Introduction to ROS
Group Project: Autonomous Quadruped

1 General Information

1.1 Administration

As it is impossible to match the three offered tasks in complexity, we will account for it by matching the grading distributions of the three tasks.

1.2 Timeline

- Today: Announcing the projects
- 07.06.2023: Release of code
- 02.08.2023: Submission and presentations

1.3 Submission

The project submission consists of three parts:

- Source Code (Deadline: 29.07.2024 - End of day)
- Documentation (Deadline: 29.07.2024 - End of day)
- Presentation (Deadline: 30.07.2024 - detailed times will be shared!)

The presentation will not be graded but shall help the reviewers to quicker understand your code structure, limitations and issues you had during implementation.

Your source code and documentation must be submitted via a link to a github repository. In both cases you should include the src folder of your catkin workspace as well as the source code folder(s). In case you modified your unity simulations, please communicate that clearly in your documentation and give us access to your simulator source and binary.

1.3.1 Source Code

While writing code, please pay attention to writing clean code, think about how to structure your code into individual ROS packages and nodes. Grading will as well be dependent on the structure of your project.

Please also include a `readme.md` file with detailed step by step installation and launch instructions necessary to run your code base. Make sure that all code can be launched by a single launch file. We don't want to launch your code from multiple terminals. Indicate in your readme which Ubuntu version has been used.

As a hint, before submission run your code from a fresh installation of Ubuntu to make sure you have all necessary installation steps in your documentation!

You are allowed to use any free available code but you are NOT allowed to use source code from other groups of this course.

1.3.2 Documentation

The documentation shall be approx. 4 pages but is not limited to that. The goal of the documentation is to help us understanding your code. In your documentation you should include a short description of every ROS node and package and its functionality. Please let us as well know in case you were not able to fulfill all requirements and explain the problems. Document where you used external code, not written by yourself. Finally, discuss who of the team members worked on which components. Beyond that, you must include

- a ROS graph, indicating who worked on which part
- figures, and plots presenting your results,
- a bibliography.

1.3.3 Presentation

Please prepare an oral presentation of your documentation. The presentation length shall not be longer than 5 minutes. An introduction to the field and topic is not required. The goal of the presentation is to give us a quick overview over your project, explain how you implemented the individual components, discuss limitations and issues and give us feedback about the project work.

2 Task

2.1 Goals

The goal of the project is passing a parkour with your robot in minimal time, while not hitting any of the cones.

The core parts, including but not limited are:

- A Unity simulation environment. A base version will be provided to you.
- A ROS-Simulation-Bridge providing ROS interfaces (topics, services). It will communicate with the simulation via TCP while at the same time providing relevant information to other ROS nodes. A base version will be provided to you. You are free to adjust the code.
- A basic position controller leveraging gait walking.
- A state machine for your robot.
- A perception pipeline that converts the depth image, first to a **point cloud** and second to a **voxel-grid** representation of the environment.
- A path planner that generates a path through the environment.
- A trajectory planner that plans a trajectory based on the found path.

It is required that you at least once implement one of the following ROS elements by yourself:

- ROS service - document where you used a ROS service call
- Implement an own message type - document which message you defined by yourself

2.2 Grading

You can reach a maximum of 100 points in the following categories:

- Functionality and Performance (max. 50p)
  - Successfully working perception pipeline (max. 10p)
  - Successfully working path planning (max. 8p)
  - Successfully working trajectory planning (max. 8p)
  - Successfully working planner for coordinated motion (max. 6p)
  - Successfully crossing the parkour (max. 12p)
  - Time to complete the mission (max. 6p)
- Code and Architecture Quality (max. 30p)
  - Sensible separation in packages, nodes and functions (max. 10p)
  - Sensible separation in services, topics, etc (max. 10p)
  - Code quality (use in a group same coding style!), readability, and traceability (max. 10p)
- Written Summary (max. 20p)
  - Sound explanation of your architecture, model, and design choices (max. 15p)
  - Clear documentation who did what, which code is your own, which code is reused (max. 5p)
- Bonus Points (max. 10p)
  - Solving the problem in the shortest time (max. 10p)

Very important: We will subtract up to 30p if your code does not build or performes differently then documented.

2.3 Tips

Here are a couple of hints regarding the implementation. The hints are just suggestions, you are free so solve the task differently:

- Generating point cloud from depth image: use *depth_image_proc* in http://wiki.ros.org/depth_image_proc.
- Generating occupancy Grid: use *Octomap* in http://wiki.ros.org/octomap.
- **Please ping us in case you have any questions or if you get stuck in some subtasks.**