# 附录A：Python代码实现

本附录提供了无人水面艇可视化系统的关键Python代码实现细节。

## A.1 可视化后端服务

后端使用Flask和Flask-SocketIO构建，提供实时数据传输和WebSocket通信。

### A.1.1 服务器启动与端口管理

def find\_available\_port(start\_port, max\_attempts=MAX\_PORT\_ATTEMPTS):  
 """  
 Find an available port starting from start\_port.  
   
 Args:  
 start\_port (int): The port to start searching from  
 max\_attempts (int): Maximum number of ports to try  
   
 Returns:  
 int: An available port number  
   
 Raises:  
 RuntimeError: If no available port is found after max\_attempts  
 """  
 for port in range(start\_port, start\_port + max\_attempts):  
 sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 try:  
 sock.bind(('0.0.0.0', port))  
 sock.close()  
 return port  
 except OSError:  
 print(f"Port {port} is already in use, trying next port...")  
 continue  
   
 # If we get here, we couldn't find an available port  
 raise RuntimeError(f"Could not find an available port after {max\_attempts} attempts")  
  
def get\_port\_from\_env\_or\_config():  
 """  
 Get port from environment variable, config, or use default.  
   
 Returns:  
 int: The port number to use  
 """  
 # Try to get port from environment variable  
 try:  
 env\_port = os.environ.get('USV\_VISUALIZATION\_PORT')  
 if env\_port is not None:  
 return int(env\_port)  
 except (ValueError, TypeError):  
 print("Warning: Invalid port in environment variable, using default")  
   
 # Try to get port from config file  
 config = load\_config()  
 if config and 'visualization' in config and 'port' in config['visualization']:  
 try:  
 return int(config['visualization']['port'])  
 except (ValueError, TypeError):  
 print("Warning: Invalid port in config file, using default")  
   
 # Return default port  
 return DEFAULT\_PORT  
  
# Main entry point  
if \_\_name\_\_ == '\_\_main\_\_':  
 try:  
 # Get starting port from environment or config  
 default\_port = get\_port\_from\_env\_or\_config()  
   
 # Find available port  
 port = find\_available\_port(default\_port)  
   
 # Save port to file for external applications  
 save\_port\_to\_file(port)  
   
 # Log information  
 if port != default\_port:  
 print(f"Default port {default\_port} was in use, using port {port} instead")  
 else:  
 print(f"Starting server on port {port}")  
   
 # Start server  
 socketio.run(app, debug=True, host='0.0.0.0', port=port)  
 except Exception as e:  
 print(f"Error starting server: {e}")  
 sys.exit(1)

### A.1.2 USV模拟器初始化

def initialize\_simulator(scenario='waypoint', controller\_type='pid'):  
 """Initialize the USV simulator with specified scenario and controller."""  
 global simulator, current\_scenario, current\_controller  
   
 config = load\_config()  
 if not config:  
 return False  
   
 with simulation\_lock:  
 # Create simulator  
 simulator = USVSimulator(  
 dt=config['simulation']['dt'],  
 simulation\_time=config['simulation']['simulation\_time'],  
 x0=np.array(config['simulation']['initial\_state']),  
 vessel\_params=config['vessel'],  
 env\_params=config['environment'],  
 controller\_type=controller\_type,  
 controller\_params=config['controllers'][controller\_type]  
 )  
   
 # Set reference trajectory  
 simulator.set\_reference\_trajectory(config['scenarios'][scenario]['reference\_trajectory'])  
   
 current\_scenario = scenario  
 current\_controller = controller\_type  
   
 return True

### A.1.3 WebSocket事件处理

@socketio.on('start\_simulation')  
def handle\_start\_simulation(data):  
 """Start the simulation with the specified parameters."""  
 global simulation\_thread, simulation\_running, simulation\_paused  
   
 # Stop any existing simulation  
 if simulation\_running:  
 simulation\_running = False  
 if simulation\_thread:  
 simulation\_thread.join()  
   
 # Extract parameters  
 scenario = data.get('scenario', 'waypoint')  
 controller = data.get('controller', 'pid')  
   
 # Initialize simulator  
 success = initialize\_simulator(scenario, controller)  
 if not success:  
 emit('simulation\_status', {'status': 'error', 'message': 'Failed to initialize simulator'})  
 return  
   
 # Start simulation thread  
 simulation\_paused = False  
 simulation\_thread = threading.Thread(target=simulation\_worker)  
 simulation\_thread.daemon = True  
 simulation\_thread.start()  
   
 emit('simulation\_status', {'status': 'started'})

## A.2 前端可视化实现

前端使用Three.js进行3D可视化和Chart.js进行数据图表展示。

### A.2.1 3D场景初始化

function initVisualization() {  
 // Create scene  
 scene = new THREE.Scene();  
 scene.background = new THREE.Color(0x87CEEB); // Sky blue background  
   
 // Create camera  
 const container = document.getElementById('visualization-container');  
 const aspect = container.clientWidth / container.clientHeight;  
 camera = new THREE.PerspectiveCamera(75, aspect, 0.1, 1000);  
 camera.position.set(30, 30, 30);  
 camera.lookAt(0, 0, 0);  
   
 // Create renderer  
 renderer = new THREE.WebGLRenderer({ antialias: true });  
 renderer.setSize(container.clientWidth, container.clientHeight);  
 container.appendChild(renderer.domElement);  
   
 // Create camera controls  
 controls = new THREE.OrbitControls(camera, renderer.domElement);  
 controls.enableDamping = true;  
 controls.dampingFactor = 0.05;  
   
 // Add ambient light  
 const ambientLight = new THREE.AmbientLight(0xffffff, 0.5);  
 scene.add(ambientLight);  
   
 // Add directional light (sunlight)  
 const directionalLight = new THREE.DirectionalLight(0xffffff, 0.8);  
 directionalLight.position.set(10, 20, 10);  
 scene.add(directionalLight);  
   
 // Create water surface, grid, axes, USV model, etc.  
 createWaterSurface();  
 createGrid();  
 createAxes();  
 createUSV();  
 createReferenceMarker();  
 createPathLine();  
   
 // Start animation loop  
 animate();  
}

### A.2.2 实时数据更新

function updateVisualization(data) {  
 // Update USV position and orientation  
 usv.position.x = data.state.x;  
 usv.position.z = data.state.y; // Y in simulation is Z in Three.js  
 usv.rotation.y = -data.state.heading; // Negative to match coordinate system  
   
 // Update reference marker  
 reference.position.x = data.reference.x;  
 reference.position.z = data.reference.y;  
   
 // Update trajectory  
 trajectoryPoints.push(new THREE.Vector3(data.state.x, 0.1, data.state.y));  
 updatePathLine();  
   
 // Update camera based on view mode  
 updateCamera();  
}  
  
function updateCharts(data) {  
 // Update data arrays  
 timeData.push(data.time);  
 xData.push(data.state.x);  
 yData.push(data.state.y);  
 headingData.push(data.state.heading \* 180 / Math.PI); // Convert to degrees  
 surgeData.push(data.state.surge);  
 swayData.push(data.state.sway);  
 yawRateData.push(data.state.yaw\_rate);  
 thrustData.push(data.control.thrust);  
 momentData.push(data.control.moment);  
   
 // Update trajectory data  
 usvTrajectoryData.push({x: data.state.x, y: data.state.y});  
 referenceTrajectoryData.push({x: data.reference.x, y: data.reference.y});  
   
 // Update charts  
 positionChart.update();  
 velocityChart.update();  
 controlChart.update();  
 trajectoryChart.update();  
}

## A.3 脚本工具

### A.3.1 跨平台启动脚本

def main():  
 # Parse command line arguments  
 parser = argparse.ArgumentParser(description="USV Visualization Server Runner")  
 parser.add\_argument("-p", "--port", type=int, default=DEFAULT\_PORT,   
 help=f"Port to run the server on (default: {DEFAULT\_PORT})")  
 parser.add\_argument("-d", "--debug", action="store\_true", help="Run in debug mode")  
 parser.add\_argument("-b", "--browser", action="store\_true", help="Automatically open browser")  
 args = parser.parse\_args()  
  
 # Set environment variables  
 os.environ["USV\_VISUALIZATION\_PORT"] = str(args.port)  
  
 # Get script directory  
 script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))  
 app\_path = os.path.join(script\_dir, 'app.py')  
 port\_file = os.path.join(script\_dir, 'port.txt')  
  
 # Run the server  
 if args.debug:  
 # Run in foreground with debug output  
 try:  
 subprocess.run([sys.executable, app\_path])  
 except KeyboardInterrupt:  
 print("\nServer stopped by user")  
 else:  
 # Run in background and capture output  
 log\_file = os.path.join(script\_dir, 'server.log')  
   
 with open(log\_file, 'w') as log:  
 proc = subprocess.Popen(  
 [sys.executable, app\_path],  
 stdout=log,  
 stderr=log,  
 universal\_newlines=True  
 )  
   
 # Wait for server to start  
 time.sleep(2)  
   
 # Check if port.txt exists to get actual port  
 port = args.port  
 if os.path.exists(port\_file):  
 actual\_port = read\_port\_file(port\_file)  
 if actual\_port is not None and actual\_port != args.port:  
 port = actual\_port  
 print(f"Note: Server is using port {port} instead of {args.port}")  
   
 # Open browser if requested  
 if args.browser:  
 url = f"http://localhost:{port}"  
 print("Opening browser...")  
 webbrowser.open(url)

### A.3.2 测试工具

def run\_test\_with\_occupied\_port():  
 """  
 Test that the server can find an available port when the default port is occupied.  
 """  
 # Get the path to app.py  
 current\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))  
 app\_path = os.path.join(current\_dir, 'app.py')  
 port\_file = os.path.join(current\_dir, 'port.txt')  
   
 # Remove port.txt if it exists  
 if os.path.exists(port\_file):  
 os.remove(port\_file)  
   
 # Occupy the default port  
 sock = occupy\_port(DEFAULT\_PORT)  
 if not sock:  
 print(f"Could not occupy port {DEFAULT\_PORT} for testing, test skipped")  
 return False  
   
 try:  
 # Run app.py in a subprocess  
 print(f"Running {app\_path}...")  
 proc = subprocess.Popen([sys.executable, app\_path],   
 stdout=subprocess.PIPE,   
 stderr=subprocess.PIPE,  
 universal\_newlines=True)  
   
 # Wait for the port file to be created  
 port = None  
 for \_ in range(TIMEOUT):  
 port = read\_port\_file(port\_file)  
 if port is not None:  
 break  
 time.sleep(1)  
   
 # Check if server started on a different port  
 if port is not None and port != DEFAULT\_PORT:  
 print(f"Success: Server started on port {port} instead of default {DEFAULT\_PORT}")  
 success = True  
 else:  
 print(f"Error: Server should have used a different port than {DEFAULT\_PORT}")  
 success = False  
   
 return success  
 finally:  
 # Clean up  
 if sock:  
 sock.close()  
   
 # Terminate the server process  
 if 'proc' in locals() and proc:  
 proc.terminate()