ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab

Exercise 1
Introduction to CoppeliaSim

1 Introduction

CoppeliaSim is a general purpose robot simulator with integrated development environment. It can be used for various applications (fast algorithm development, factory automation simulations, fast prototyping and verification, robotics related education, etc.) and allows to quickly create and simulate models of robotic contents. The product recently changed its name from "V-Rep Simulator", therefore several folders in the lecture content still contain the name "vrep".

2 Installation

2.1 MATLAB

- Please make sure that your MATLAB installation includes Image Processing Toolbox.

2.2 CoppeliaSim V4.0.0

- **Windows**: download CoppeliaSim EDU and execute the installer (the target folder will be **`Program Files (x86)`**). You should be able to start CoppeliaSim via the application menu, or by double-clicking a CoppeliaSim scene.
- **Mac OSX**: download CoppeliaSim EDU and unzip it. You should be able to start CoppeliaSim by executing **`coppeliaSim.app`** in that folder. You can also start it via the terminal by typing `./coppeliaSim.app/Contents/MacOS/coppeliaSim`.
- **Linux** (Ubuntu, preferrably 18.04 LTS): download CoppeliaSim EDU for your architecture (32 or 64 bits), and unpack the archive. Then you should be able to start CoppeliaSim via command line (navigate to the installation folder and run `./coppeliaSim.sh`. You might need to run `chmod +x coppeliaSim.sh` before for appropriate execution permissions).

3 First steps with CoppeliaSim

Once you have launched CoppeliaSim, the main window will display three distinct parts (see fig. 1):

- The model browser: the model browser supports drag-and drop of CoppeliaSim models into the scene. When you drop a model into the scene hierarchy, the model appears in a default position (usually at the center of the scene). If you drop a model into the scene view, it will be positioned at the drop location.

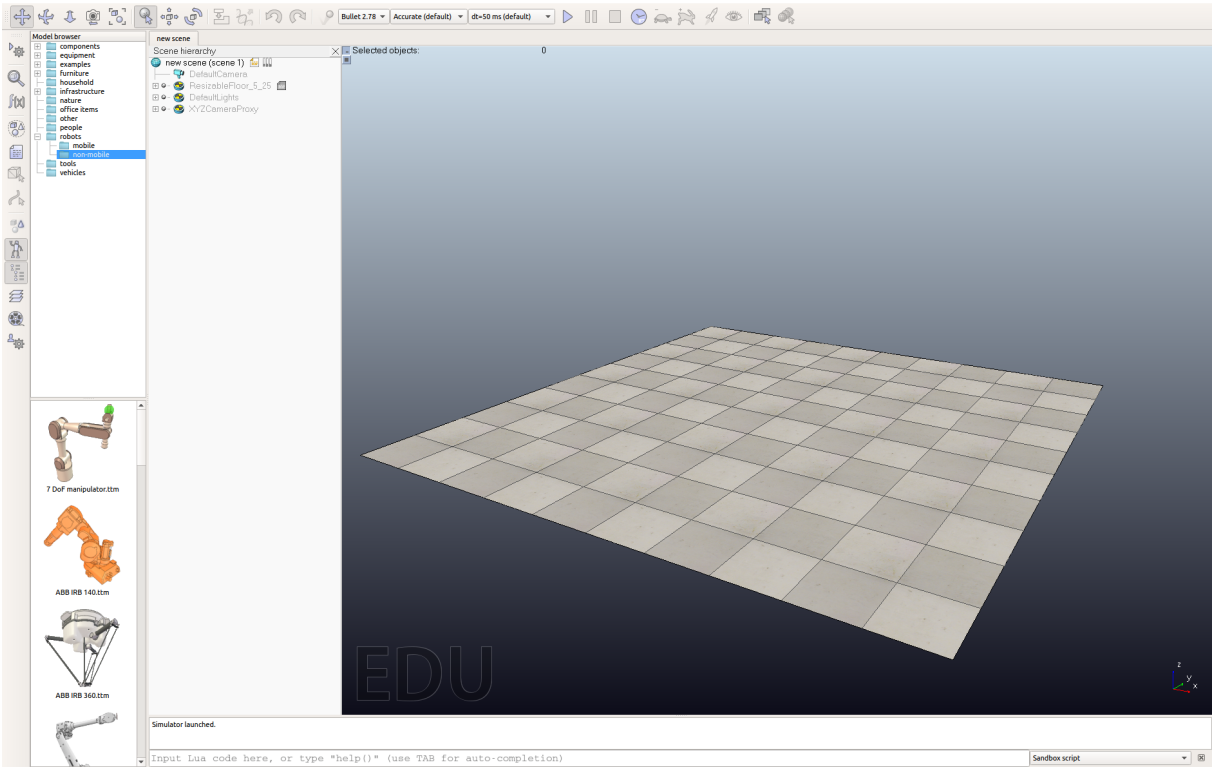

Figure 1: Empty CoppeliaSim environment on startup

- The scene hierarchy: the scene hierarchy displays all objects in the scene, visualizing their parent-child relationships. Each object also reveals its type via its display icon. The icons can be double-clicked to open their respective object property dialogs. On the right-hand side of some object icons you will find another icon representing a script: those icons can also be double-clicked in order to open the attached child script.

- The scene view, or current page: by default, the current page displays the 3D content of a scene. You can click objects/models to select them, and you can manipulate them via the various toolbar buttons, the menu bar, or various key combinations (e.g. delete, copy/paste, etc.)

At the top of the main window you will find the navigation toolbar buttons. They allow changing the navigation mode (camera pan, camera rotate, object shift, object rotate, etc.), they allow changing the main settings of a simulation (e.g. the physics engine that will be used), and they allow starting/pausing and stopping a simulation.

Try to familiarize yourself with the simulator by drag-and-dropping a few robot models into the scene, then hitting the start button. Try also to run a few of the demo scenes (**menu bar –> File –> Open scene...**).

Make sure to also follow a few of the provided tutorials, and to refer to the CoppeliaSim user manual.

4 Scene for the exercises

Some of the exercises of this series of exercises can be run together with CoppeliaSim, in order to visualize in a 3D scene what is going on. Those exercises are based on the CoppeliaSim scene **`scene/mooc_exercises.ttt`**. Via CoppeliaSim's file menu, open the scene. You will see:

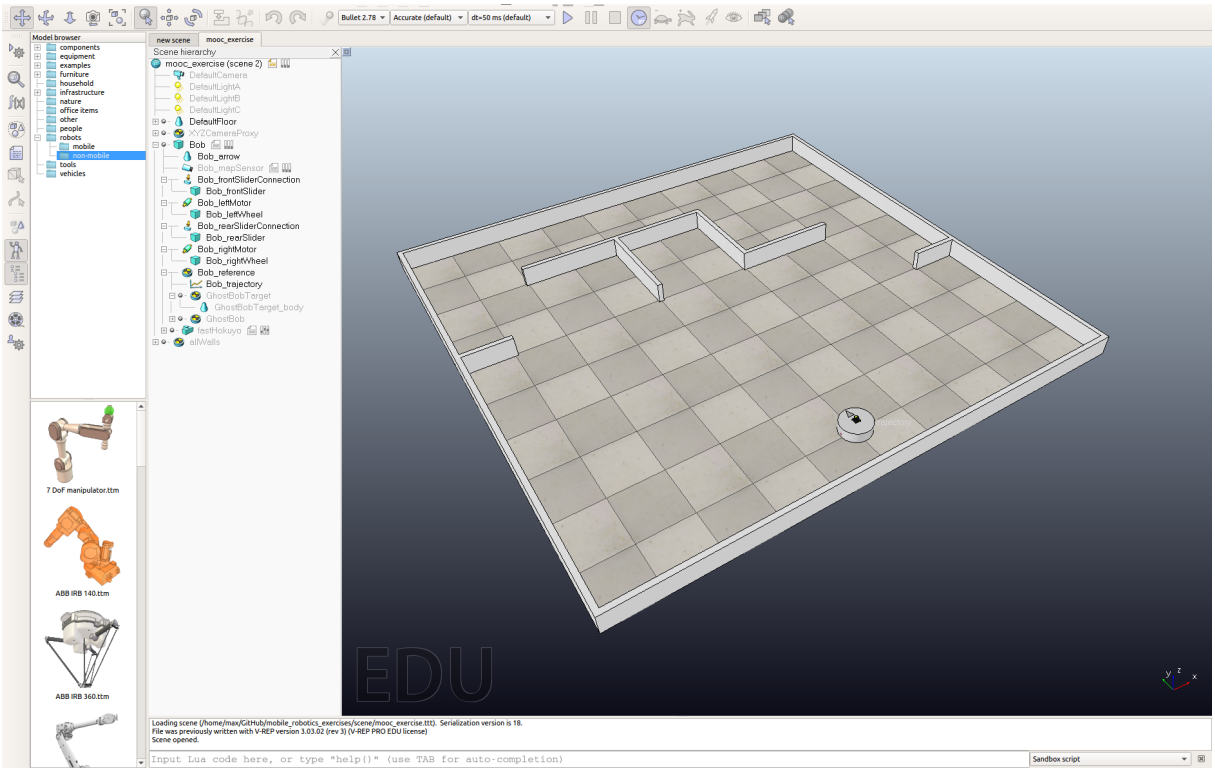

Figure 2: After loading **`scene/mooc_exercises.ttt`**.

- a 5 meter by 5 meter floor
- a mobile robot (*Bob*). The robot has a differential wheel system (i.e. two actuated wheels), and a laser scanner placed on top of the robot. Individual objects of the robot cannot directly be selected in the scene view, you can however do so in the scene hierarchy. You can move and rotate the robot via the appropriate toolbar buttons.
- a series of walls. Those can also be moved and rotated via the appropriate toolbar buttons. In order to vertically shift an item, or to rotate an item around a different axis, make sure to hold down the shift key during manipulation. Additional models can be drag-and-dropped into the scene at any moment (wall sections can be found in the model browser under `Models/infrastructure/walls`).

4.1 Running the simulation

Hit the start button (play button above the scene view). A bitmap of the scene will be displayed in a floating view (*Bob_mapSensor*). The robot itself will not appear in that bitmap. The resolution of the bitmap is 512 by 512 pixels. The laser scanner will visualize its activity (red detection lines), and Bob will receive a tag (*Bob0* for the first robot in the scene).

4.2 Further model info and configuration

Expand the robot model in the scene hierarchy, and double-click the icon of *Bob_leftMotor*. The **joint** property dialog opens. Click **Show dynamic parameters** in order to open the joint dynamic properties dialog. Change the **Target velocity** of the selected joint (e.g. 90 degrees/second). The robot starts moving. Try to do the same with the robot's other joint/motor, then hit the stop button.

Double-click the child script icon attached to Bob in the scene hierarchy to inspect or modify Bob's main program.

By default, the simulation will run in real-time, if not otherwise constrained by the exercises' script. A client application (e.g. an exercise's script) can also add visual items to the scene programmatically, such as the planned route for the robot.

4.3 Ghost models

What you will not see in the scene are two *ghost* models of the robot: the ghost models are *dummy* robot models that can be shown/hidden to indicate a desired target pose of the robot for instance. When simulation is not running, the ghosts are attached to the robot, but at simulation start, they are detached from it, and a client application (e.g. some of the exercises' scripts) can toggle their visibility state, and change their pose. You can also manually toggle the visibility of the ghosts by opening their model properties dialog (double-click the *bullet* icon on the left of *GhostBob* or *GhostBobTarget* in the scene hierarchy).

5 Running the exercises

In order to test that everything is set-up properly, please do the following:

1. Unpack exercise zip folder to some *local_folder*.
2. Launch CoppeliaSim and load the scene file *local_folder*/scene/mooc_exercise.ttt (via *File*, *Open scene*).
3. Start MATLAB, navigate to *local_folder*/code/common/vrep/ and execute the file test.m.
4. CoppeliaSim should start the simulation and perform various taks. Eventually, a loop in `test.m` tells CoppeliaSim to simulate 20 timesteps.
5. You can send different commands to *Bob*, e.g. you can add **bob_setWheelSpeeds(connection, 10, 10);** in the loop to set the left and right wheel speeds to 10 rad/s.

If you do not receive any errors and CoppeliaSim performs the simulation it means that your installation is complete and working.

Further details and background information

The code that sets-up a remote API client, required to perform Matlab/Octave - CoppeliaSim communication, is located in the **`vrep`** folder. Notice following main functions:

- **simulation_setup**: adds the appropriate folders to the search path, so that Matlab/Octave will be able to locate the remote API libraries specific to your platform/setup (located in the **`libs`** folder). The function will also load the library, if not already previously loaded.
- **simulation_openConnection**: opens a communication line to CoppeliaSim. Make sure that CoppeliaSim is launched and the correct scene is loaded (i.e. **`scene/mooc_exercises.ttt`**).
- **bob_init**: retrieves static data related to *Bob* from the scene, and starts to stream data from CoppeliaSim to the remote API client (i.e. Matlab or Octave).
- **simulation_start**: requests CoppeliaSim to start the simulation.

- **simulation_stop**: requests CoppeliaSim to stop the simulation.
- **simulation_closeConnection**: closes a communication line to CoppeliaSim previously opened via **simulation_openConnection**.
- **test**: Implements some basic CoppeliaSim functionalities.

Above functions, and several others in the **vrep** folder, can be called irrespective of whether you are running Matlab or Octave (internally they will branch to different code appropriately). Make sure your OS is not blocking CoppeliaSim or the remote API via its firewall. If you are able to run **test** file, you are good to go.

6 Troubleshooting

6.1 Troubleshooting the CoppeliaSim- Matlab connection

In case you run into problems or are having difficulties establishing a connection to CoppeliaSim, remember that the Matlab remote API client requires following items in its path, or current Matlab folder:

- **remApi.m**
- **remoteApiProto.m**
- the remote API client library for Matlab (**remoteApi.dll**, **remoteApi.so** or **remoteApi.dylib**). Make sure to select the correct library, depending on your platform and architecture (the 32bit library will not work with the 64bit Matlab, and vice-versa).

Above files can be found in the **vrep/matlab** and **libs/matlab** folders respectively, or in CoppeliaSim's `programming/remoteApiBindings/matlab` and `programming/remoteApiBindings/lib` folders. Make sure to also read this section if needed.

If you see the following error messages "There was an error loading the library", "in remApi (line 554)", or "Error using loaddefinedlibrary" from MATLAB console. This may happen due to input locales and invisible characters within your MATLAB path string (e.g., /user/down loäd/exercise 1). If you are using MATLAB with the default locale (i.e., en_US_POSIX.US-ASCII, also known as C locale), please make sure there are no space and characters other than English in the path.

6.2 Troubleshooting the CoppeliaSim- Octave connection

In case you run into problems or are having difficulties establishing a connection to CoppeliaSim, remember that the Octave remote API client requires following items in its path, or current Matlab folder:

- **remApiSetup.m**
- the remote API client library for Octave (**remApi.oct**). Make sure to select the correct library, depending on your platform and architecture (the 32bit library will not work with the 64bit Octave, and vice-versa).

Above files can be found in the **vrep/octave** and **libs/octave** folders respectively, or in CoppeliaSim's `programming/remoteApiBindings/octave/octave` and `programming/remoteApiBindings/octave/lib` folder. Make sure to also read this section if needed.

6.3 Troubleshooting CoppeliaSim

If you are runing CoppeliaSim on one of the supported platforms (see further up), then you might still experience problems, mainly related to your graphic card or graphic card driver. Make sure to always use the official drivers, and that they are up-to-date. If the openGl rendering of the scene is slow, faulty, or if the vision sensors do not operate correctly (e.g. the sensor that is used to display the map of the terrain in the floating view of the **`scene/mooc_exercises.ttt`** scene), then:

- try selecting a different item in the **FBO type** section of the openGL settings dialog.
- try selecting a different item in the **Offscreen context type** section of the openGL settings dialog.
- make sure you are not running a debug mode (all items should be unchecked in [menu bar –> Help –> Debug]), since that might drastically slow down the simulation.

6.4 Troubleshooting on MacOS

- If you can't see messages even though you use `print` function on MacOS, the messages are displayed only once CoppeliaSim quits, and sometimes, for obscure reasons. What you can do is open a coppeliaSim from your terminal, i.e., "coppeliaSim.app/Contents/MacOS/coppeliaSim", and type "coppeliaSim" to run it. All messages should be printed via the terminal (not CoppeliaSim console).
- In case, you have experienced periodic crashing (Segmentation fault: 11) on MacOS, try to run CoppeliaSim from a terminal (not clicking CoppeliaSim icon from GUI). This happens on Mac Sierra 10.12.3 (16D32) with CoppeliaSim EDU (ver. 3.3.2, rev. 3, 64bits serialization ver. 18). Symptoms were periodic crashing (every 2∼5 mins) and nothing appears in the model browser.
- If you face the following error message "The model folder was not found." and can't see the model browser (the leftmost panel) from CoppeliaSim. This may happen because your system has automatically added attributes to all CoppeliaSim files. A possible workaround is running "sudo xattr -r -d com.apple.quarantine *" from your terminal in the CoppeliaSim folder (e.g., /Applications/CoppeliaSim_EDU_V4_0_0_Mac).

7 Appendix

7.1 Further details about CoppeliaSim connection to client applications

CoppeliaSim can be programmed/controlled via many different means. One of them are the child scripts previously mentioned. Those are used in the **`scene/mooc_exercises.ttt`** scene to handle everything that is local to CoppeliaSim (i.e. inside of CoppeliaSim).

CoppeliaSim offers also several ways to control a simulation from an external application (i.e. a client application). One of them is the remote API.

Some of the Matlab and Octave exercises' scripts have the ability to connect to CoppeliaSim via the remote API, and to control the simulation or Bob. By default, they will connect to the local port 19997. On the CoppeliaSim side, a server service is normally automatically started on that port at CoppeliaSim start-up. That behaviour and some other settings can be changed in CoppeliaSim's folder, by modifying the file **`remoteApiConnections.txt`**. You can also modify that file in order to debug the connection (set the item *portIndex1_debug* to **true**). In case you

run an older version of CoppeliaSim, or if you have difficulties connecting via that method, the **`scene/mooc_exercises.ttt`** scene will also start a temporary remote API server service at simulation start on port 19999, and you could try to connect to that one instead. Refer to the function call **simExtRemoteApiStart(19999)** in the child script attached to Bob.

8 From here on

Refer to the different exercises. Refer also to following items for more information:

- CoppeliaSim website
- CoppeliaSim user manual
- CoppeliaSim overview
- CoppeliaSim forum