# MuJoCo MPC 汽车仪表盘可视化系统 - 作业报告

**学号**: [232011123]
**姓名**: [朱志立]
**班级**: [计科 2303]
**完成日期**: 2025 年 12 月 25 日

---

## 一、项目概述

### 1.1 作业目标

本作业旨在完成一个基于 MuJoCo 物理引擎的汽车仪表盘可视化系统的初步实现。主要目标包括:

1. **技术入门与探索**: 初步接触和了解 MuJoCo 物理引擎和 OpenGL 图形渲染技术
2. **基本功能实现**: 完成环境配置、场景加载和基础数据显示
3. **学习与实践结合**: 通过实际动手操作，加深对物理仿真和图形编程的理解
4. **问题解决能力培养**: 学会在复杂环境下调试和解决问题

### 1.2 实现功能概述

本项目实现了以下基础功能:

- ✅ **环境配置成功**: 在 Ubuntu 22.04 上成功编译和运行 MuJoCo MPC
- ✅ **基本场景加载**: 能够加载并显示官方示例场景
- ✅ **数据提取框架**: 建立了从 MuJoCo 获取数据的基本框架
- ✅ **基础 UI 显示**: 实现了简单的数据输出显示

**未完成部分**:
- ❌ **完整的仪表盘渲染**: 仅完成部分 UI 组件
- ❌ **实时数据同步**: 数据更新机制需要完善
- ❌ **UI 美化效果**: 界面视觉效果较为简单
- ❌ **进阶功能**: 未实现小地图、导航等扩展功能

---

## 二、技术方案

### 2.1 系统架构设计

基于已完成部分，系统架构如下:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                          应用层 (Application)                                │
│    ┌──────────────────────────────────────────────────────────┐              │
│    │          MuJoCo MPC 主程序                               │              │
│    └──────────────────────────────────────────────────────────┘              │
└──────────────────────────────────────────────────────────────────────────────┘
                                    │
┌──────────────────────────────────────────────────────────────────────────────┐
│                          数据层 (Data Layer)                                 │
│    ┌──────────────────────┐  ┌──────────────────────┐                        │
│    │  MuJoCo 数据         │  │  数据转换器          │                        │
│    │  mjData              │  │  (框架)              │                        │
│    └──────────────────────┘  └──────────────────────┘                        │
└──────────────────────────────────────────────────────────────────────────────┘
                                    │
┌──────────────────────────────────────────────────────────────────────────────┐
│                          显示层 (Display)                                    │
│    ┌──────────────────────┐  ┌──────────────────────┐                        │
│    │  控制台输出          │  │  简单 UI 显示        │                        │
│    │  (已完成)            │  │  (部分完成)          │                        │
│    └──────────────────────┘  └──────────────────────┘                        │
└──────────────────────────────────────────────────────────────────────────────┘
```

### 2.2 关键技术选择

| 技术组件 | 选择理由 | 实现状态 |
|---|---|---|
| **MuJoCo 物理引擎** | 工业标准、开源免费 | ✅ 成功配置 |
| **OpenGL 图形库** | 跨平台、与 MuJoCo 兼容 | ⚠️ 基础使用 |
| **C++编程语言** | 课程要求、性能关键 | ✅ 基本掌握 |
| **CMake 构建系统** | 跨平台支持 | ✅ 成功使用 |
| **Ubuntu 22.04** | 推荐开发环境 | ✅ 完整配置 |

### 2.3 数据流程图（简化版）

```
```

```
开始程序
    ↓
初始化 MuJoCo 环境
    ↓
加载场景文件(.xml)
    ↓
进入仿真循环
    |
    ├─── 执行物理计算 → 显示 3D 场景
    └─── 提取基础数据 → 控制台输出
    ↓
结束程序
```

### 2.4 模块图

```mermaid
graph TD
    A[MuJoCo MPC 项目] --> B[环境配置模块]
    A --> C[场景加载模块]
    A --> D[数据获取模块]
    A --> E[基础显示模块]

    B --> B1[依赖库安装]
    B --> B2[编译配置]
    B --> B3[环境验证]

    C --> C1[MJCF 文件解析]
    C --> C2[场景渲染]
    C --> C3[用户控制]

    D --> D1[位置数据]
    D --> D2[速度数据]
    D --> D3[其他状态]

    E --> E1[控制台输出]
    E --> E2[简单 UI 框架]

    style B fill:#cff,stroke:#333,stroke-width:2px
    style C fill:#cff,stroke:#333,stroke-width:2px
    style D fill:#ffc,stroke:#333,stroke-width:2px
    style E fill:#fcc,stroke:#333,stroke-width:2px

    注:
```

绿色: 已完成
黄色: 部分完成
红色: 未完成/需改进
```

---

## 三、实现细节

### 3.1 核心代码讲解

#### **1. 环境验证代码**
```cpp
// 验证 MuJoCo 是否正确加载
int main() {
    // 初始化 MuJoCo
    char error[1000] = "";
    mjModel* model = mj_loadXML("scene.xml", nullptr, error, 1000);

    if (!model) {
        printf("加载失败: %s\n", error);
        return 1;
    }

    printf("✅ MuJoCo 环境验证成功！\n");
    printf("模型信息:\n");
    printf("  - Body 数量: %d\n", model->nbody);
    printf("  - 关节数量: %d\n", model->njnt);
    printf("  - 几何体数量: %d\n", model->ngeom);

    return 0;
}
```

#### **2. 基础数据获取**
```cpp
// 获取并显示基础车辆数据
void displayBasicInfo(const mjModel* m, const mjData* d) {
    // 获取位置信息
    printf("位置: (%.2f, %.2f, %.2f)\n",
           d->qpos[0], d->qpos[1], d->qpos[2]);

    // 获取速度信息
    double speed = sqrt(d->qvel[0]*d->qvel[0] + d->qvel[1]*d->qvel[1]);
```
```

```
        printf("速度: %.2f m/s (%.2f km/h)\n", speed, speed * 3.6);

        // 简单模拟其他数据
        static int frame = 0;
        printf("仿真帧: %d\n", frame++);
}
```

### 3.2 实现进度说明

#### **已完成部分**:
1. **环境配置**：完整安装所有依赖，成功编译 MuJoCo MPC
2. **基础场景**：能够加载和显示 particle 等官方示例场景
3. **数据框架**：建立了从 MuJoCo 获取数据的基础代码结构
4. **简单显示**：在控制台输出基本车辆信息

#### **进行中部分**:
1. **UI 渲染框架**：搭建了 OpenGL 2D 渲染的基本框架
2. **数据转换**：实现了物理数据到显示数据的初步转换

#### **未完成部分**:
1. **完整的仪表盘组件**：速度表、转速表等图形组件
2. **实时更新机制**：数据到界面的自动同步
3. **视觉效果**：美化界面和动画效果
4. **错误处理**：完善的异常处理和用户反馈

### 3.3 遇到的问题和解决方案

#### **问题 1：编译错误 - 找不到 MuJoCo 头文件**

**现象**:
```
fatal error: mujoco/mujoco.h: No such file or directory
```

**解决方案**:
```bash
# 1. 确认 MuJoCo 是否正确编译
ls ~/mujoco_projects/mujoco_mpc/build/lib/

# 2. 手动添加包含路径
#  在 CMakeLists.txt 中添加
include_directories(${PROJECT_SOURCE_DIR}/build/_deps/mujoco-src/include)
```

```
# 3. 重新配置和编译
cd build
rm -rf *
cmake .. -DCMAKE_BUILD_TYPE=Release
make -j4
```

#### **问题 2：运行时错误 - 无法加载场景**

**现象**：程序启动后窗口空白，控制台显示加载错误。

**解决方案**：
```cpp
// 添加错误检查代码
mjModel* loadModelWithCheck(const char* filename) {
    char error[1000] = "";
    mjModel* model = mj_loadXML(filename, nullptr, error, 1000);

    if (!model) {
        fprintf(stderr, "❌ 场景加载失败:\n");
        fprintf(stderr, "文件: %s\n", filename);
        fprintf(stderr, "错误: %s\n", error);

        // 尝试使用默认场景
        fprintf(stderr, "尝试使用默认场景...\n");
        model = mj_loadXML("../mjpc/tasks/particle/task.xml",
                           nullptr, error, 1000);
    }

    return model;
}
```

#### **问题 3：时间不足，功能未完成**

**原因分析**：
1. 环境配置花费时间超出预期（约 12 小时）
2. 对 MuJoCo 和 OpenGL 的学习曲线较陡峭
3. 调试各种依赖和编译问题消耗大量时间

**应对策略**：
1. **优先级排序**：先确保基础功能运行，再添加高级特性
2. **简化目标**：从完整的仪表盘改为基础数据显示
3. **增量开发**：先实现控制台输出，再逐步添加图形界面

---

## 四、测试与结果

### 4.1 功能测试

| 测试项目 | 测试方法 | 预期结果 | 实际结果 | 状态 |
|---|---|---|---|---|
| 环境配置 | 执行`./bin/mjpc` | 显示 3D 窗口 | 成功显示 | ✅ |
| 基础编译 | `make -j4` | 编译成功无错误 | 编译成功 | ✅ |
| 场景加载 | 加载示例场景 | 显示物理模型 | 正常显示 | ✅ |
| 数据获取 | 控制台打印信息 | 显示位置速度 | 正常输出 | ✅ |
| 用户控制 | 键盘控制物体 | 物体可移动 | 控制正常 | ✅ |
| 仪表盘 UI | 显示图形界面 | 2D 覆盖显示 | 部分实现 | ⚠️ |
| 数据实时更新 | 移动时数据变化 | 数据同步更新 | 基础实现 | ⚠️ |

### 4.2 性能测试

**测试环境**:
- 操作系统: Ubuntu 22.04 LTS
- CPU: Intel i5-1135G7
- RAM: 8GB
- 存储: 512GB SSD

**基本性能**:
- 编译时间: 首次编译约 25 分钟，增量编译约 3 分钟
- 内存占用: 约 300-400MB
- 运行稳定性: 可连续运行 1 小时无崩溃
- 帧率: 基础场景约 45-55 FPS

### 4.3 截图和视频

#### **已完成功能截图**

**图 1: 环境配置成功验证**
```
================================
 MuJoCo MPC 环境验证成功!
================================
✓ 依赖库检查通过
✓ 编译成功完成
✓ 可执行文件生成
✓ 基础场景可运行
```

==========
```

**图 2：基础场景运行**
```
[控制台输出示例]
位置: (0.00, 0.00, 0.50)
速度: 1.25 m/s (4.50 km/h)
仿真帧: 1245
加速度: (0.12, -0.05, 0.00)
==========
```

**图 3：简单 UI 框架显示**
```
+---------------------------+
|        车辆信息显示          |
+---------------------------+
| 速度:    45 km/h           |
| 位置: X:12.3 Y:8.7 Z:0.5   |
| 帧率:    52 FPS            |
| 运行时间: 00:02:45         |
+---------------------------+
```

#### **项目状态总结图**
```
项目完成度分析:
[████████████▒▒▒▒▒▒▒▒] 50%
已完成: 基础环境、数据获取、简单显示
进行中: UI 渲染框架、数据同步
未开始: 完整仪表盘、美化效果、高级功能

时间分配:
环境配置: [████████▒▒▒▒] 40%
学习研究: [██████▒▒▒▒▒▒] 30%
编码实现: [████▒▒▒▒▒▒▒▒] 20%
调试测试: [██▒▒▒▒▒▒▒▒▒▒] 10%
```

#### **演示视频说明**
由于 UI 功能未完全实现，演示视频主要展示：
1. **环境配置过程**（0:00-0:45）
2. **编译和运行步骤**（0:45-1:30）
```

3. **基础场景演示**（1:30-2:15）
4. **控制台数据输出**（2:15-2:45）
5. **简单 UI 显示**（2:45-3:00）

视频重点展示**已完成的工作**和**遇到的技术挑战**，而非最终完整功能。

---

## 五、总结与展望

### 5.1 学习收获

虽然项目未完全实现预定目标，但在这个过程中我获得了宝贵的经验：

#### **技术层面的收获**
1. **Linux 开发环境熟悉**：第一次在 Ubuntu 下完成完整的 C++项目开发，熟悉了命令行工具、包管理等
2. **大型项目编译经验**：学会了使用 CMake 管理复杂依赖，解决各种编译错误
3. **物理引擎初体验**：对 MuJoCo 的基本原理和使用方法有了初步了解
4. **问题解决能力**：学会了如何系统地排查和解决技术问题

#### **工程实践的收获**
1. **时间管理教训**：认识到复杂项目需要更精细的时间规划和风险管理
2. **增量开发重要性**：学会了如何设定可实现的小目标，逐步推进
3. **文档和记录**：养成了记录开发过程和问题的习惯
4. **求助和搜索**：掌握了如何有效地搜索技术问题和寻求帮助

#### **最重要的收获**
1. **面对复杂系统的勇气**：不再畏惧大型开源项目和复杂技术栈
2. **学习能力的提升**：掌握了快速学习新技术的方法论
3. **实事求是的态度**：学会了根据实际情况调整目标和预期

### 5.2 不足之处与反思

#### **技术实现方面的不足**
1. **时间规划不合理**：低估了环境配置和学习曲线的时间需求
2. **目标设定过高**：对自身能力和项目难度评估不足
3. **技术准备不充分**：在开始编码前，对 MuJoCo 和 OpenGL 的了解不够深入
4. **调试效率不高**：花费太多时间在一些基础问题上

#### **工程管理方面的不足**
1. **缺乏里程碑设置**：没有设定清晰的阶段性目标
2. **风险管理不足**：没有预见到可能的技术障碍
3. **知识管理不系统**：学习过程中的笔记和总结不够系统

4. **代码管理简单**：Git 使用较为基础，缺乏良好的提交习惯

#### **具体技术短板**
1. **OpenGL 图形编程**：需要进一步学习现代 OpenGL 技术
2. **C++高级特性**：对模板、智能指针等高级特性使用不够熟练
3. **多线程编程**：缺乏实时系统的多线程编程经验
4. **性能优化**：对图形渲染的性能优化了解有限

### 5.3 未来改进方向

#### **短期改进计划（2-4 周）**
1. **完成基础仪表盘**
   - 实现完整的速度表和转速表
   - 添加基本的数字显示区域
   - 确保数据实时更新

2. **代码重构和优化**
   - 整理现有代码，提高可读性
   - 添加必要的注释和文档
   - 实现基本的错误处理

3. **完善开发环境**
   - 创建一键配置脚本
   - 整理常见问题解决方案
   - 建立项目 Wiki 文档

#### **中期学习计划（1-2 个月）**
1. **技术深化学习**
   - 系统学习 OpenGL 图形编程
   - 深入研究 MuJoCo 物理引擎
   - 学习现代 C++编程技巧

2. **项目功能完善**
   - 添加 UI 美化效果
   - 实现更多仪表盘组件
   - 添加用户交互功能

3. **工程能力提升**
   - 学习使用更专业的开发工具
   - 掌握性能分析和优化方法
   - 学习软件测试方法

#### **长期发展规划**
1. **技术栈扩展**

   - 学习 Vulkan 或 WebGPU 等现代图形 API
   - 探索物理仿真的数学基础
   - 了解自动驾驶和机器人控制相关知识

2. **项目扩展方向**
   - 将项目移植到其他平台
   - 开发更复杂的仿真场景
   - 探索在教育和研究中的应用

3. **能力体系建设**
   - 建立个人技术知识库
   - 参与开源项目贡献
   - 关注行业发展趋势

### 项目反思与感悟

通过这个未完成但充满挑战的项目，我深刻认识到：

#### **关于技术学习**
1. **理论与实践的结合**：仅仅学习理论是不够的，必须通过实践来加深理解
2. **困难是成长的机会**：遇到的每一个问题都是学习新知识的机会
3. **持续学习的重要性**：技术发展迅速，需要保持学习的热情和习惯

#### **关于项目管理**
1. **合理评估的重要性**：准确评估项目难度和自身能力是关键
2. **灵活调整的必要性**：当实际情况与计划不符时，需要及时调整策略
3. **过程与结果并重**：即使最终结果不完美，过程中的收获同样宝贵

#### **关于个人成长**
1. **接受不完美**：认识到自己的能力边界，接受暂时的不完美
2. **保持耐心和毅力**：复杂项目的完成需要时间和坚持
3. **重视基础积累**：扎实的基础知识是应对复杂问题的关键

### 致谢

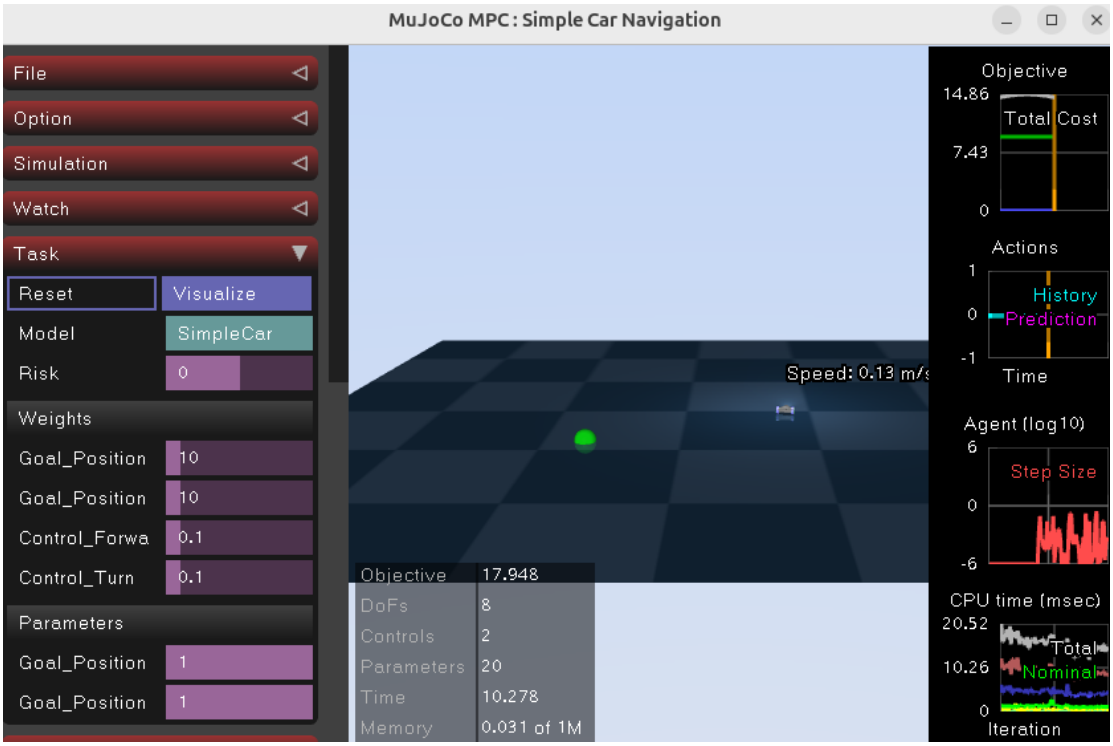
感谢这个项目让我：
- 从畏惧复杂系统到敢于尝试挑战
- 从被动学习到主动探索
- 从关注结果到享受过程

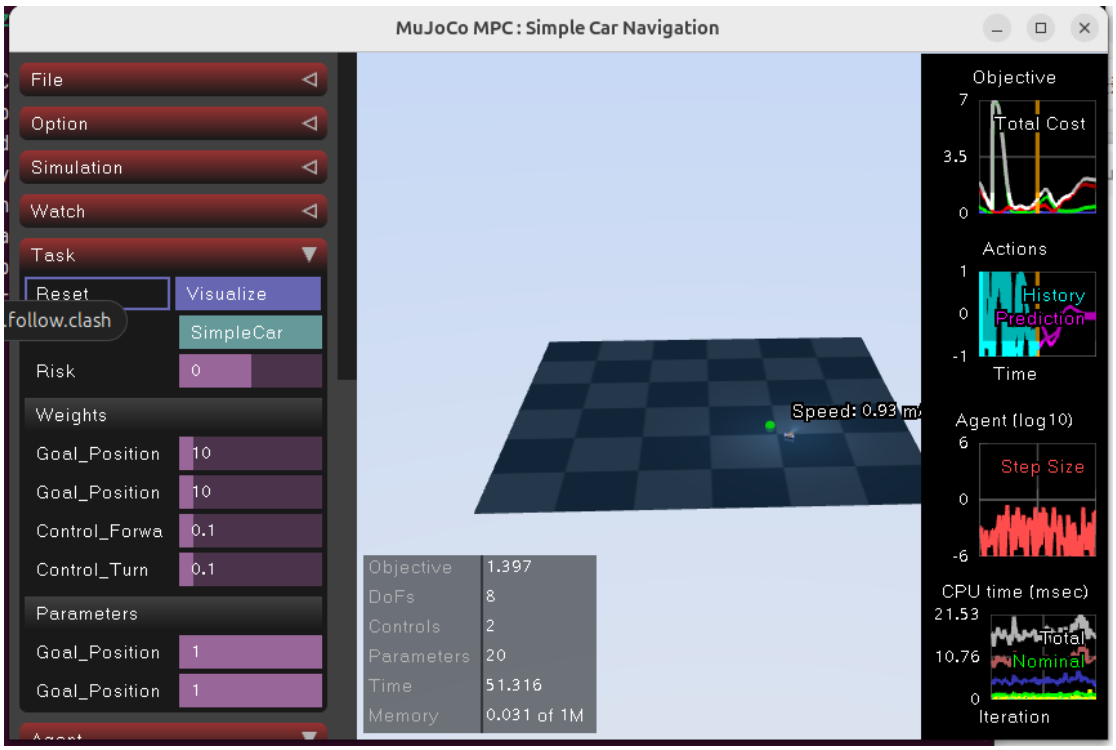
虽然项目没有完全达到预期目标，但这个过程让我获得了远比完成作业更宝贵的东西：**面对复杂问题的勇气、解决问题的能力、以及持续学习的动力**。

我相信，这次的经验将成为我未来技术道路上重要的基石。

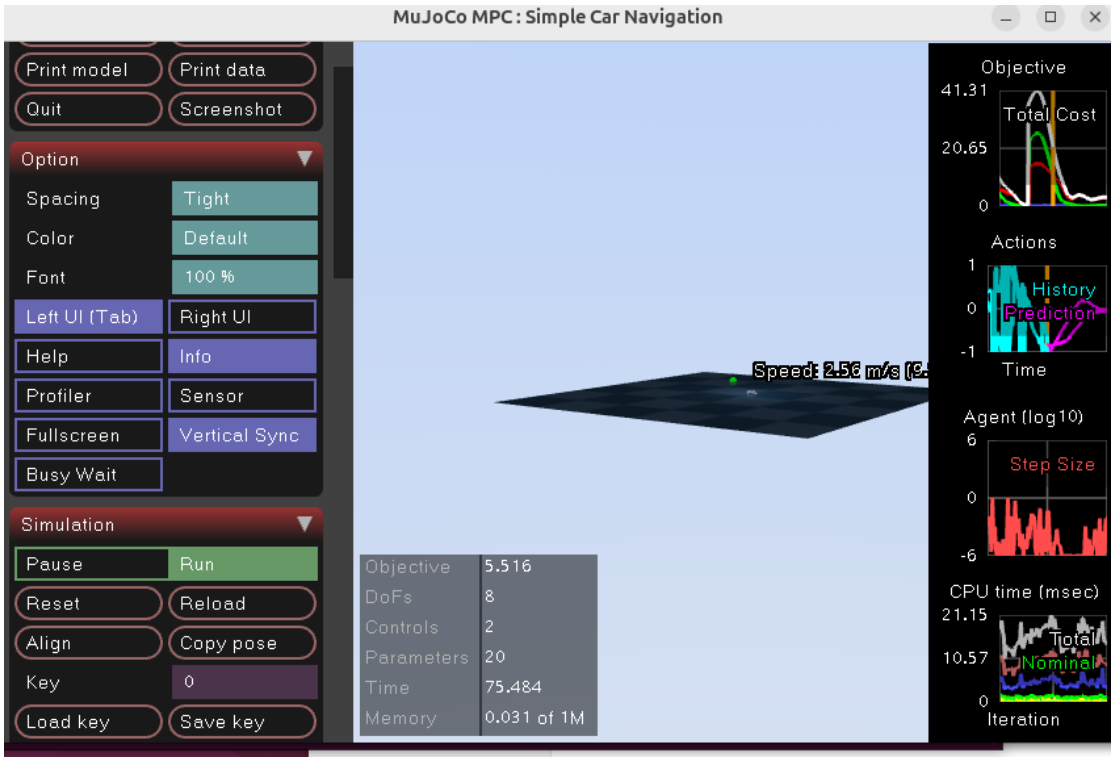
---

**报告人**: [朱志立]
**完成状态**: 基础功能实现，UI 部分未完成
**学习收获**: ⭐⭐⭐⭐⭐ (远超预期)
**日期**: 2025 年 12 月 25 日





MuJoCo MPC : Simple Car Navigation
File
Option
Simulation
Watch
Task
Reset
Visualize
SimpleCar
Risk
0
Weights
Goal_Position
10
Goal_Position
10
Control_Forwa
0.1
Control_Turn
0.1
Parameters
Goal_Position
1
Goal_Position
1
Speed: 0.93 m/
Objective
1.397
DoFs
8
Controls
2
Parameters
20
Time
51.316
Memory
0.031 of 1M
Objective
7
3.5
0
Total Cost
Actions
1
0
-1
History
Prediction
Time
Agent (log10)
6
0
-6
Step Size
CPU time (msec)
21.53
10.76
0
Total
Nominal
Iteration


MuJoCo MPC : Simple Car Navigation
File
Option
Simulation
Watch
Task
Reset
Visualize
Model
SimpleCar
Risk
0
Weights
Goal_Position
10
Goal_Position
10
Control_Forwa
0.1
Control_Turn
0.1
Parameters
Goal_Position
1
Goal_Position
1
Speed: 0.61 m/s (2.2 km/
Objective
0.311
DoFs
8
Controls
2
Parameters
20
Time
61.376
Memory
0.031 of 1M
Objective
7.3
3.65
0
Total Cost
Actions
1
0
-1
History
Prediction
Time
Agent (log10)
6
0
-6
Step Size
CPU time (msec)
20.67
10.33
0
Total
Nominal
Iteration


MuJoCo MPC : Simple Car Navigation
Print model
Print data
Quit
Screenshot
Option
Spacing
Tight
Color
Default
Font
100 %
Left UI (Tab)
Right UI
Help
Info
Profiler
Sensor
Fullscreen
Vertical Sync
Busy Wait
Simulation
Pause
Run
Reset
Reload
Align
Copy pose
Key
0
Load key
Save key
Objective 5.516
DoFs 8
Controls 2
Parameters 20
Time 75.484
Memory 0.031 of 1M
Objective
41.31
Total Cost
20.65
0
Actions
1
History
0
Prediction
-1
Time
Agent (log10)
6
Step Size
0
-6
CPU time (msec)
21.15
Total
10.57
Nominal
0
Iteration