MuJoCo MPC 汽车仪表盘项目

##项目信息

- **学号**: 232011060# MuJoCo MPC 汽车仪表盘项目

##项目信息

- **学号**: 232011060

- **姓名**: 邓志锦

- **班级**: 计科 2302 班

- **完成日期**: 2025 年 12月23日

##项目概述

这个实验实现了一个基于 MuJoCo 物理引擎的汽车仪表盘可视化系统，通过 C++编程将 3D 物理仿真与 2D 图形界面相结合，能够实时模拟车辆的运动状态（如加速、转向和刹车），并动态显示关键驾驶数据（包括速度、转速、油量和温度等）在虚拟仪表盘上，从而在虚拟环境中构建了一个数据驱动的交互式驾驶仿真界面。

##环境要求

- 操作系统: Ubuntu 24.04

- 编译器: gcc 11.3.0

- CMake: 3.22.1

##操作

//更新 apt

```
sudo apt update

sudo apt upgrade -y
```

//工具

```
sudo apt install -y \
```

```
   build-essential \
   cmake \
   git \
   pkg-config \
   unzip \
   wget
//依赖
sudo apt install -y \
   libgl1-mesa-dev \
   libglfw3-dev \
   libglew-dev \
   libxinerama-dev \
   libxcursor-dev \
   libxrandr-dev \
   libxi-dev \
   libeigen3-dev \
   libopenblas-dev \
   mesa-utils
apt install -y freeglut3-dev
cmake ..
make -j$(nproc)
./bin/mjpc --task=SimpleCar
```

MuJoCo MPC (MJPC) is an interactive application and software framework for real-time predictive control with [MuJoCo](https://mujoco.org/), developed by Google DeepMind.

MJPC allows the user to easily author and solve complex robotics tasks, and currently supports multiple shooting-based planners. Derivative-based methods include iLQG and Gradient Descent, while derivative-free methods include a simple yet very competitive planner called Predictive Sampling.

Overview

To read the paper describing this software package, please see our

[preprint](https://arxiv.org/abs/2212.00541).

For a quick video overview of MJPC, click below.

[![Video](http://img.youtube.com/vi/Bdx7DuAMB6o/hqdefault.jpg)](https://dpmd.ai/mjpc)

For a longer talk at the MIT Robotics Seminar in December 2022 describing our results, click

below.

[![2022Talk](http://img.youtube.com/vi/2xVN-qY78P4/hqdefault.jpg)](https://www.youtube.com/watch?v=2xVN-qY78P4)

A more recent, December 2023 talk at the IEEE Technical Committee on Model-Based Optimization

is available here:

[![2023Talk](https://img.youtube.com/vi/J-JO-lgaKtw/hqdefault.jpg)](https://www.youtube.com/watch?v=J-JO-lgaKtw\&t=0s)

Example tasks

Quadruped task:

[![Quadruped](http://img.youtube.com/vi/esLuwaWz4oE/hqdefault.jpg)](https://www.youtube.com/watch?v=esLuwaWz4oE)

Bimanual manipulation:

[![Bimanual](http://img.youtube.com/vi/aCNCKVThKIE/hqdefault.jpg)](https://www.youtube.com/watch?v=aCNCKVThKIE)

Rubik's cube 10-move unscramble:

[![Unscramble](http://img.youtube.com/vi/ZRRvVWV-Muk/hqdefault.jpg)](https://www.youtube.com/watch?v=ZRRvVWV-Muk)

Humanoid motion-capture tracking:

[![Tracking](http://img.youtube.com/vi/tEBVK-MO1Sw/hqdefault.jpg)](https://www.youtube.com/watch?v=tEBVK-MO1Sw)

Graphical User Interface

For a detailed dive of the graphical user interface, see the

[MJPC GUI](docs/GUI.md) documentation.

Installation

MJPC is tested with [Ubuntu 20.04](https://releases.ubuntu.com/focal/) and [macOS-12](https://www.apple.com/by/macos/monterey/). In principle, other versions and Windows operating system should work with MJPC, but these are not tested.

Prerequisites

Operating system specific dependencies:

macOS

Install [Xcode](https://developer.apple.com/xcode/).

Install ninja and zlib:

```
brew install ninja zlib
```

Ubuntu 20.04

```
sudo apt-get update \&\& sudo apt-get install cmake libgl1-mesa-dev libxinerama-dev libxcursor-dev libxrandr-dev libxi-dev ninja-build zlib1g-dev clang-12
```

Clone MuJoCo MPC

```
git clone https://github.com/google-deepmind/mujoco\_mpc
```

Build and Run MJPC GUI application

1. Change directory:

```
cd mujoco\_mpc
```

2. Create and change to build directory:

```
mkdir build

cd build
```

3. Configure:

macOS-12

```
cmake .. -DCMAKE\_BUILD\_TYPE:STRING=Release -G Ninja -DMJPC\_BUILD
\_GRPC\_SERVICE:BOOL=ON
```

Ubuntu 20.04

```
cmake .. -DCMAKE\_BUILD\_TYPE:STRING=Release -G Ninja -DCMAKE\_C\_C
OMPILER:STRING=clang-12 -DCMAKE\_CXX\_COMPILER:STRING=clang++-12 -D
MJPC\_BUILD\_GRPC\_SERVICE:BOOL=ON
```

Note: gRPC is a large dependency and can take 10-20 minutes to initially download.

4. Build

```
cmake --build . --config=Release
```

6. Run GUI application

```
cd bin

./mjpc
```

Build and Run MJPC GUI application using VSCode

We recommend using [VSCode](https://code.visualstudio.com/) and 2 of its extensions ([CMake Tools](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cmake-tools) and [C/C++](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools))
to simplify the build process.

1. Open the cloned directory `mujoco_mpc`.
2. Configure the project with CMake (a pop-up should appear in VSCode)
3. Set compiler to `clang-12`.
4. Build and run the `mjpc` target in "release" mode (VSCode defaults to "debug"). This will open and run the graphical user interface.

Build Issues

If you encounter build issues, please see the [Github Actions configuration](https://github.com/google-deepmind/mujoco_mpc/blob/main/.github/workflows/build.yml).
This provides the exact setup we use for building MJPC for testing with Ubuntu 20.04 and macOS-12.

Python API

We provide a simple Python API for MJPC. This API is still experimental and expects some more experience from its users. For example, the correct usage requires that the model (defined in Python) and the MJPC task (i.e., the residual and transition functions defined in C++) are compatible with each other. Currently, the Python API does not provide any particular error handling for verifying this compatibility and may be difficult to debug without more in-depth knowledge about MuJoCo and MJPC.

Installation

Prerequisites

1. Build MJPC (see instructions above).

2. Python 3.10

3. (Optionally) Create a conda environment with **Python 3.10**:

```
conda create -n mjpc python=3.10

conda activate mjpc
```

4. Install MuJoCo

```
pip install mujoco
```

Install API

Next, change to the python directory:

```
cd python
```

Install the Python module:

```
python setup.py install
```

Test that installation was successful:

```
python "mujoco\_mpc/agent\_test.py"
```

Example scripts are found in python/mujoco_mpc/demos. For example from python/:

```
python mujoco\_mpc/demos/agent/cartpole\_gui.py
```

will run the MJPC GUI application using MuJoCo's passive viewer via Python.

Python API Installation Issues

If your installation fails or is terminated prematurely, we recommend deleting the MJPC build directory and starting from scratch as the build will likely be corrupted. Additionally, delete the files generated during the installation process from the python/ directory.

Predictive Control

See the [Predictive Control](docs/OVERVIEW.md) documentation for more

information.

Contributing

See the [Contributing](docs/CONTRIBUTING.md) documentation for more information.

Known Issues

MJPC is not production-quality software, it is a **research prototype**. There are likely to be missing features and outright bugs. If you find any, please report them in the [issue tracker](https://github.com/google-deepmind/mujoco_mpc/issues).

Below we list some known issues, including items that we are actively working on.

- We have not tested MJPC on Windows, but there should be no issues in principle.
- Task specification, in particular the setting of norms and their parameters in XML, is a bit clunky. We are still iterating on the design.
- The Gradient Descent search step is proportional to the scale of the cost function and requires per-task tuning in order to work well. This is not a bug but a property of vanilla gradient descent. It might be possible to ameliorate this with some sort of gradient normalisation, but we have not investigated this thoroughly.

Citation

If you use MJPC in your work, please cite our accompanying [preprint](https://arxiv.org/abs/2212.00541):

```
@article{howell2022,

  title={{Predictive Sampling: Real-time Behaviour Synthesis w
ith MuJoCo}},

  author={Howell, Taylor and Gileadi, Nimrod and Tunyasuvunako
ol, Saran and Zakka, Kevin and Erez, Tom and Tassa, Yuval},

  archivePrefix={arXiv},

  eprint={2212.00541},

  primaryClass={cs.RO},

  url={https://arxiv.org/abs/2212.00541},

  doi={10.48550/arXiv.2212.00541},
```

Acknowledgments

The main effort required to make this repository publicly available was

undertaken by [Taylor Howell](https://thowell.github.io/) and the Google

DeepMind Robotics Simulation team.

License and Disclaimer

- **姓名**: 邓志锦

- **班级**: 计科2302班

- **完成日期**: 2025年12月23日

##项目概述

这个实验实现了一个基于 MuJoCo 物理引擎的汽车仪表盘可视化系统，通过 C++编程将 3D 物理仿真与 2D 图形界面相结合，能够实时模拟车辆的运动状态（如加速、转向和刹车），并动态显示关键驾驶数据（包括速度、转速、油量和温度等）在虚拟仪表盘上，从而在虚拟环境中构建了一个数据驱动的交互式驾驶仿真界面。

##环境要求

- 操作系统: Ubuntu 24.04

- 编译器: gcc 11.3.0

- CMake: 3.22.1

##操作

//更新 apt

```
sudo apt update
sudo apt upgrade -y
//工具
sudo apt install -y \
   build-essential \
   cmake \
   git \
   pkg-config \
   unzip \
   wget
//依赖
sudo apt install -y \
   libgl1-mesa-dev \
   libglfw3-dev \
   libglew-dev \
   libxinerama-dev \
   libxcursor-dev \
   libxrandr-dev \
   libxi-dev \
   libeigen3-dev \
   libopenblas-dev \
   mesa-utils
apt install -y freeglut3-dev
```

```
cmake ..

make -j$(nproc)

./bin/mjpc --task=SimpleCar
```

MuJoCo MPC (MJPC) is an interactive application and software framework for real-time predictive control with [MuJoCo](https://mujoco.org/), developed by Google DeepMind.

MJPC allows the user to easily author and solve complex robotics tasks, and currently supports multiple shooting-based planners. Derivative-based methods include iLQG and Gradient Descent, while derivative-free methods include a simple yet very competitive planner called Predictive Sampling.

Overview

To read the paper describing this software package, please see our

[preprint](https://arxiv.org/abs/2212.00541).

For a quick video overview of MJPC, click below.

[![Video](http://img.youtube.com/vi/Bdx7DuAMB6o/hqdefault.jpg)](https://dpmd.ai/mjpc)

For a longer talk at the MIT Robotics Seminar in December 2022 describing our results, click

below.

[![2022Talk](http://img.youtube.com/vi/2xVN-qY78P4/hqdefault.jpg)](https://www.youtube.com/watch?v=2xVN-qY78P4)

A more recent, December 2023 talk at the IEEE Technical Committee on Model-Based Optimization

is available here:

[![2023Talk](https://img.youtube.com/vi/J-JO-lgaKtw/hqdefault.jpg)](https://www.youtube.com/watch?v=J-JO-lgaKtw\&t=0s)

Example tasks

Quadruped task:

[![Quadruped](http://img.youtube.com/vi/esLuwaWz4oE/hqdefault.jpg)](https://www.youtube.com/watch?v=esLuwaWz4oE)

Bimanual manipulation:

[![Bimanual](http://img.youtube.com/vi/aCNCKVThKIE/hqdefault.jpg)](https://www.youtube.com/watch?v=aCNCKVThKIE)

Rubik's cube 10-move unscramble:

[![Unscramble](http://img.youtube.com/vi/ZRRvVWV-Muk/hqdefault.jpg)](https://www.youtube.com/watch?v=ZRRvVWV-Muk)

Humanoid motion-capture tracking:

[![Tracking](http://img.youtube.com/vi/tEBVK-MO1Sw/hqdefault.jpg)](https://www.youtube.com/watch?v=tEBVK-MO1Sw)

Graphical User Interface

For a detailed dive of the graphical user interface, see the

[MJPC GUI](docs/GUI.md) documentation.

Installation

MJPC is tested with [Ubuntu 20.04](https://releases.ubuntu.com/focal/) and [macOS-12](https://www.apple.com/by/macos/monterey/). In principle, other versions and Windows operating system should work with MJPC, but these are not tested.

Prerequisites

Operating system specific dependencies:

macOS

Install [Xcode](https://developer.apple.com/xcode/).

Install ninja and zlib:

```
brew install ninja zlib
```

Ubuntu 20.04

```
sudo apt-get update \&\& sudo apt-get install cmake libgl1-mesa-dev libxinerama-dev libxcursor-dev libxrandr-dev libxi-dev ninja-build zlib1g-dev clang-12
```

Clone MuJoCo MPC

```
git clone https://github.com/google-deepmind/mujoco\_mpc
```

Build and Run MJPC GUI application

1. Change directory:

```
cd mujoco\_mpc
```

2. Create and change to build directory:

```
mkdir build

cd build
```

3. Configure:

macOS-12

```
cmake .. -DCMAKE\_BUILD\_TYPE:STRING=Release -G Ninja -DMJPC\_BUILD
\_GRPC\_SERVICE:BOOL=ON
```

Ubuntu 20.04

```
cmake .. -DCMAKE\_BUILD\_TYPE:STRING=Release -G Ninja -DCMAKE\_C\_C
OMPILER:STRING=clang-12 -DCMAKE\_CXX\_COMPILER:STRING=clang++-12 -D
MJPC\_BUILD\_GRPC\_SERVICE:BOOL=ON
```

Note: gRPC is a large dependency and can take 10-20 minutes to initially download.

4. Build

```
cmake --build . --config=Release
```

6. Run GUI application

```
cd bin

./mjpc
```

Build and Run MJPC GUI application using VSCode

We recommend using [VSCode](https://code.visualstudio.com/) and 2 of its extensions ([CMake Tools](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cmake-tools) and [C/C++](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools))
to simplify the build process.

1. Open the cloned directory `mujoco_mpc`.
2. Configure the project with CMake (a pop-up should appear in VSCode)
3. Set compiler to `clang-12`.
4. Build and run the `mjpc` target in "release" mode (VSCode defaults to "debug"). This will open and run the graphical user interface.

Build Issues

If you encounter build issues, please see the [Github Actions configuration](https://github.com/google-deepmind/mujoco_mpc/blob/main/.github/workflows/build.yml).
This provides the exact setup we use for building MJPC for testing with Ubuntu 20.04 and macOS-12.

Python API

We provide a simple Python API for MJPC. This API is still experimental and expects some more experience from its users. For example, the correct usage requires that the model (defined in Python) and the MJPC task (i.e., the residual and transition functions defined in C++) are compatible with each other. Currently, the Python API does not provide any particular error handling for verifying this compatibility and may be difficult to debug without more in-depth knowledge about MuJoCo and MJPC.

Installation

Prerequisites

1. Build MJPC (see instructions above).

2. Python 3.10

3. (Optionally) Create a conda environment with **Python 3.10**:

```
conda create -n mjpc python=3.10

conda activate mjpc
```

4. Install MuJoCo

```
pip install mujoco
```

Install API

Next, change to the python directory:

```
cd python
```

Install the Python module:

```
python setup.py install
```

Test that installation was successful:

```
python "mujoco\_mpc/agent\_test.py"
```

Example scripts are found in python/mujoco_mpc/demos. For example from python/:

```
python mujoco\_mpc/demos/agent/cartpole\_gui.py
```

will run the MJPC GUI application using MuJoCo's passive viewer via Python.

Python API Installation Issues

If your installation fails or is terminated prematurely, we recommend deleting the MJPC build directory and starting from scratch as the build will likely be corrupted. Additionally, delete the files generated during the installation process from the python/ directory.

Predictive Control

See the [Predictive Control](docs/OVERVIEW.md) documentation for more

information.

Contributing

See the [Contributing](docs/CONTRIBUTING.md) documentation for more information.

Known Issues

MJPC is not production-quality software, it is a **research prototype**. There are likely to be missing features and outright bugs. If you find any, please report them in the [issue tracker](https://github.com/google-deepmind/mujoco_mpc/issues).

Below we list some known issues, including items that we are actively working on.

- We have not tested MJPC on Windows, but there should be no issues in principle.
- Task specification, in particular the setting of norms and their parameters in XML, is a bit clunky. We are still iterating on the design.
- The Gradient Descent search step is proportional to the scale of the cost function and requires per-task tuning in order to work well. This is not a bug but a property of vanilla gradient descent. It might be possible to ameliorate this with some sort of gradient normalisation, but we have not investigated this thoroughly.

Citation

If you use MJPC in your work, please cite our accompanying [preprint](https://arxiv.org/abs/2212.00541):

```
@article{howell2022,

  title={{Predictive Sampling: Real-time Behaviour Synthesis w
ith MuJoCo}},

  author={Howell, Taylor and Gileadi, Nimrod and Tunyasuvunako
ol, Saran and Zakka, Kevin and Erez, Tom and Tassa, Yuval},

  archivePrefix={arXiv},

  eprint={2212.00541},

  primaryClass={cs.RO},

  url={https://arxiv.org/abs/2212.00541},

  doi={10.48550/arXiv.2212.00541},
```

Acknowledgments

The main effort required to make this repository publicly available was undertaken by [Taylor Howell](https://thowell.github.io/) and the Google DeepMind Robotics Simulation team.

License and Disclaimer