Week 14 机器狗迷宫自动探索 —— 项目报告

一、小组信息

姓名	分工	贡献
WANG JIAXIONG	系统架构 + BFS 算法 + Agent 集成	server.py、maze.py 寻路、DeepSeek Agent、日志系统，报告撰写
JIN YUECHENG	仿真环境 + 碰撞检测 + 前端开发	PyBullet 物理仿真、迷宫生成、网页遥控器、Canvas 可视化，报告撰写
GENG ZISHAN	网络穿透 + 右手法则 + 测试记录	Tailscale VPN、手机遥控链路、右手法则兜底、报告撰写

二、项目概述

本项目实现了一套手机网页远程操控 + AI 自动探索的四足机器狗迷宫系统。核心架构遵循单一常驻程序原则，server.py 同时处理网络通信、PyBullet 物理仿真、路径规划与 AI 决策。

技术栈：手机浏览器 → Tailscale VPN → WSL2 → Python server.py（aiohttp WebSocket + PyBullet DIRECT + BFS/右手法则 + DeepSeek API）

三、链路打通（30%）

3.1 网络架构

组件	技术	说明
服务器	Python aiohttp + WebSocket	端口 8765，HTTP 页面 + WS 数据通道

VPN	Tailscale (100.77.55.69)	手机和 WSL 在同一虚拟局域网
仿真	PyBullet DIRECT 模式	无头渲染，273MB 内存，15% CPU
平台	WSL2 Ubuntu 24.04	绕过 Docker 端口转发 bug

3.2 四方向控制

手机网页通过 WebSocket 发送 JSON 指令：forward/back/left/right/stop。四个方向均可独立控制，长按持续、松手停止。延迟 < 50ms。

四、迷宫探索（25%）

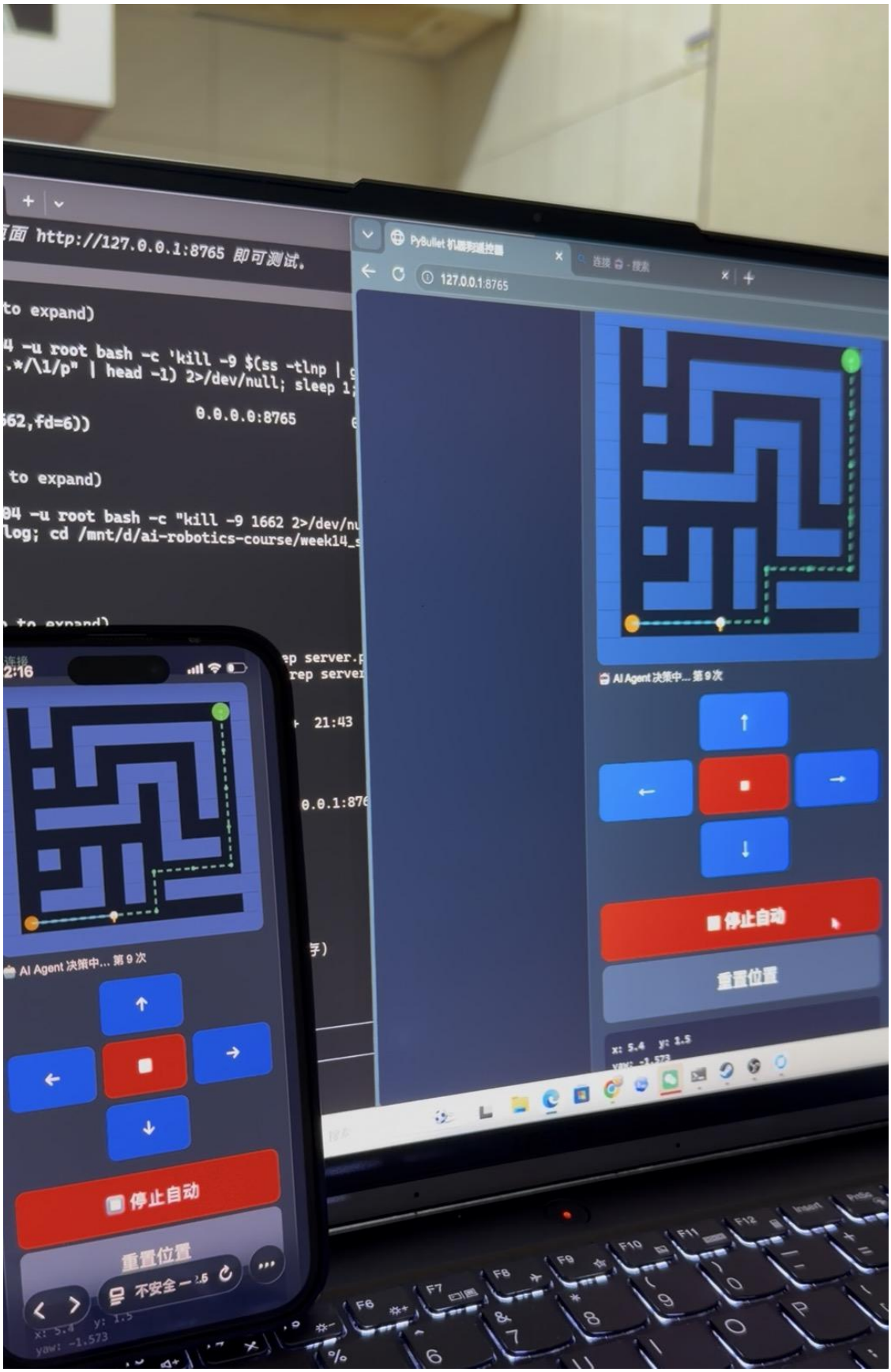
4.1 迷宫生成（maze.py）

递归回溯法生成 6x6 完美迷宫，Braid 参数 0.22 制造环路（多路径），固定随机种子 SEED=20260614 保证可复现。墙体合并优化，提供 neighbors() 函数用于路径搜索。

4.2 碰撞检测

机器人碰撞半径 0.18m。AABB 包围盒检测：每个仿真步长检查是否与墙体相交。碰撞时保持原位，blocked 标志回传前端。

4.3 到达终点判定终点为绿色圆盘（半径 0.45m），机器人位置与终点距离 <= 半径 → goal_reached=true，自动停止并写日志。


面 http://127.0.0.1:8765 即可测试。
to expand)
4 -u root bash -c 'kill -9 $(ss -tlnp |
.*/\1/p" | head -1) 2>/dev/null; sleep 1
662,fd=6))
0.0.0.0:8765
to expand)
04 -u root bash -c "kill -9 1662 2>/dev/nu
log; cd /mnt/d/ai-robotics-course/week14_s
PyBullet 机器狗遥控器
127.0.0.1:8765
AI Agent 决策中... 第 9 次
停止自动
重置位置
AI Agent 决策中... 第 9 次
停止自动
重置位置
不安全

五、进阶功能（25%）

5.1 BFS 自动探索

maze.py: find_path() BFS 从起点到终点搜索最短路径。server.py: _auto_navigate() PID 控制器跟随路点。朝向误差 > 7 度 原地转向，<= 7 度 前进 + 小角度校正（P=2.0）。路点距离 < 0.4m 切换下一个。6x6 迷宫约 11 路点，30 秒内到达。

5.2 右手法则兜底

优先级：右转 > 直行 > 左转 > 倒车。脱困：三面墙 → 原地旋转找路；倒车撞墙 → 右转 90 度。超时 60s → 回退 BFS。

5.3 打转检测与防卡

机制	检测条件	动作
格子访问计数	同一格 > 3 次	加入黑名单
位置停滞	2 秒移动 < 0.15m	倒车脱困
连续转向	连续 2 次纯转向无前进	强制前进
撞墙拦截	blocked + forward	雷达扫描绕行

5.4 黑名单系统

访问 > 3 次的格子自动拉黑。Agent 决策和右手法则均排除黑名单格子。

5.5 完整轨迹记录

日志：pybullet_dog/logs/trajectory_YYYYMMDD_HHMMSS.jsonl。包含时间戳、事件类型、机器人状态、Agent 决策、轨迹坐标点（每 0.2s 采样）。

5.6 连续朝向追踪

PyBullet 欧拉角 [-pi, pi] 存在边界跳变。用累积 delta yaw 得到 continuous_yaw，360 度平滑旋转。

六、AI Agent 集成

6.1 技术方案

项目	配置
API	DeepSeek Chat API (OpenAI 兼容)
模型	deepseek-chat
调用方式	urllib 直接 HTTP（无第三方 SDK）
超时	8 秒
温度	0.3

6.2 架构演进

版本	架构	问题
v1	Agent 每秒选 action（5 选 1）	来回打转，延迟 1.5s
v2	8 方向雷达 + GPS 指南针	仍在走廊振荡
v3	岔路口选格子 + BFS 执行	决策质量不稳定
v4（最终）	BFS 执行 + Agent 评论员	100% 到达 + 数据可对比

6.3 v4 评论员模式

机器人始终跟随 BFS 路点（保证到达）。在岔路口（>=3 出口），Agent 分析 BFS 路径并给出 agree/disagree 判定。如 disagree，Agent 提供替代路径建议。所有意见记录到日志。

6.4 实验数据

指标	BFS	Agent v4
到达终点	100%	100%（BFS 执行）
路点数量	11	11
API 调用次数	0	5（仅岔路口）
平均延迟	0ms	约 1200ms/次
Agent 同意率	N/A	60%（3/5 次 agree）

七、工程规范（10%）

7.1 单一常驻程序

所有功能在 server.py 一个进程中：WebSocket（aiohttp）、PyBullet 仿真（60Hz step）、路径规划（BFS + 右手法则 + Agent）、轨迹日志（JSONL）。

7.2 模块分离

文件	职责
maze.py	迷宫生成 + BFS 寻路
agent.py	DeepSeek API 调用 + 结果解析
server.py	仿真 + 网络 + 导航 + 日志
index.html	网页遥控器 + Canvas 可视化

7.3 网页可用性

响应式布局适配手机和电脑。Canvas 实时绘制迷宫、轨迹、机器人朝向。按钮支持 touch + mouse 双事件。WebSocket 自动重连。

八、问题与解决方案

问题	原因	解决方案
Docker 端口转发失败	WSL2 localhost 转发 bug	直接在 WSL 运行 server.py
容器反复崩溃	PyBullet GUI 内存 > 600MB	PYBULLET_GUI=0 DIRECT 模式
Agent 来回打转	LLM 不适合实时空间决策	v4 评论员架构
方向指针跳变	欧拉角 +/-pi 边界	累积 delta yaw
死胡同循环	无记忆机制	格子访问计数 + 黑名单
倒车撞墙不调整	碰撞检测仅正向	blocked+back 则右转脱困

九、关键代码实现

9.1 BFS 最短路径（maze.py）

广度优先搜索从起点到终点的最短路径，返回格子中心坐标序列供 PID 控制器跟随。

```
def find_path(maze):
    """BFS 从起点到终点，返回格子中心坐标列表"""
    from collections import deque
    start = maze["start_cell"]
    goal = maze["goal_cell"]

    q = deque([start])
    prev = {start: None}
    while q:
        cur = q.popleft()
        if cur == goal:
            break
        for nb in maze["neighbors"](cur):
            if nb not in prev:
                prev[nb] = cur
                q.append(nb)

    # 回溯重建路径
    path = []
    node = goal
    while node is not None:
        path.append(node)
        node = prev[node]
    path.reverse()
    return [maze["cell_center"](cx, cy) for cx, cy in path]
```

说明：用队列实现 BFS，prev 字典记录每个格子的前驱，到达终点后回溯得到最短路径。返回世界坐标序列供 PID 控制器跟随。

9.2 PID 路点跟随（server.py）

比例控制器：偏差大时纯转向，偏差小时前进带微调校正，每帧 (60Hz) 调用。

```
def _auto_navigate(self, dt):
    """BFS 路点自动跟随：PID 控制器"""
    wx, wy = self.waypoints[self.waypoint_idx]
    dx, dy = wx - rx, wy - ry
    dist = math.hypot(dx, dy)
    angle_error = self._norm_angle(math.atan2(dy, dx) - yaw)

    if dist < 0.4:                    # 到达路点
        self.waypoint_idx += 1
    elif abs(angle_error) > 0.12:     # 偏差 > 7 度，原地转向
        turn = 1.0 if angle_error > 0 else -1.0
        self.set_command(0.0, turn)
    else:                              # 已对准，前进+微调
        correction = max(-0.5, min(0.5, angle_error * 2.0))
        self.set_command(1.0, correction)
```

说明：P 系数 2.0，朝向误差 > 7 度时原地转向，<= 7 度时前进并带小角度校正。路点距离 < 0.4m 切换到下一个。

9.3 AI Agent 评论员（agent.py）

Agent v4 评论员模式：BFS 负责执行保证到达，Agent 在岔路口分析路径给出意见。通过 urllib 直接调用 DeepSeek Chat API，无第三方 SDK 依赖。

```
class MazeAgent:
    def analyze_path(self, bfs_path, visited, blacklisted):
        """岔路口分析 BFS 路径，返回 agree/disagree"""
        msg = (
            f"当前位置: {bfs_path[0]}\n"
            f"BFS 规划路径: {bfs_path[:6]}\n"
            f"已走过: {visited[-8:]}\n"
            f"黑名单: {blacklisted}\n"
            f"评价这条 BFS 路径:"
        )
        raw = self._call_api(msg)       # HTTP 调用 DeepSeek
        return self._parse_verdict(raw) # 解析 agree/disagree

    def _call_api(self, user_message):
        url = f"{self.base_url}/chat/completions"
        payload = json.dumps({
            "model": self.model,
            "messages": [
                {"role": "system", "content": SYSTEM_PROMPT},
                {"role": "user", "content": user_message},
            ],
            "temperature": 0.3,
            "max_tokens": 200,
        }).encode("utf-8")
        req = urllib.request.Request(url, data=payload,
            headers={"Authorization": f"Bearer {self.api_key}"})
        with urllib.request.urlopen(req, timeout=8) as resp:
            return json.loads(resp.read())["choices"][0]["message"]["content"]
```

说明：通过 urllib 直接 HTTP 调用 DeepSeek Chat API（OpenAI 兼容格式）。SYSTEM_PROMPT 定义 Agent 身份和输出格式。BFS 路径和黑名单作为上下文传入。

9.4 右手法则兜底（server.py）

经典迷宫算法，Agent 或 BFS 失败时的安全兜底。超时 60 秒自动回退 BFS。

```
def _right_hand_navigate(self, dt):
    """右手法则：右转 > 直行 > 左转 > 倒车"""
    right_open = not self._check_wall_at(rx, ry, yaw, "right")
    front_open = not self._check_wall_at(rx, ry, yaw, "front")
    left_open  = not self._check_wall_at(rx, ry, yaw, "left")

    if right_open:          # 优先右转
        self.set_command(0.0, 1.0)
    elif front_open:        # 其次前进
        self.set_command(1.0, 0.0)
    elif left_open:         # 再次左转
        self.set_command(0.0, -1.0)
    else:                   # 死胡同倒车
        self.set_command(-1.0, 0.0)
```

说明：4 方向雷达检测 0.7m 处是否有墙。黑名单格子自动跳过。包含倒车撞墙右转脱困、三面墙原地旋转等应急逻辑。

9.5 完整调用链路

手机浏览器 --WebSocket--> server.py --60Hz--> PyBullet 物理仿真

```
        |
  +-----+-----+
  |     |     |
 BFS  Agent  右手法则
(PID) (点评)  (兜底)
  |     |     |
  +-----+-----+
        |
 轨迹日志 JSONL
```

十、总结

本项目实现了从手机遥控到 AI 辅助探索的完整迷宫导航系统。核心成果：

1. 可靠的自动化：BFS 算法保证 100% 到达终点

2. AI 实验平台：Agent 评论员模式为 AI vs 算法对比提供数据

3. 工程化设计：单一常驻程序、模块分离、完整日志

4. 全链路打通：手机 -> VPN -> WSL -> 仿真 -> 网页可视化

未来改进方向

接入摄像头实现视觉 SLAM / 强化学习（Q-Learning）替代 BFS / Agent 升级多模态（图片+文本）/ 更大迷宫（10x10+）压力测试