Lab - Interacive Robotic — 2h

Cédric Buche

Contents

1 Install

`qiBullet` is a simulation tool enabling users to experiment with robots in virtual environments.

```
$ pip install --user qibullet
```

2 Simulator

The `qiBullet` simulation has been designed to inherit the cross-platform properties of the `PyBullet` Python module and `Bullet` physics engine.

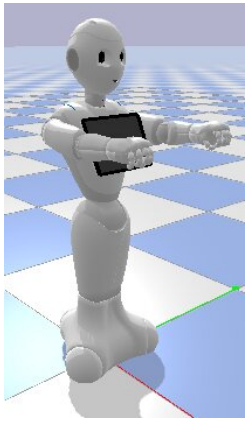
2.1 Setup

The simulation can be launched using a SimulationManager object:

```
from qibullet import SimulationManager

if __name__ == "__main__":
    simulation_manager = SimulationManager()
    client_id = simulation_manager.launchSimulation(gui=True)

    # Your code here...
```

2.2 Spawning a virtual robot

```
from qibullet import SimulationManager
from qibullet import RomeoVirtual
from qibullet import PepperVirtual
from qibullet import NaoVirtual

if __name__ == "__main__":
    simulation_manager = SimulationManager()
    client_id = simulation_manager.launchSimulation(gui=True)
    pepper = simulation_manager.spawnPepper(
        client_id,
        spawn_ground_plane=True)

    nao = simulation_manager.spawnNao(
        client_id,
        translation=[0, 2, 0],
        quaternion=[0, 0, 0, 1])

    romeo = simulation_manager.spawnRomeo(
        client_id,
        spawn_ground_plane=False)

    # Your code here...
```



A virtual robot can also be spawned using the PepperVirtual, NaoVirtual or RomeoVirtual class directly, but the virtual environment needs to be launched first:

```
pepper = PepperVirtual()
nao = NaoVirtual()

pepper.loadRobot(
    translation=[0, 0, 0],
    quaternion=[0, 0, 0, 1],
    physicsClientId=client_id)

nao.loadRobot(
    translation=[0, 2, 0],
    physicsClientId=client_id)

# Your code here...
```

2.3 Create 3D oject

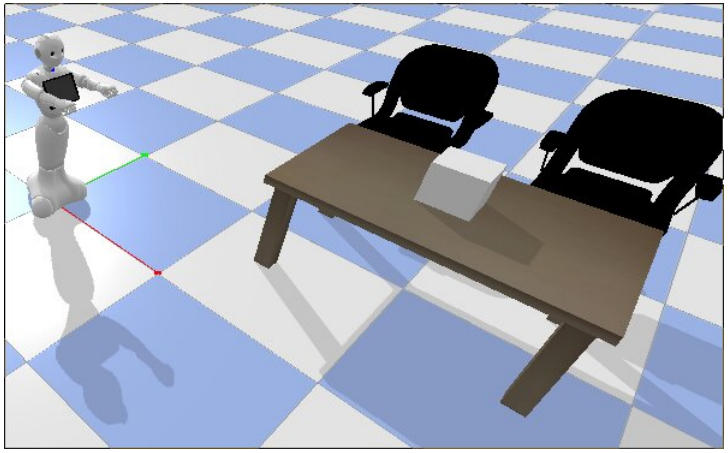
```
    p.connect(p.DIRECT)
    cube_visual = p.createVisualShape(p.GEOM_BOX, halfExtents=[0.125,0.125,0.125])
    cube_collision = p.createCollisionShape(p.GEOM_BOX, halfExtents=[0.25,0.25,0.25])
    cube_body = p.createMultiBody( baseMass=0, baseCollisionShapeIndex=cube_collision,
baseVisualShapeIndex=cube_visual, basePosition = [2,1, 0.725])
```

2.4 Load 3D oject

Unified Robot Description Format (URDF) file is used to describe the model of a virtual robot. The different links of the model, their masses, inertia matrices and thejoints connecting them are extracted from this file by theengine. Mesh files are associated to each link, allowing theengine to render the visual aspect of the robot model and toperform collision checking

```
    p.loadURDF("./urdf/table/table.urdf", basePosition = [2,1,0], globalScaling = 1)
    p.loadURDF("./urdf/chair/chair.urdf", basePosition = [3,1,0], globalScaling = 1)
    p.loadURDF("./urdf/chair/chair.urdf", basePosition = [4,1,0], globalScaling = 1)
```

Question :
Develop an environment with 3D objects (loaded from .urdf and created within the code). Add a robot.

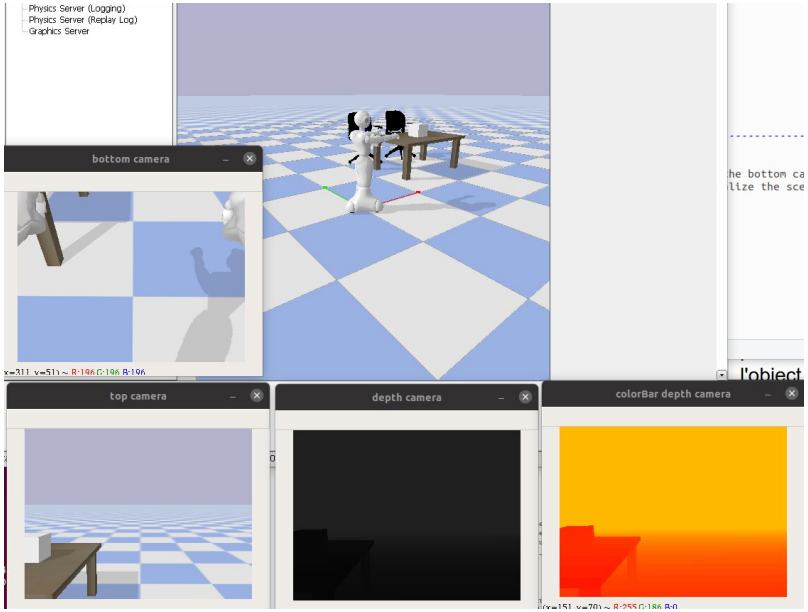


2.5 Sensors

2.5.1 Camera

```
# camera
    handle = pepper.subscribeCamera(PepperVirtual.ID_CAMERA_BOTTOM)
    handle2 = pepper.subscribeCamera(PepperVirtual.ID_CAMERA_TOP)
    handle3 = pepper.subscribeCamera(PepperVirtual.ID_CAMERA_DEPTH)



    try:
        while True:

            img = pepper.getCameraFrame(handle)
            cv2.imshow("bottom camera", img)

            img2 = pepper.getCameraFrame(handle2)
            cv2.imshow("top camera", img2)

            img3 = pepper.getCameraFrame(handle3)
            cv2.imshow("depth camera", img3)

            filename="ImageDepth.png"
            cv2.imwrite(filename,img3)
            im_gray = cv2.imread(filename, cv2.IMREAD_GRAYSCALE)
            im_color = cv2.applyColorMap(im_gray, cv2.COLORMAP_HSV)
            cv2.imshow("colorBar depth camera", im_color)

            cv2.waitKey(1)

    except KeyboardInterrupt:
        simulation_manager.stopSimulation(client)
```

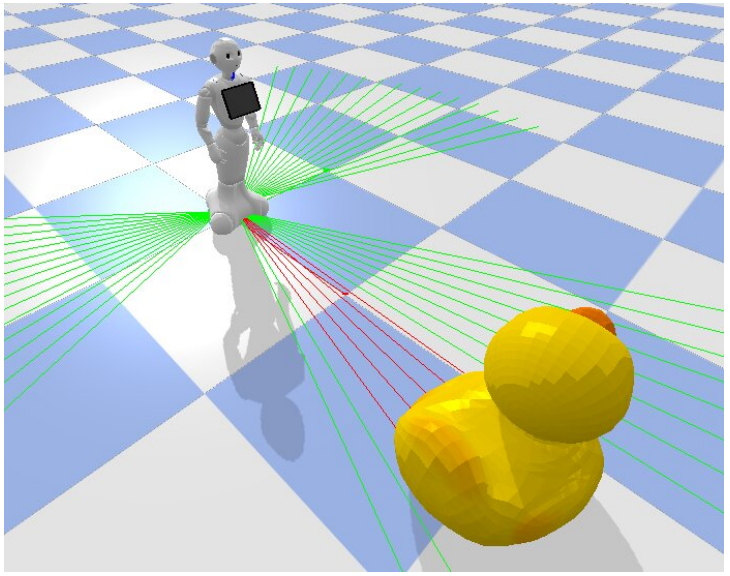
Question :
Use the previous environment by adding the top camera view, the bottom camera view, and the depth view. Modify the image coming from the depth camera to better visualize the scene.



2.5.2 Laser

```
pepper.showLaser(True)
pepper.subscribeLaser()


while True:
    laser_list = pepper.getRightLaserValue()
    laser_list.extend(pepper.getFrontLaserValue())
    laser_list.extend(pepper.getLeftLaserValue())

    if all(laser == 5.6 for laser in laser_list):
        print("Nothing detected")
    else:
        print("Detected")
        pass
```

Question :
Create a new environment with a robot and a 3d object. Use lasers to detect the object. Move the object to test the laser detection.

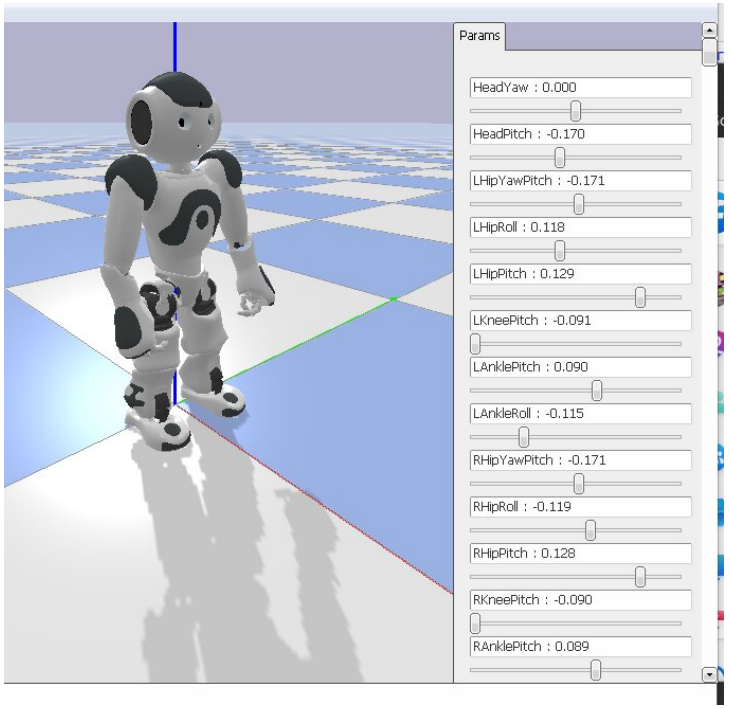


2.6 Actuators

2.6.1 Joint

```
joint_parameters = list()

for name, joint in pepper.joint_dict.items():
    if "Finger" not in name and "Thumb" not in name:
        joint_parameters.append((
            p.addUserDebugParameter(
                name,
                joint.getLowerLimit(),
                joint.getUpperLimit(),
                pepper.getAnglesPosition(name)),
            name))

while True:
    for joint_parameter in joint_parameters:
        pepper.setAngles(
            joint_parameter[1],
            p.readUserDebugParameter(joint_parameter[0]), 1.0)
```

Question :
Create a new environment allwing to play with joint using the previous code. Test with different robots.



2.6.2 Posture

```
pepper.goToPosture("Crouch", 0.6)
time.sleep(3)
pepper.goToPosture("Stand", 0.6)
time.sleep(3)
pepper.goToPosture("StandZero", 0.6)
time.sleep(5)
```

Question :
Test various postures.

2.6.3 Motion

```
pepper.moveTo(3,3,0,frame=2,_async=False)
```

Question :
Test motions.

3 Complex behaviour

Question :
Create a complex test using lasers, camera, 3d obstacles. Robot should move according to obstacles.