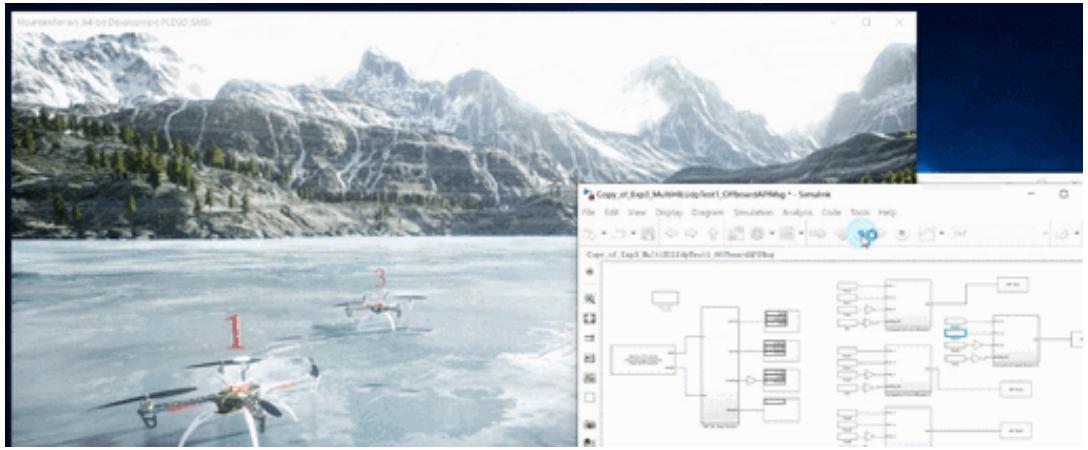


Figure: multi-vehicle mission control (historical demo)

Main links: Chapters 9 and 10

Single-vehicle tasks link perception and control; multi-vehicle APIs aid collaboration research.

1.7.5 Research and Individual Learning Paths: Personal Plan

Plan a Learning Session Around a Task, Example, and Output

Plan Element	Visual-Tracking Example
Learning Task	Understand how an image target affects vehicle motion
Starting Example	Chapter 1 e7_RunPythonProject, with Chapters 6 and 8
Required Foundations	Runtime environment, Python, image coordinates, and control interfaces
Stage Outputs	Original run record, signal diagram, and one-parameter comparison
Next Extension	Choose either detection or control rules for further study

Make the plan concrete: one task, one working baseline, and one change you can explain.

Class Discussion

Use One Example to Check Your Learning Path

- **Resources:** where is a visual-tracking example? Which chapters and documents help?

Guide: Chapter 1 e7; Chapters 6 and 8 as needed, plus Intro.pdf and Readme.pdf.

- **Data:** what are the tracking input, processing, output, and feedback?

Guide: image → red region and errors → velocity commands → motion → new image.

- **Controlled comparison:** what changes, what stays fixed, and what explains the result?

Guide: change area threshold; fix scene, start, and camera; compare zero-command and stopping points.

Explain resource choices, data links and experiment comparisons to choose the next step.

Chapter Experiments: Interface Learning · Programming and Modeling

Tool Tutorials: Programming, Modeling, and Development Foundations

Back to Guide

1.RflySimIntro / 0.ApiExps

Directory / Resource	Learning Content
1.RflySim_SupportBook	Find companion resources and entry points by textbook topic.
2.PX4&Pixhawk_Tutorials	Understand autopilot hardware, firmware, and toolchain relationships.
3.Python_Tutorials	Learn scripting syntax for data and control programming.
4.MATLAB&Simulink_Tutorials	Practice computation and block-diagram modeling; obtain the software separately.
5.MicrosoftVisualStudio_Tutorials	Learn project organization, compilation, and debugging tools.

Use tutorials to fill task-specific gaps, then test your understanding in the example.

Chapter Experiments: Interface Learning · Systems and Resources

Tool Tutorials: Scenes, Systems, and Hardware as Needed

Back to Guide

1.RflySimIntro / 0.ApiExps

Directory / Resource	Learning Content
6.3DsMax&Unreal-Engine_Tutorials	Learn model and scene development; obtain the tools separately.
7.Linux_Tutorials	Learn system foundations: files, processes, and the command line.
8.ROS_Tutorials	Learn Robot Operating System (ROS 1) communication.
9.HardwareSys_Introduction	Connect hardware components with interfaces and wiring.
10.PDF_File_Processing	Combine PDFs with Python and generate a bookmarked document.

Build scene, system, and hardware knowledge gradually as your development task requires.

Chapter Experiments: Basic Use · Principles and Control

Basic Resources: Read Around Vehicle and Control Questions

Back to Guide

1.RflySimIntro / 1.BasicExps (textbook guides; obtain full textbooks and external material separately)

Directory / Resource	Learning Content
e1_MulticopterTheory	Build foundations in dynamics, state estimation, and control.
e2_MulticopterPractice	Explore practice from propulsion design to failsafe protection.
e3_MulticopterDesighFly	Connect flight principles, assembly, debugging, and maintenance.
e4_MulticopterRemoteCtrl	Understand remote-control links; connect to Chapter 6 practice.
e5_SmallAircraftCtrlPractice	Explore fixed-wing design, control, and planning.

Learn the motion and control principles your task needs, then connect them to practice.

Chapter Experiments: Basic Use · Programming and Assembly

Basic Practice: Programming, Systems, Assembly, and Debugging

Back to Guide

1.RflySimIntro / 1.BasicExps

Directory / Resource	Learning Content
e6_OverviewPython	Learn the editor and syntax; practice parameter changes.
e7_RunPythonProject	Run red-ball tracking and understand the vision-control feedback loop.
e8_IntroductionLinux	Learn Linux systems, distributions, and file organization.
e9_LinuxCommandLine	Practice paths and commands; check the environment with examples.
e10.DroneAssTutorial	Prepare the equipment, then study assembly and debugging with Section 1.5.

Coding and system basics support examples; assembly and debugging connect to real vehicles.

Chapter Experiments: Advanced Development · Hardware Selection

Hardware Guide: Input Devices, Computers, and Vehicles

Back to Guide

1.RflySimIntro / 2.AdvExps

Directory / Resource	Learning Content
e1.RCIntro	Learn RC channels, protocols, and calibration; connect operator input to vehicle response.
e2.FCUIntro	Learn autopilot hardware, firmware, and connections; prepare for HITL experiments.
e4.ComputerIntro	Understand host-computer requirements for scenes, vision, and multi-vehicle tasks.
e5.UAVConnfig	Use MiniQuad150 configuration resources to connect the real vehicle, firmware, and simulation.

Choose control devices, host computers, and vehicle configurations for the task.

Chapter Experiments: Advanced Development · Development Environments

Configure Interpreters, Compilers, and WSL for the Example

Back to Guide

1.RflySimIntro / 2.AdvExps

Directory / Resource	Learning Content
e3.PythonConfig	Configure the interpreter and dependencies; check the environment with a Python example.
e6.VisualStudioInstall	Install C++ compilation components for code builds and model development.
e7.WslGUI	Run graphics and communication examples in Windows Subsystem for Linux (WSL).
e8.WslVsCode	Connect VS Code to WSL for editing, running, and debugging.

Judge environment readiness by whether the corresponding program actually runs.

Chapter Experiments: Advanced Development · Computing and Communication

Compare Computation Time and Image-Transfer Latency

Back to Guide

1.RflySimIntro / 2.AdvExps

Graphics processing unit (GPU) items: Windows 11 22H2 or later, NVIDIA GPU and driver.

Directory / Resource	Learning Content
e11.WSL2_GPUAccConfig	Compare CPU and GPU computation time. Requires platform v4.10 or later and computing dependencies.
e12.ROS2SharedMem	Compare image-transfer latency, frame rate, and resource use. Requires WSL 2, ROS 2, and a graphical environment.
e13.WinWSL2-GPU	Deploy a separate GPU setup; verify computation. Obtain the incremental image separately; prepare the system as instructed.

Check computing and communication gains with metrics such as execution time and frame rate.

Chapter Experiments: Advanced Development · AI Assistance

AI Tools: From Resource Search to Natural-Language Control

Back to Guide

1.RflySimIntro / 2.AdvExps

Directory / Resource	Learning Content
e14.LocalLLMDepUse	Use local Ollama models for natural language; send simulation actions, e.g., takeoff and landing. Needs GPU setup, models and Python dependencies; obtain MATLAB/Simulink R2022b+ separately.
e15.RflySim_Skills	Use companion skills to find examples and interfaces; study flight-controller code and simulation scenes. Requires a skills-capable AI host and Python; installation depends on the host.

Use AI tools for concrete tasks and check results against documentation and experiments.

Chapter Experiments: Advanced Development · Ethernet Configuration

Ethernet: From One Flight Controller to Multiple Vehicles

Back to Guide

1.RflySimIntro / 2.AdvExps / e9.ConfigTools

Directory / Resource	Learning Content
e1.FCUETH_NetConfig	Use an Ethernet flight controller and network devices; configure and verify single-vehicle HITL communication.
e2.FeisiSwarmBox_Config	Batch-configure flight-controller networks; requires Full Edition and a dedicated swarm simulation unit.
e3.FCUETH_ModifyParam	Networked flight controller: set airframe parameters; check that the vehicle configuration matches the simulation.

Verify one flight-controller connection, then add vehicles as equipment allows.

Chapter Experiments: Advanced Development · Data Acquisition

Begin Log Analysis by Obtaining Usable Data

Back to Guide

1.RflySimIntro / 2.AdvExps / e10.Log-GetAnalysis

Directory / Resource	Learning Content
1.LogGet	Download ULog files via QGroundControl (QGC); requires a flight controller with recorded logs.
9.SwarmLogGet	Run multi-vehicle SITL and obtain a separate ULog for each vehicle.
10.UAVsStatusRTGet_Py	Read and print states and ground truth with Python as a data source for custom recording.
11.UAVsStatusRTGet_Mat	Collect two-vehicle states in the Simulink workspace; R2022b+; real hardware needs two flight controllers.

Offline logs support later review; real-time data supports custom observation and recording.

Chapter Experiments: Advanced Development · Offline Analysis

Choose Log Tools for the Analysis Question

Back to Guide

1.RflySimIntro / 2.AdvExps / e10.Log-GetAnalysis

Directory / Resource	Learning Content
2.LogAnalysis_Wed	Browse states with Flight Review (internet required).
3.LogAnalysis_CMD	Convert and filter log messages with pyulog.
4.LogAnalysis_Mat	Compare curves in MATLAB (R2022b or later).
5.LogAnalysis_Py	Parse data and plot attitude curves with Python.
6.LogAnalysis_PlotJuggler	Explore curves interactively; obtain PlotJuggler separately.

Chapter Experiments: Advanced Development · Custom Logging

Record New Data and Control Variables in the Experiment

Back to Guide

1.RflySimIntro / 2.AdvExps / e10.Log-GetAnalysis

First two need MATLAB/Simulink, a PX4 code-generation setup, a flight controller, and an RC kit.

Directory / Resource	Learning Content
8.CusmsgSDLog	Generate firmware, record and read custom data; requires an SD card for storage.
12.CusuORBmsgLog	Record custom controller-message variables; requires compilation, upload, and HITL verification.
7.CopterSimLogging **Link to another chapter**	Go to 2.RflySimUsage / 1.BasicExps / e13_Log-Get to learn CopterSim ground-truth logging.

Keep interpretable data for new algorithms so results can be reviewed and compared.

Section Summary

Move Between Knowledge, Tools, and Experiments

- **Course map:** four parts and ten chapters locate the knowledge your task needs.
- **Unified launcher:** Home, Tools, Examples, Custom Development, AI Assistant, and Research each serve a purpose.
- **Experimental method:** run defaults, trace the data flow, and compare a change to one factor.
- **Further study:** fill gaps in your specialization, then connect to custom development and research.
- **Common pitfalls:** directory numbers are not a required sequence; area ratio is not calibrated distance; check assistant answers against Readme.pdf and source code.

Choose a path by the question; build an example baseline; explain and change it to build skills.