A Software Installation

Since all of the software is freely available on GitHub, you can always find a build section within the provided readme file there. The links to the respective GitHub folders are provided within each following section. For the case of an error that may occur during the build procedure, do no hesitate to open up an issue there. I will then help you to get the software running on your device.

A.1 Build the Pattern Generator

The pattern generator can be found on GitHub at the following link. Proceed as described below.

1. Make sure all dependencies are installed. The pattern generator only requires the necessary dependencies (section A.5.1). For communication with the real robot, one further needs the real robot and simulation dependencies (section A.5.2, and the simulation models A.4). For the deep learning support, it is further required to have the deep learning dependencies installed (section A.5.3).

2. Clone the pattern generator from GitHub with
   `https://github.com/mhubii/nmpc_pattern_generator.git`

3. Then do one of the following steps

   a) To just build the pattern generator, do
   ```
   mkdir build && cd build
   cmake ..
   make
   ```

   b) To build the pattern generator with communication to the real robot or the simulation do
   ```
   mkdir build && cd build
   cmake -DCMAKE_BUILD_WITH_YARP=ON
   make
   ```

   c) To build the pattern generator with deep learning support do
   ```
   mkdir build && cd build
   cmake -DBUILD_WITH_LEARNING=ON \
   -DCMAKE_PREFIX_PATH=/absolute/path/to/libtorch ..
   make
   ```

4. This is not necessary, but you can then install or uninstall the pattern generator with
   `make install`
   `make uninstall`

The pattern generator comes with some tests and examples. They can be executed as follows

1. To run the tests do
   `cd build/bin`
   `./pattern_generator_tests`

2. To run an example do
   `cd build/bin`
   `./nmpc_pattern_generator_example`

3. The results may be visualized with
   `cd plot`
   `python plot_pattern.py`

A.2 Build the Android Joystick App

The Joystick app can be found on GitHub at the following link. Proceed as described below.

1. Clone the Android app from GitHub with
   `git clone https://github.com/mhubii/ijoy.git`

2. Copy the `.apk` file that you cloned from GitHub to your Android smartphone.

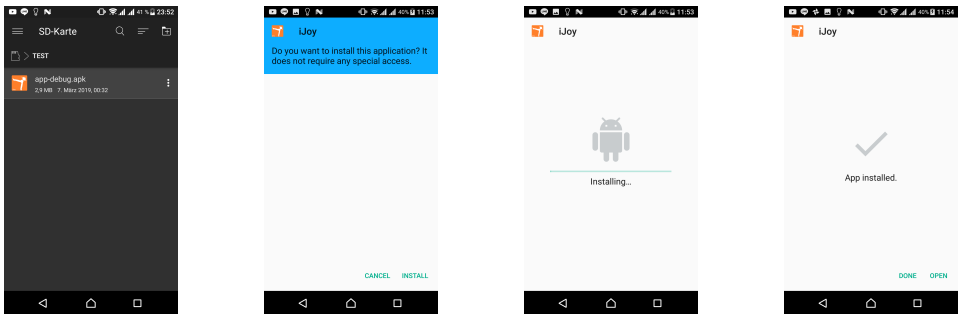
3. Make sure your device allows installation of apps from unknown sources. Under Android 7:

   a) Go to settings and open lock `screen&security`.

   b) Find entry `unkown sources` and enable it.

4. Find the `.apk` using the file browser of your choice and execute it (figure A.1 (a) and (b)).

5. Follow the on screen instructions and wait for the installation to finish (figure A.1 (c) and (d)).



(a) File browser. (b) Menu. (c) Installing. (d) Finished.

Figure A.1: Installation process.

A.3 Build Proximal Policy Optimization

The proximal policy optimization can be found on GitHub at the following link. Proceed as described below.

1. Make sure the C++ API of PyTorch got install as described in section A.5.
2. Clone proximal policy optimization from GitHub with
   `git clone https://github.com/mhubii/ppo_libtorch.git`
3. Then do
   `mkdir build && cd build`
   `cmake -DCMAKE_PREFIX_PATH=/absolute/path/to/libtorch ..`
   `make`

You can then train and evaluate the neural network as described below.

1. Train the neural network with
   `cd build`
   `./train_ppo`
2. Test the trained neural network with
   `cd build`
   `./test_ppo`
3. Visualize the results with
   `python plot.py`

A.4 Build Simulation Models

The simulation models can be found on GitHub at the following link. Proceed as described below.

1. Clone the models from GitHub with
   `git clone https://github.com/mhubii/gazebo_models.git`

2. You can either install the models to a location that Gazebo knows, or update the path, where Gazebo is looking for models.

3. To update the path add following line to your `bashrc`
   `export GAZEBO_MODEL_PATH=<>/gazebo_models:$GAZEBO_MODEL_PATH`
   Therein, replace `<>` by the location to which you cloned the repository.

4. To install the models do
   `mkdir build && cd build`
   `cmake -DCMAKE_INSTALL_PREFIX=~/.gazebo ..`
   `make install`

5. To uninstall the models do
   `cd build`
   `make uninstall`

A.5 Build Third Party Software

The software that was developed as part of this thesis has some third party dependencies, of which not all are necessary but required for some additional features.

A.5.1 Necessary Dependencies

These dependencies need to be installed.

Eigen

1. The pattern generator is based on the blazingly fast Eigen library. To install it do
   `sudo apt install libeigen3-dev`

2. You may need to create a symbolic link
   `sudo ln -s /usr/include/eigen3/Eigen/ /usr/include/`
   `sudo ln -s /usr/include/eigen3/unsupported/ /usr/include/`

qpOASES

To solve the sequential quadratic program, we need to install qpOASES. Please follow the install instructions, or head on as described below

1. Download qpOASES
   `wget https://www.coin-or.org/download/source/qpOASES/qpOASES-3.2.1.zip`
   `unzip qpOASES-3.2.1.zip`
   `cd qpOASES-3.2.1`

2. Now since we want a shared library, in the `CMakeLists.txt` change
   `ADD_LIBRARY(qpOASES STATIC ${SRC})` to
   `ADD_LIBRARY(qpOASES SHARED ${SRC})`
   Then proceed with
   `mkdir build && cd build`
   `cmake ..`
   `make`
   `sudo make install`

YAML

The configurations are read in using the YAML file format. Run the command

1. `sudo apt install libyaml-cpp-dev`

A.5.2 Real Robot and Simulation Dependencies

To run the NMPC generator on a real robot or the simulation, we will need to install some more dependencies.

Gazebo

The simulation environment can be installed according to the install instructions. The main steps are

1. `sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable"\`
   `"`lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'`
   `wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -`
   `sudo apt-get update`
   `sudo apt-get install gazebo9`
   `sudo apt-get install libgazebo9-dev`

RBDL

The rigid body kinematics are solved with RBDL. To install RBDL, do

1. `hg clone https://bitbucket.org/rbdl/rbdl`
   `cd rbdl`

```
hg checkout dev
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE=Release \
-DRBDL_BUILD_ADDON_URDFREADER=ON ..
```

YARP

Additionally, for communicating with the real robot, or the simulation, we need YARP. To install YARP, follow the installation instructions, or head on as described below

1. ```
   git clone https://github.com/robotology/yarp.git
   cd yarp && mkdir build && cd build
   ```

2. If you have previously installed Anaconda, YARP may complain here. Go and install OpenCV within your Anaconda distribution
   ```
   # activate your anaconda environment, if you followed the
   # instructions before in PyTorch, do conda activate py37_torch
   conda install opencv
   ```

3. Then do
   ```
   cmake -DOpenCV_DIR=$HOME/anaconda3/envs/py37_torch/share/OpenCV ..
   make
   sudo make install
   ```

Gazebo YARP Plugins

Plugins for Gazebo are used to clone the behavior of the real robot into the simulation environment. Proceed as below

1. ```
   git clone https://github.com/robotology/gazebo-yarp-plugins.git
   cd gazebo-yarp-plugins
   mkdir build && cd build
   cmake -DCMAKE_INSTALL_PREFIX=$HOME/gazebo-yarp-plugins ..
   make
   make install
   ```

2. Next, you need to tell Gazebo where to find the plugins, therefore add following to the `bashrc`
   ```
   export GAZEBO_PLUGIN_PATH=${GAZEBO_PLUGIN_PATH}
   :$HOME/gazebo-yarp-plugins/lib
   ```

Gazebo Models

The models are used within the simulation environment Gazebo. See section A.4.

NCurses

For the visualization of the control pannel, we need to install ncurses, do

1. `sudo apt install libncurses5-dev`

A.5.3 Deep Learning Dependencies

To learn Nonlinear Model Predictive Control or simple navigation on top of Nonlinear Model Predictive Control, we will need to install PyTorch. For PyTorch to work in combination with RBDL, we need a source installation. Please checkout this gist to figure out how to perform a clean setup.

B Startup and Shutdown Procedures

In this chapter, we will briefly document how to run the software that was developed within this thesis. In section B.1, we will introduce how the Heicub robot is supposed to be started, while in the subsequent section B.2, we will explain how to run the robot within the simulation environment Gazebo. Furthermore, once the robot has been started, either in real or in simulation, it is possible to use the provided pattern generator for control in realtime with the terminal interface or the Android joysitck app. Both possibilities will be explained in section B.3. Finally, a short demonstration for the behavioral augmentation is given in section B.4.

B.1 Real Robot Startup

1. Turn on the icubsrv (figure B.1), username is `icub`, password is `icub`.

Figure B.1: The icubsrv is the Dell laptop. Below it the switch.

2. Turn on the power suppliers (figure B.2), and keep the red button pressed (figure B.7 (a)). The suppliers should initially provide 13V and 0V, if not, do no proceed.

(a) Turn on these buttons.

(b) Expected initial voltage.

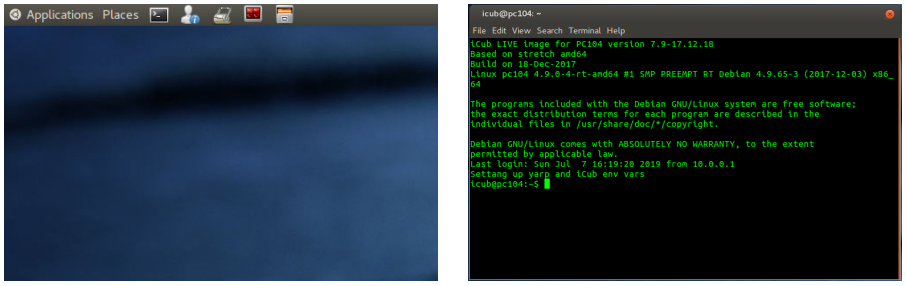
Figure B.2: Power suppliers.

3. Turn on the pc104 (figure B.3 (a)). At this step you may want to wait up to 5 minutes, to give the board computer enough time to start.

(a) Turn on the CPU of pc104. (b) Turn on the motors of Heicub.

Figure B.3: Heicub's switches.

4. On the icubsrv, connect via ssh to pc104. Therefore, click on the highlighted symbol within figure B.4 (a). If it fails to connect, turn off the CPU, and go back to the previous step.



(a) Run ssh to connect to pc104. (b) Terminal to pc104.

Figure B.4: Connect to pc104. TODO highlight symbol

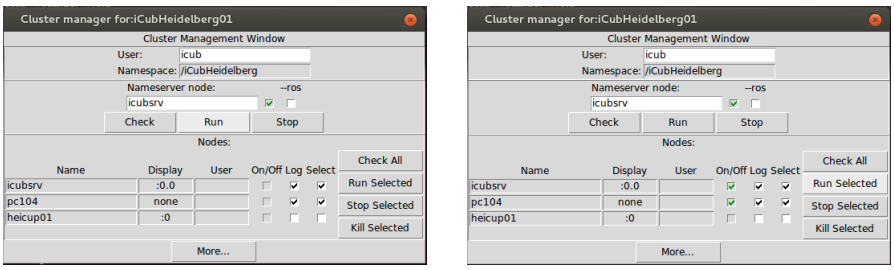
5. Run the cluster manager from a new terminal on the icubsrv, therefore do
   `cd /usr/local/src/robot/icub-main/build-pc104/bin`
   `python icub-cluster.py`

6. Within the cluster manager, run the yarp name server, and then yarp on all other devices (figure B.5 (a), then (b)).

7. We can now turn on the motors and the cameras, as well as to connect our own laptop, which is explained in the following sections.

B.1.1 Start the Motors

If the real robot is up and running (section B.1), you can now start Heicub's motors. Therefore proceed as described below.



(a) Run the yarp nameserver. (b) Run yarp on other devices.

Figure B.5: Run yarp. TODO highlight run

1. Turn on the motors (figure B.3 (b)). The power suppliers should now show 13V and 40V. Wait until the blue lights at Heicub stopped blinking (figure B.6).

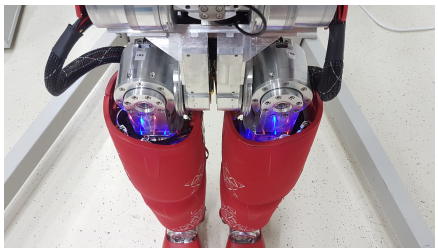


Figure B.6: Blue motor lights.

2. Release the safety button (figure B.7 (b)).

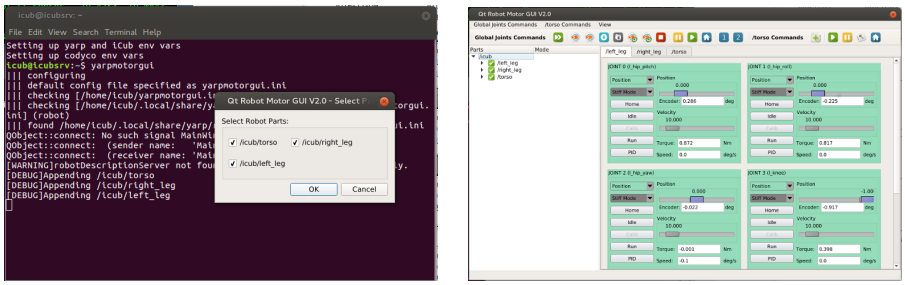
(a) Pressed safety button. (b) Released safety button.

Figure B.7: Safety button.

3. Within your terminal to pc104 (figure B.4 (b)) run the command
   `yarprobotinterface`
   This will run all the motor drivers and connect them to the yarp network.

4. If no errors occured, we can now run the yarpmotorgui to play around with the motors. This step is not necessary. To run the yarpmotorgui, open a new terminal on the icubsrv, and run the command

`yarpmotorgui` (figure B.8 (a))
Then, press `OK`. The motors can now be manipulated from within the GUI (figure B.8 (b)), e.g. by clicking on the house, which will bring all motors to the home position.



(a) Run the yarpmotorgui. (b) The yarpmotorgui.

Figure B.8: Accessing the motors.

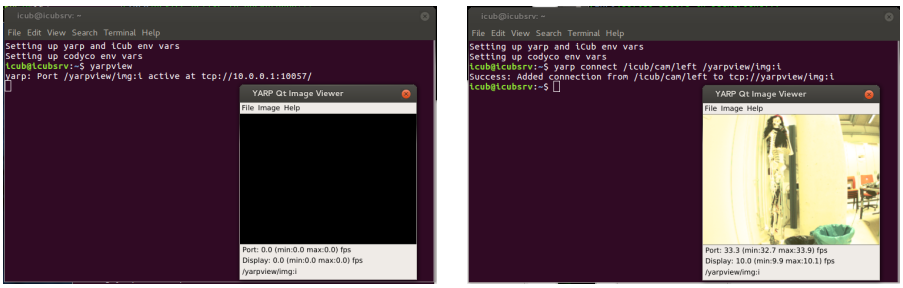
5. The motors may need to be calibrated when the robot runs for a long time. Therefore, lift up the robot, and bring it to the home position via the yarpmotorgui. Then, start a new terminal on pc104 and run the command
`yarp rpc /wholeBodyDynamics/rpc`
In the interface that will open up, type
`calib all 300`.

B.1.2 Start the Cameras

If the real robot is up and running (section B.1), you can now start Heicub's cameras. Therefore proceed as described below.

1. Within a new terminal to pc104 (figure B.4 (a)), do
`cd ./local/share/yarp/robots/iCubHeidelberg01/camera`
Then, start the camera via
`yarpdev --from dragonfly2_config_left.ini`
This will run the left camera and connect it to the yarp network. Repeat the above steps for the right camera, but replace left by right within the `.ini` file.

2. If no errors occured, we can now run a yarpview to see what Heicub sees. This step is not necessary. To run a yarpview, open a new terminal on the icubsrv, and run the command
`yarpview` (figure B.9 (a))
Then, connect a camera to the yarpview. Therefore, open a new terminal on the icubsrv and run the command
`yarp connect /icub/cam/left /yarpview/img:i` (figure B.9 (b))
You should now see an image.



(a) Run a yarpview.

(b) Connect a camera to the yarpview.

Figure B.9: Accessing the cameras.

B.1.3 Connect your own Laptop

Make sure you installed YARP, as described in section A.5. You can then connect your own laptop via ethernet to the same network as Heicub. Therefore, proceed as described below.

1. Connect a LAN cable to the switch to which the icubsrv is connected as well (figure B.1). You will then get assigned an IP address within the same domain as the icubserver, such as `10.0.0.x`. Check this by running `ifconfig`. The IP address of the icubsrv is `10.0.0.1`, while that of the pc104 is `10.0.0.2`.

2. Check the connection by pinging the icubsrv via `ping 10.0.0.1`. If this works, skip this point, otherwise you can create a manual connection. Therefore, search for `Network Connections` among your applications and open it. Then, click `Add`. If it is not set already, check the hardware address by running `ifconfig`, it should show something similar to `HWaddr 9C:EB:E8:B2:AB:27`, choose this as your device. Then go to IPv4 settings and set the IP address to `10.0.0.3`, the netmask to `24`, and the gateway to `10.0.0.255`. Then press save.

3. If you connected successfully, open a terminal on your device and run
   `yarp detect --write`
   If it does not find the running yarpserver, manually configure the connection via the following commands from a terminal
   `yarp conf 10.0.0.1 1000`
   `yarp namespace /iCubHeidelberg`
   `yarp detect`

B.2 Simulated Robot Startup

Make sure you installed Gazebo, YARP, and the Gazebo YARP plugins as described in section A.5. Also, you need to have the simulation model installed, which is

described in section A.4. When these requirements are satisfied, then proceed as described below.

1. Open a terminal and start YARP via
   `yarpserver −−write`

2. Open another terminal and start Gazebo via
   `gazebo −s libgazebo_yarp_clock.so`
   The clock library will thereby synchronize the YARP clock, and the Gazebo clock.

3. In Gazebo, go to the `Insert` bar, and insert `heicub_without_weights`.

B.3 Start the Pattern Generator

Make sure you installed the pattern generator library as described in section A.1. The robot needs to be running, either in real (section B.1, and section B.1.1) or in simulation (section B.2). By construction, the startup procedure for the simulation and the real robot is the same. You can choose to control the robot via the terminal, or the provided Android app. Both possibilities are described below in section B.3.1, and section B.3.2, respectively.

B.3.1 Control via the Terminal

The terminal user interface comes with the pattern generator, so there is no additional software that needs to be installed. Proceed as described below.

1. Open a new terminal and go to the shell scripts within the pattern generator folder via
   `cd nmpc_pattern_generator/sh`
   On the icubsrv, this folder is located at `/usr/local/src/robot`. Then, run the keyboard user interface via
   `sh run_keyboard_user_interface.sh`

2. The shell script will then ask you whether to run the robot in real or in simulation, write `y` or `n` and press enter (figure B.10 (a)).

3. The shell script will then ask you for the mode to run in. Write `uc` and press enter (figure B.10 (a)).

4. The user interface should now show up and explain how to control the robot (figure B.10 (b)).

(a) Select the mode to run the pattern generator in.

(b) Keyboard user interface.

Figure B.10: Run the keyboard user interface.

B.3.2 Control via the Android Joystick App

Make sure you installed the Android joystick app as described in section A.2. Proceed as described below.

1. Open a hotspot from your phone and connect the icubsrv or your laptop to it.

2. Open a new terminal and go to the shell scripts within the pattern generator folder via
   `cd nmpc_pattern_generator/sh`
   On the icubsrv, this folder is located at `home/icub/TODO`. Then, run the app user interface via
   `sh run_app_user_interface.sh`

3. The shell script will then ask you whether to run the robot in real or in simulation, write `y` or `n` and press enter (figure B.11 (a)).

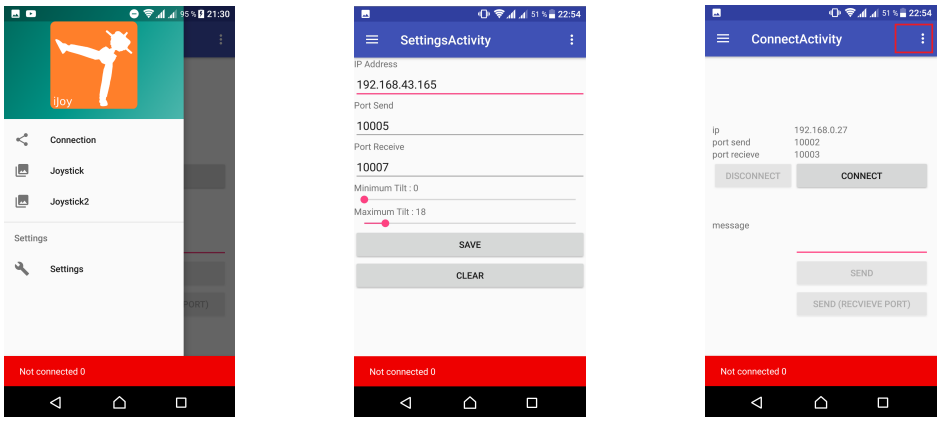
4. The user interface should now show up and explain how to control the robot (figure B.11 (b)).

(a) Select the mode to run the pattern generator in.

(b) App user interface.

Figure B.11: Run the app user interface.

5. Open the Android app on your smartphone. Within the app, choose settings from the navigation drawer (figure B.12 (a)). In the settings activity (figure B.12 (b)), choose the IP address and the ports as proposed by the app user interface (figure B.11 (b)).



(a) Navigation drawer. (b) Settings activity. (c) Connect activity.

Figure B.12: Connect the app to the computer.

6. Now go to the connection activity (figure B.12 (c)), and press connect. You should now be connected.

7. Go to the app user interface on the laptop (figure B.11 (b)), and press `r` to run the pattern generator.

8. Within the navigation drawer, choose from one of two joysticks to control the robot.

B.4 Start the Behavioral Augmentation Demo

There is a trained neural network available on heicub01 to run a demonstration of the behavioral augmentation as trained in section 5.4. The network will navigate Heicub towards a fire extinguisher, and look around if it does not see one. Make sure the robot and its cameras are running (section B.1, section B.1.1, and section B.1.2). Proceed as described below.

1. Login to heicub01, username is `icub`, password it `icubheidelberg01`.

2. Make sure that YARP is running on heicub01 (figure B.5 (b)). Therefore, select heicub01 and click `Run Selected`.

3. On heicub01, open a new terminal and go to the shell scripts within the pattern generator folder via
   `cd /home/icub/Documents/nmpc_pattern_generator/sh`

Then run the keyboard user interface with
`sh run_keyboard_user_interface.sh`

4. The shell script will then ask you whether to run the robot in real or in simulation, write `n` and press enter (figure B.10 (a)).

5. The shell script will then ask you for the mode to run in. Write `ba` and press enter.

6. The user interface should now show up (figure B.10 (b)). Press `r` to run the pattern generator. The robot will now be controlled by the trained neural network.

B.5 Real Robot Shutdown

Before you shutdown the robot, make sure it is in a safe position, that is, lift it up. Proceed as described below.

1. Lift the robot from the floor.

2. Close all running applications. `Ctrl+C` the yarprobotinterface and the camera interfaces.

3. Type `sudo poweroff` in a terminal that is connected to pc104.

4. Turn off the motors.

5. Turn off the CPU of pc104.

6. Turn off the power suppliers.

7. Just to be sure, press the red button.