1 Introduction

SentiBotics Navigation Software Development Kit (SDK) is designed for researchers and engineers working on autonomous robot navigation. SDK comes with either simulated robot and environment (using Gazebo 7.0.0 simulator), or with optional ready-to-run physical mobile robot reference platform, fully integrated with the kit. This document provides guidelines how to install and use SentiBotics Navigation SDK in simulator. Throughout this tutorial we also assume that the reader is familiar with Robot Operating System (ROS)\(^1\).

2 Configuring development machine

This section provides instructions on how to install SentiBotics Navigation SDK into your computer. SentiBotics Navigation SDK requires that clean Linux Ubuntu 18.04 operating system (OS) is installed, and other recommended system requirements are satisfied, and that testing machine is constantly connected to the internet (persistent internet connection is required for simulation only if trial package is used, in the case of full SDK the internet connection is not required).

2.1 Installing Sentibotics trial

Provided set of scripts installs required software dependencies, sets up the environment variables, installs SentiBotics Navigation SDK trial packages and associated data (e.g. pretrained neural network models, etc.).

- Extract SentiBotics navigation SDK trial into folder.
- Open terminal, go to extraction folder and type (everywhere without sudo):
  
  ```bash
  ./ installer .sh
  ```

  After script is completely executed, close the terminal. Open new terminal and type:

  ```bash
  ./ prepare_environment_gpu .sh
  ```

  for GPU environment (recommended), or

  ```bash
  ./ prepare_environment_cpu .sh
  ```

  for CPU-only environment.

  Close terminal after script is completely executed. In the case of GPU environment also reboot your machine.

- Open new terminal and activate SentiBotics workspace

```
conda activate sentibotics_ws
```

After completing installation we recommend to watch video tutorials (see Sec. 5), which provide an examples of how to use the functionality of SDK.

\(^1\)http://www.ros.org
3 Starting simulator environment

This section describes how to start SentiBotics Navigation SDK simulation software. The following command starts SentiBotics Navigation SDK, Gazebo simulator and Rviz visualization tool:

```bash
roslaunch sentibotics_launch simulator_controller.launch
```

Gazebo simulator GUI Fig. 1 and Rviz Fig. 2 should appear. Press ”Play” button in Gazebo GUI, to begin simulation.

![Figure 1: SentiBotics robot in Gazebo simulator.](image)

4 Sentibotics Navigation SDK functionality

This section describes how to use SentiBotics Navigation SDK functionality. The same interface is used either in simulated or real robot cases.

4.1 Control using joystick

Plug the joystick into USB port of your computer and start joystick node

```bash
rosrun sentibotics_external_joy joy_node
```

You may test it with `jstest` tool, which can be installed from APT repository. `START` button is used for switching between platform ON/OFF. Robot status can be viewed by:

```bash
rostopic echo /sentibotics_platform/platform_status
```

- For robot platform controls see Fig. 3
4.2 Control using behaviour interface

SentiBotics Navigation SDK includes visuomotor trajectory controller learning and executing, object learning and recognition functionality. It can be accessed via ROS interface (publishers, subscribers, services and actions) or included console-like interface, implemented in `sentibotics_behaviour` package. This section provides an examples how to use SDK functionality through aforementioned tool.

4.2.1 Behaviour interface

`Behaviour` interface functions as a glue between lower level modules (e.g. navigation controller training and execution, object recognition, etc.). It provides console-like interface for using SDK functionality and serves as a framework for implementing higher level robot behaviour (e.g. autonomous recharging behaviour, which relies on navigation and object recognition).

In order to start behaviour interface, open another terminal and type

```bash
roslaunch sentibotics_behaviour behaviour.launch
```

4.2.2 Testing pretrained trajectory controller

SentiBotics Navigation SDK includes pretrained controller, which can be executed just after Gazebo simulation is started (the robot should be located in the initial place at the beginning of execution). In order to test it in the behaviour interface type

```python
self.get_trajectory_controllers()
```
Figure 3: Mobile platform controls

```python
self.run_controller(os.path.expanduser("~") + "/.ros/sentibotics_trajectory/trained_models/20181128-163351/model_save_dir/model-20.meta")
```
and wait until the controller starts to drive the robot. In order to stop the controller type:

```python
self.stop_controllers()
```
4.2.3 Trajectory controller training and execution

In the following we provide an example how new trajectory controller can be trained. We recommend to watch video tutorial for trajectory controller training and execution (see Sec. 5) before reading further.

![Image of running behavior commands](image.png)

Figure 4: Example of running behavior commands.

- Start collecting the training data:
  
  ```python
  self.start_direct_data_collection(FOLDER)
  e.g. self.start_direct_data_collection("/tmp/tmp")
  ```

- Drive the robot few times in the same closed trajectory using control pad.

- Stop training data collection:
  
  ```python
  self.stop_direct_data_collection()
  ```

- Train trajectory controller using collected data (input image and corresponding joystick command pairs):
  
  ```python
  self.learn_controller(FOLDER, NUMBER_OF_ITERATIONS)
  e.g. self.learn_controller("/tmp/tmp", 20)
  ```

  After the learning is complete, it will be indicated by informative message in the behaviour console.

- Get learned trajectory controllers:
self.get_trained_controllers()

The list provides all trained controllers. Note, that controller training
timestamp is included into controller path.

- Run trained trajectory controller:

```
self.run_controller(CONTROLLER_CHECKPOINT)
e.g. self.run_controller(os.path.expanduser("\~") +"/\.ros/
sentibotics_trajectory/trained_models
/20181128-163351/model_save_dir/model
-20.meta")
```

- During controller execution correct the robot with the control pad if it nav-
ignates incorrectly. All the corrections are saved and added to the training
set for next training iteration. Aforementioned iterative training process
is repeated until trajectory execution is stable enough.

In order to stop navigation controller type

self.stop_controllers()

4.2.4 Object detection

SentiBotics Navigation SDK object detection package mainly relies on
find_object_2d library. This functionality is essentially the same either in
simulator, or in the case of real robot operating in real environment. Note,
that object recognition engine can recognize only sufficiently large objects with
sufficiently large amount of features (e.g. it can not learn to recognize gray wall
segment, but objects like chairs, doors are ok). Also note, that object recogni-
tion engine can recognize only the same object, which was enrolled into it (i.e.
it cannot recognize all chairs, if trained only on few types of them). In Sec. 5
you can find an example how object learning and recognition process looks like.

4.2.5 Object learning

After running behaviour console

```
roslaunch sentibotics_behaviour behaviour.launch
```

enter directory where learned object instances should stored, e.g.

```
self.set_object_memory_folder('docking_station')
```

Once this directory is specified, run

```
roslaunch sentibotics_object_detection
sentibotics_object_learning.launch
```

and select rectangles from current camera view for enrolling into object recog-
nition engine.
4.2.6 Object detection

Object detection messages are streamed to topic defined by `object_detection_topic` parameter (default value: "/sentibotics_object_detection/detected_objects"). By default, image with detections marked on it are streamed to topic defined by `output_image_topic` local parameter (default value: "/sentibotics_object_detection/output_image").

5 Video tutorials

- Trajectory controller training and execution tutorial (real environment).
- Trajectory controller training and execution (simulator environment).
- Object detection tutorial.

6 Package structure

This section contains information about SentiBotics Navigation SDK package structure.

6.1 Navigation

- `sentibotics_trajectory` - trajectory controller learning and execution.

6.2 Object recognition

- `sentibotics_external_find_object_2d` - Find object 2d library.
- `sentibotics_object_detection` - ROS wrapper for Find object 2d library, and ROS node to run mobilenet SSD object detector on Movidius NCS.

6.3 Message packages

- `sentibotics_navigation_msgs` - navigation messages.

6.4 Higher level behavior

- `sentibotics_behavior` - package implementing higher level behavior and demonstration of robot capabilities.

6.5 Robot description

- `sentibotics_description` - URDF description of SentiBotics robot.

6.6 Robot simulation

- `sentibotics_gazebo` - SentiBotics Gazebo simulator package.
6.7 Other

- `sentibotics_launch` - launch files for various scenarios.
- `sentibotics_common` - common library and configuration parameters.
- `sentibotics_external_joy` - custom joystick package (allows to rescale output).