Configuring deployment environment and project steps

Configuring deployment environment

Dual boot ubuntu22 with Win

Install according to your computer's configuration and refer to online resources for guidance.

We recommend a dual boot, but you can try virtual machine yourself.

Install Python 3.6+ and the required packages:

1. Check Your Python Version

First, check if Python 3.6 or later is installed:

```
python3 --version
```

If it is not installed, install it according to your operating system:

```
sudo apt update && sudo apt install python3 python3-pip
```

2. Upgrade `pip`

Ensure that `pip` is up to date:

```
python3 -m pip install --upgrade pip
```

3. Install Required Packages

Run the following command to install all the required libraries:

```
pip install numpy pyserial termcolor matplotlib scipy osqp numba dynamixel-sdk posix_ipc
```

---

4. Verify Installation

After installation, verify that all packages are correctly installed by running:

```
python3 -c "import numpy, serial, termcolor, matplotlib, scipy, osqp, numba, dynamixel_sdk, posix_ipc; print('All packages installed successfully')"
```

If you see the message **"All packages installed successfully"**, everything is set up correctly.

---

5. Troubleshooting

- If you encounter permission issues, try using `pip` with `-user`:

  ```
  pip install --user numpy pyserial termcolor matplotlib scipy osqp numba dynamixel-sdk posix_ip
  ```

- If installation fails due to missing dependencies, install system dependencies:
  - **Ubuntu/Debian:**

    ```
    sudo apt install build-essential python3-dev
    ```

Install PyBEAR

PyBEAR is the Python driver for BEAR series actuators from Westwood Robotics:

---

1. Clone the Repository

First, download the PyBEAR repository from GitHub:

```
git clone https://github.com/Westwood-Robotics/PyBEAR.git
cd PyBEAR
```

2. Modify Permissions for Serial Port Access

Ensure your user has the correct permissions to access the serial port:

```
sudo chown -R $USER /usr/local
sudo usermod -a -G dialout $USER
```

Then, restart your terminal or log out and log back in to apply the changes.

3. Install PyBEAR Using `pip`

Since PyBEAR uses Python 3, install it using `pip3`:

```
pip3 install .
```

4. Install Dependencies

Ensure that `pyserial` and `numpy` are installed:

```
pip3 show pyserial numpy
```

If they are missing, install them:

```
pip3 install pyserial numpy
```

5. Set Up Udev Rules for USB2BEAR

If you are using the **Boosted USB2BEAR/USB2RoMeLa** device, configure **udev rules**:

1. Move the `00-WestwoodRobotics.rules` file to `/etc/udev/rules.d/`:

```
sudo cp 00-WestwoodRobotics.rules /etc/udev/rules.d/
```

2. Reload the udev rules:

```
sudo udevadm control --reload
```

3. Now, when you plug in a **Boosted USB2BEAR/USB2RoMeLa** device, a symlink will be created under `/dev/` with its serial number.

---

6. Verify Installation

To confirm that PyBEAR is installed correctly, try importing it in Python:

```
python3 -c "import pybear; print('PyBEAR installed successfully')"
```

If you see **"PyBEAR installed successfully"**, then everything is working properly.

Download and unzip bruce_clover.zip in your ubuntu

```
unzip bruce_clover.zip
```

Install Gazebo using Ubuntu packages

1. Installation Gazebo

https://classic.gazebosim.org/tutorials?tut=install_ubuntu

The latest version, Gazebo11, is recommended.

2. Link BRUCE model to Gazebo

```
cd bruce_clover
mkdir ~/.gazebo/models
ln -s $PWD/Simulation/models/bruce ~/.gazebo/models/bruce
```

3. Add BRUCE Gym plugins to Gazebo

```
cd bruce_clover
cp -r Library/BRUCE_GYM/GAZEBO_PLUGIN ~/.gazebo
echo "export GAZEBO_PLUGIN_PATH=$GAZEBO_PLUGIN_PATH:~/.gazebo/GAZEBO_PLUGIN"  >>  ~/.bashrc
source ~/.bashrc
```

BRUCE Gym

BRUCE Gym is the simulation environment for bruce_clover. It utilizes a custom library to interact BRUCE in Gazebo using python.

Running in a simulated environment

Indicating Running in Simulation

Modify line 14 of `Play/config.py`.

```
SIMULATION = True  # if in simulation or not
```

Compiling Code Ahead of Time

Make sure to precompile all the Python scripts in the `Library/ROBOT_MODEL` and `Library/STATE_ESTIMATION` folders before use.

```
cd bruce_clover
python3 -m Library.ROBOT_MODEL.BRUCE_DYNAMICS_AOT
python3 -m Library.ROBOT_MODEL.BRUCE_KINEMATICS_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ORIENTATION_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ESTIMATION_CF_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ESTIMATION_KF_AOT
```

Loading BRUCE model in Gazebo

```
cd bruce_clover/Simulation/worlds
gazebo --verbose bruce.world
```

If everything goes well, BRUCE should be fixed in the air in its nominal posture.

ATTENTION: BRUCE Gym is built on Ubuntu 22.04.1 ×86_64. Any lower version might need an upgrade of the GNU C libraries, e.g., GLIBC and GLIBCXX. Please refer to the error messages in this regard.

Launch

1. Make all terminals go to bruce_clover folder.

   ```
   cd bruce_clover
   ```

2. In terminal 1, run shared memory modules. Start Gazebo simulation afterwards.

   ```
   python3 -m Startups.memory_manager
   ```

   ```
   python3 -m Simulation.sim_bruce
   ```

3. In terminal 2, start state estimation thread.

   ```
   python3 -m Startups.run_estimation
   ```

4. In terminal 3, start low-level whole-body control thread.

   !!! You must correctly complete Project I before running `low_level.py` to ensure that Bruce can stand and maintain balance.

   ```
   python3 -m Play.Walking.low_level
   ```

5. In terminal 4, start high-level DCM footstep planning thread.

   !!! Incorrect controller design in low_level.py may cause Bruce to fail to walk

   ```
   python3 -m Play.Walking.high_level
   ```

6. In terminal 5, start top-level user keyboard input thread.

```
python3 -m Play.Walking.top_level
```

Got Errors?

1. There is probably another Gazebo process running ...

```
killall gzserver
killall gzclient
```

Note: A video of Bruce standing and walking in simulation can be submitted as a direct result of Project I.

Controlling Real Robots

Note: We've done most of the preparation for controlling the real robot, when your simulation has been completed, it's time to book your experiment!

Compiling Code Ahead of Time

Make sure to precompile all the Python scripts in the `Library/ROBOT_MODEL` and `Library/STATE_ESTIMATION` folders before use.

```
cd BRUCE/bruce_clover
python3 -m Library.ROBOT_MODEL.BRUCE_DYNAMICS_AOT
python3 -m Library.ROBOT_MODEL.BRUCE_KINEMATICS_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ORIENTATION_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ESTIMATION_CF_AOT
python3 -m Library.STATE_ESTIMATION.BRUCE_ESTIMATION_KF_AOT
```

SSH Manual

1. Make sure your laptop and BRUCE's onboard computer share the same network, e.g., you can use a Wi-Fi hotspot.
2. Check the username and ip address (`ip addr`) of the onboard computer.

3. Login to remote server with the password `khadas`. The ip address may vary, e.g.,

```
ssh khadas@khadas.local
ssh khadas@192.168.10.32 or ssh khadas@192.168.10.188
```

Serial Port & MAC Address

Please config the serial port names (of BEAR, Dynamixel, and Pico) and MAC address of your gamepad (if applicable) in `BRUCE_SERIAL_PORT` or `Settings/BRUCE_macros.py` (line 272-283) for your BRUCE before use.

You can check the serial port names by entering the following command in terminal:

```
ls /dev/serial/by-id/
```

Allow Executing Bash/Binary File As Program

1. Go to bruce_clover folder.

```
cd $your_path$/bruce_clover
```

2. Run the following commands.

```
chmod +x Play/bootup.sh
chmod +x Play/bootup_gamepad.sh
chmod +x Play/sim_bootup.sh
chmod +x Play/terminate.sh
chmod +x Startups/startup_setup.sh
chmod +x Startups/usb_latency_setup.sh
```

Startup Setup Before Launch

BRUCE after VERSION 0.0.4 will be automatically configured at bootup with Startup Applications.

1. Go to bruce_clover folder.

```
cd $your_path$/bruce_clover
```

2. Run the bash file.

```
Startups/startup_setup.sh
```

Full Operating

We suggest use terminator as an alternative terminal for Linux since we need to run 7 threads concurrently.

1. Make all terminals go to bruce_clover folder.

```
cd $your_path$/bruce_clover
```

2. In terminal 1, run USB low latency setup and shared memory modules.Start Dynamixel motor thread afterwards.

```
Startups/usb_latency_setup.sh
python3 -m Startups.memory_manager
```

```
python3 -m Startups.run_dxl
```

3. In terminal 2, start BEAR actuator thread.

```
python3 -m Startups.run_bear
```

4. In terminal 3, initialize BRUCE (enter `s`) and then start sense communication thread after BRUCE can stand on the ground on its own.

```
python3 -m Play.initialize
python3 -m Startups.run_sense
```

5. In terminal 4, start state estimation thread.

```
python3 -m Startups.run_estimation
```

6. In terminal 5, start low-level whole-body control thread.

```
python3 -m Play.Walking.low_level
```

7. In terminal 6, start high-level DCM footstep planning thread.

```
python3 -m Play.Walking.high_level
```

8. In terminal 7, start top-level user input thread.

```
python3 -m Play.Walking.top_level
```