Lunar Scout: Task 2C Path Traversal

Krutarth e-Yantra Staff Oct 2023

Aim:

Great Job! Take a time to give a pat on your back yourselves → as you have reached this milestone. Take this task to perfection - marking a good conclusion to your stage-1 learning journey with us… also making your candidature strong for stage-2.

Now as your bike & control algorithm are both mature enough - in this task you have to actuate the drive motor and make the bike move around. It has to move bike to reach and trigger one checkpoint by going near it within a proximity, without colliding with it and come back to the start point.

Few things new in this task:

- **Manual Control:** In this task, the balancing will be AUTOMATIC - by your control algorithm from child script. But the motion of the bike and *actions like forward, backward, turning* etc. will be MANUAL, via ***keyboard input***.
- **Checkpoint :** A sphere object will be added in scene at runtime, once you execute the evaluator. Let's refer it as ***checkpoint*** (which strictly **should not be tampered with**). It will be placed at any random position on the floor.
    - As bike moves within proximity of checkpoint and is close enough - then it will change color to **GREEN**. If it collides with bike then it will change color to **RED**. Few points will be allotted for successfully triggering the green indication without colliding, and a no points will be given for collision. But it will be considered as a successful attempt if you return back to start point.
- **Start Point & TIMEOUT :** Start point is same as end point and denoted by a YELLOW disc. As bike returns to the "start point" **after**

Skip to main content

completing the checkpoint OR **TIMEOUT of 120 seconds** happens… the simulation will be stopped.

> The bike is allowed to move.
> Apart from this, **all other RULES from Task2A & Task2B will still be applied.** Do not violate any of those.

Downloadables

The boilerplate **Task2C.zip** contains a folder Task2C which will have the following files:

- Task_2C.ttt (scene without the bike)
- linux
  - task2c (evaluator for Linux)
- mac (coming soon)
  - task2c (evaluator for MacOs)
- windows
  - task2c.exe (evaluator for Windows)

Understanding Boilerplate:

Open the scene **Task_2C.ttt** → then import the bike model by dragging & dropping it from model browser

- Add a **PYTHON** child script to ***front_motor***.
- A python API stub is given below for the manual keypress signalling in CoppeliaSim Scene
  For example, you can start with the following code in ***sysCall_sensing():***

```
############### Keyboard Input ###########
    message,data,data2 = sim.getSimulatorM

    if (message == sim.message_keypress):
        if (data[0]==2007): # forward up a
            drive_speed = #add drive wheel

        if (data[0]==2008): # backward dow
            drive_speed = #add drive wheel

        if (data[0]==2009): # left arrow k
            yaw_setpoint = #change yaw_set

        if (data[0]==2010): # right arrow
            yaw_setpoint = #change yaw_set
```

Skip to main content

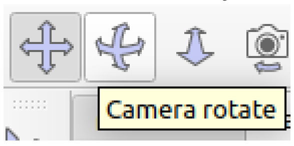
```
else:
    drive_speed = # This is an example
     yaw_setpoint = # # This is an exa
######################################
```

Balancing still has to be automatic(from PID or LQR).

- Add your control logic similar to Task2B
- The **bike_respondable**, should be made `collidable`. You can do it by enabling a checkbox in object properties by double clicking object in hierarchy.

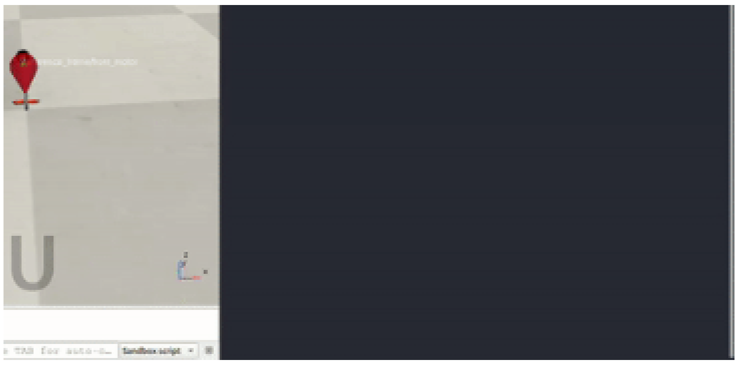
> All properties of every other object should be intact.

The evaluator/grader will start the simulation →
→ Bike will be positioned at the start point indicated by yellow disc.
→ Your solution code in Child script should help bike balance(using PID or LQR) and then user can give keyboard inputs to control forward, backward and turning motion of bike.
→ User should give commands such that bike goes near the checkpoint until it's triggered without collision.
→ The checkpoint will indicate a successful trigger or collision by color change.
→ Evaluator will throw exception if it fails
→ Evaluator will Stop simulation after timeout or return to start point and generate result + comments.
→ Evaluator will generate ***task2c_output.txt***
→ In case bike fails to stay upright and falls down, it will not generate output txt file and abort the run.

> Use the facility given by CoppeliaSim to pan or rotate your view during run, as the **checkpoint** will be added anywhere in the scene.
>
> 
>

Example RUN:

Skip to main content



Submission Instructions:

- Once you have the solution ready, open the solution scene **Task_2C.ttt**, which should have the bike model.
- (ADD childscript content in same folder as **task2c_solution.py** after copying solution code inside)
- Activate the conda environment with the following command:

```
conda activate LS_<team_id>
```

- Run the evaluator file **task2c** or **task2c.exe** by running following command
For **linux** or MacOS:

```
./task2c
```

For **windows**:

```
task2c.exe
```

- If success, it will generate the `task2c_output.txt`
- *Compress the workspace folder Task2C* as **Task2C.zip** which should contain the following :
  - Task2C (folder)
    - Task_2C.ttt (With bike and the solution)
    - task2c_solution.py (solution code of child script)
    - task2c_output.txt
- Click on this link: **https://portal.e-yantra.org/task_task2** . In the **Task 2 Upload**

Topics
My Posts
More
Categories
Lunar Scout (LS)
Guidelines
ChatbotDocs
All categories
Tags
task-0
task-1
task-2
other
task-4
Skip to main content

section, click on **Task 2C** bullet and select **Choose file** button to upload the `Task2C.zip` file. From the dialogue box, select the file and click **Open**.

- You shall see the file name in text-box besides the **Choose file** button. Click on **Upload Task** button to submit the file.

Wohoooo…!!!

🚀

Lunar Scout: Task 2

Closed on Oct 10, 2023

Related Topics

Topic	Replies	Views	Activity
Lunar Scout: Task5 - Mini Theme Implementation	1	441	Jan 19