Obtaining a ROS-Based Face Recognition and Object Detection: Hardware and Software Issues

Petri Oksa[1*], Tero Salminen[1] and Tarmo Lipping[1]

[1] Tampere University, Computing Sciences, Pohjoisranta 11, FI-28100 Pori, FINLAND
petri.oksa@tuni.fi

Abstract. This paper presents solutions for methodological issues that can occur when obtaining face recognition and object detection for ROS-based (Robot Operating System) open-source platform. Ubuntu 18.04, ROS Melodic and Google TensorFlow 1.14 are used in programming the software environment. TurtleBot2 (Kobuki) mobile robot with an additional on-board sensors are used to conduct the experiments. Entire system configurations and specific hardware modifications that were proved mandatory to make out the system functionality, are also clarified. Coding (e.g., Python) and sensors installations are detailed both in on-board and remote laptop computers. In experiments, TensorFlow face recognition and object detection are examined by using TurtleBot2 robot. Results show how objects and faces were detected when the robot is navigating in the previously 2D mapped indoor environment.

Keywords: ROS, Ubuntu, Object detection, Face recognition, 3D sensor, LiDAR.

1 Introduction

ROS is versatile software framework that can also be utilized in many other applications than in conventional robotic solutions. As a noteworthy example, mobile robot equipped with machine learning software platform can bring added value and purposes of use when connected along with environment mapping and autonomous navigation. In recent novel software frameworks, object detection and face recognition are possible to build on an open-source platform.

Google TensorFlow offers free software libraries mainly supported for machine learning application. Today, TensorFlow offers a new opportunity for researchers and developers to utilize it in many ROS-based solutions [1], [2], [3], [4], [5]. In [6] author presents an extensive ROS toolbox for object detection, tracking and face/action recognition with 2D and 3D support enabling the robot to understand the environment. At its simplest way to adapt this, only web camera and ROS computer is needed.

AutoRobo project (Autonomous Robot Ecosystem) consists of an open-source cloud-computing platform, software frameworks and a physical multi-robot environment for the automation or assisting of preprogrammed work processes. Among others, healthcare and hospice operations retain processes which can be assisted by mobile robots. The service architecture model developed in Tampere University, Pori,

consists of a novel mobile cloud robotic platform in the support of patient work. Case study presented in this article is implemented in the system environment illustrated in Fig 1.

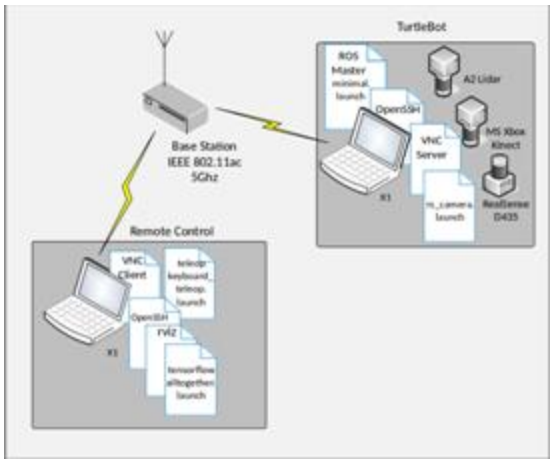


Fig. 1. AutoRobo System Overview

Compatibility of software repository packages to a certain system architecture, ROS topic subscribing, Ubuntu-TensorFlow packages and ROS distribution may cause issues. Guidelines and instructions are available in several Q&A forums and public source repositories for instance in Github/developer pages. Nevertheless, they can propose incompatible for the users ROS distribution and versions of package dependencies [7]. To overcome such difficulties that can emerge system environment represented in Figure 1, guidelines for coding, required software package installations and robot hardware set-up are given. The aim of this research was to find solutions for issues that can arise of open-source object-/face recognition and sensors software installation for ROS robot platform and also find solutions to overcome them.

The rest of the paper is organized as follows. Section 2 provides TurtleBot2 Montado package installation and remarks. Section 3 describes Ubuntu sensors installation for Real Sense and A2 LiDAR. In Section 4 TensorFlow package installation guidelines are given. Section 5 presents experiments considering the whole system functionality. Finally, results and conclusions are discussed in the last Section 6.

2 TurtleBot2 Package Installation and Remarks

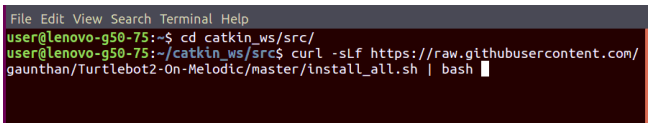
TurtleBot2 Debian installation option is not supported for ROS Melodic on Ubuntu 18.04 distribution. Therefore, only way to install the TurtleBot2 package is to download a package and build it from source code. For previous ROS distributions, like Kinetic, Turtlebot2 package can be installed via Debian packages. In our system environment, ROS Melodic distribution is formerly installed, so a Turtlebot2 package installation follows the source code compiling. The complete ROS Melodic installation instructions can be found on ROS.org web page at http://wiki.ros.org/melodic/Installation/Ubuntu.

The first thing is to build the TurtleBot2 workspace. In terminal, the following command directs to a catkin workspace directory

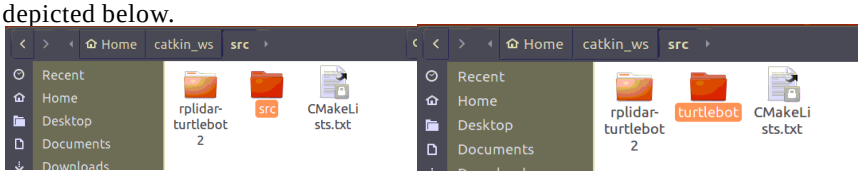
```
$ cd catkin_ws/src
```

After this, the software package should be downloaded from https://github.com/gaunthan/Turtlebot2-On-Melodic. The following command (inside the root of catkin workspace) builds up the running environment for Turtlebot2 [8].

```
$ curl -sLf https://raw.githubusercontent.com/gaunthan/Turtlebot2-
  On-Melodic/master/install_all.sh | bash
```



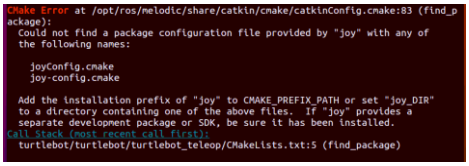
After downloading the package, the 'src' folder should be renamed to 'turtlebot' as depicted below.



The following command (inside the root of catkin workspace) builds up the running environment for Turtlebot2.

```
$ catkin_make
```

In case of the following error, the joystick package should therefore be installed.



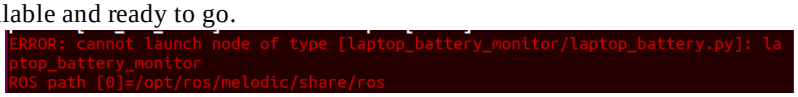
The next two command lines install the missing joystick package

```
$ sudo apt install ros-melodic-joy
$ catkin_make
```

By connecting a Turtlebot2 to the on-board computer and running the launch file below brings up the robot

```
$ roslaunch turtlebot_bringup minimal.launch
```

The following error might occur during the installation, but it does not cause any disadvantages and can be ignored in this phase. The robot should be now fully controllable and ready to go.



If TurtleBot2 is equipped with e.g. Microsoft Xbox Kinect sensor, *.bashrc* must then configured. The following line should be added on it:

```
$ export TURTLEBOT_3D_SENSOR=Kinect
```

3 Sensors Installation and Issues

This section presents the robot on-board sensors installation used in the experiments. Sensors installation procedure is quite straightforward but can be complicated in case of the older ROS-compatible robot types such as TurtleBot2 (Kobuki).

3.1 Real Sense D435

In our robotic platform, Real Sense sensor is dedicated for object detection and face recognition and Kinect for the indoor environment mapping and navigation. We start with ROS driver installation. Note that in addition to sensor driver installation, *librealsense* is also needed.

At first, the system must be updated by typing `sudo apt-get update` in a terminal. Then, software package have to download by running the following command in terminal

```
$ sudo apt-get install ros-melodic-realsense2-camera
```

When the download is completed, libraries and keys must be installed by running the next four command lines

```
$ sudo apt-key adv --keyserver keys.gnupg.net --recv-key
  F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE || sudo apt-key adv --
  keyserver hkp://keyserver.ubuntu.com:80 --recv-key
  F6E65AC044F831AC80A06380C8B3A55A6F3EFCDE
$ sudo add-apt-repository "deb http://realsense-hw-
  public.s3.amazonaws.com/Debian/apt-repo bionic main" -u
$ sudo apt-get install librealsense2-dkms
$ sudo apt-get install librealsense2-utils
```

When the installation is completed and debugged without any errors, computer should reboot by running sudo reboot in terminal to make all changes effective. Now the Real Sense sensor should be ready. To test it, the following launch file starts the sensor

```
$ roslaunch realsense2_camera rs_camera.launch
```

Camera stream coming from the sensor can be selected by opening it in own *rqt* window. We use the next command line operation to open it

```
$ rosrun rqt_image_view rqt_image_view
```

When choosing the *image_raw* from the drop-down menu, a camera image should come into view as showed in Fig. 2.

Fig. 2. Camera stream from Real Sense

3.2 A2 LiDAR

An additional sensor to accurate TurtleBot2 SLAM (Simultaneous Localization and Mapping) *gmapping* in the experiments is A2 LiDAR [9]. LiDAR improves the environment mapping by scanning the environment 360 degrees besides the other sensors. It is placed on top of a TurtleBot2; see Figure 5 for sensors fittings.

Slamtec A2 drivers are available at http://wiki.ros.org/rplidar and https://github.com/roboticslab-fr/rplidar-turtlebot2. In terminal, command `cd ~/catkin_ws/src` opens the directory into catkin workspace. The following command downloads the package

```
$ git clone https://github.com/roboticslab-fr/rplidar-
  turtlebot2.git
```

After the download is completed, following commands builds the package

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

For ensuring complete build, update is recommended

```
$ cd src/rplidar-turtlebot2
$ git pull
$ catkin_make
```

Next, USB settings should be changed

```
$ cd src/rplidar-turtlebot2/rplidar_ros/scripts
$ chmod +x create_udev_rules.sh
$ ./create_udev_rules.sh
```

After USB configurations, the *setup.bash* source is run by command `source devel/setup.bash`. Now the sensor should be ready. To test it, the following launch file starts the sensor

```
$ roslaunch rplidar_ros view_rplidar.launch
```

The resulted ROS visualization (*rviz*) view should look something like shown in Fig. 3. If the *roslaunch* do not work properly, USB unplug/plug-in helps to wake up the sensor.

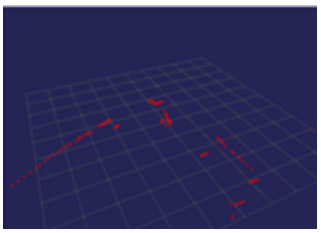


Fig. 3. A2 LiDAR view in rviz

4 TensorFlow Package Installation

We install TensorFlow version 1.14 being compatible to our laptop software/hardware architecture and Ubuntu distribution. GPU (Graphics Processing Unit) requirements must also take into account. TensorFlow uses CUDA (Compute Unified Device Architecture) which means that NVIDIA GPUs are supported. In case of GPU incompatibility, TensorFlow can be used with CPU (Central Processing Unit) support only as used in our installation method.

Preliminaries prior to TensorFlow package installation are that both ROS Melodic and camera (Real Sense) are successfully installed. Note that TensorFlow should be installed before the ROS/TensorFlow recognition package installation.
ROS/TensorFlow package is downloadable at
https://github.com/cagbal/ros_people_object_detection_tensorlow. Then, system requirements should be checked via TensorFlow (Ubuntu) webpage:
https://www.tensorflow.org/install/pip.

At first, we start with pip (Python Package Installer) and python-dev installations

```
$ sudo apt update
$ sudo apt install python-dev python-pip
```

Then, TensorFlow and TensorFlow hub installation

```
$ pip install tensorflow==1.14
$ pip install tensorflow-hub==0.7.0
```

After installations above, it is recommended to debug the code by running it in Python shell

```
$ python
$ import tensorflow as tf
$ import tensorflow_hub as hub
```

If any error messages do not show up in shell, installation had gone successfully.
As we aim to use ROS and TurtleBot2 robot for object detection and face recognition,

TensorFlow API (Application Programming Interface) for ROS is needed. To obtain that, we use repository available at https://github.com/cagbal/ros_people_object_detection_tensorflow. This repository uses a number of open-source projects to work properly:

- [Tensorflow]
- [Tensorflow-Object Detection API]
- [Tensorflow Hub]
- [ROS]
- [Numpy]
- [face_recognition] https://github.com/ageitgey/face_recognition
- [dlib]
- [cob_perception_common] https://github.com/ipa-rmb/cob_perception_common.git
- [protobuf]

For Tracker part:

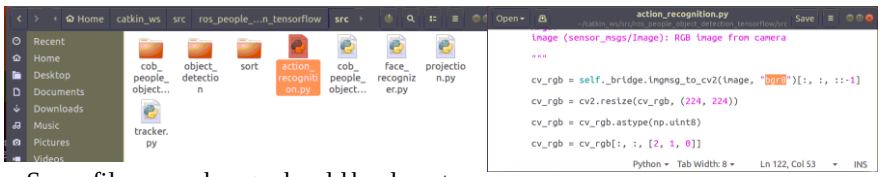
- scikit-learn
- scikit-image
- FilterPy

Firstly, it is important to notice, that the repository [cob_perception_common] compatibility depends on ROS distribution. For ROS Melodic, it is *ipa320* as showed in the following repository cloning. Next, TensorFlow should be installed on the system by running the following commands one by one in terminal.

```
$ cd catkin_ws/src
$ git clone --recursive
  https://github.com/cagbal/ros_people_object_detection_tensorflow.g
  it
$ git clone https://github.com/ipa320/cob_perception_common.git
$ cd ros_people_object_detection_tensorflow/src
$ protoc object_detection/protos/*.proto --python_out=.
$ cd ~/catkin_ws
$ rosdep install --from-path src/ -y -i
$ catkin_make
$ pip install face_recognition
```

ROS TensorFlow API package is now successfully downloaded and installed. The next phase is to change directories so that they match to destination system directory structure. Python files (*.py*) in directory path `~/catkin_ws/src/ros_people_object_detection_tensorflow/src` should be changed as follows

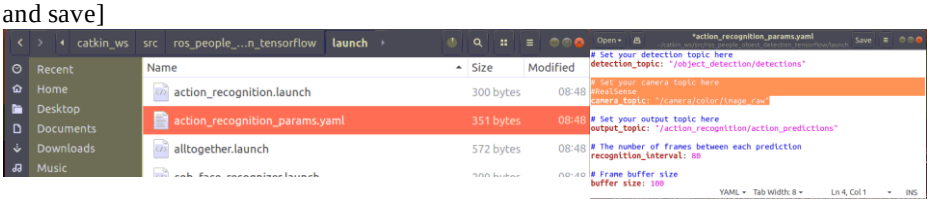
`action_recognition.py` [replace ”passthrough” to “bgr8” and save]



Same filename change should be done to `cob_people_object_detection_tensorflow.py`, `face_recognizer.py` and `projection.py`. Then *.yaml* files in directory path

`~/catkin_ws/src/ros_people_object_detection_tensorflow/launch` should be changed as follows

`action_recognition_params.yaml` [change ”camera_topic” to ”Real Sense” and save]



Same camera_topic change should also be done to `cob_face_recognizer_params.yaml`,`projection_params.yaml` and `cob_people_object_detection_tensorflow_params.yaml`. In the latter *.yaml* file, *rgb* should be used and set the *depth_image* to topics as follows.

5 Experiments

In this section the entire system is experimented and analysed. All the experiments in this article were done on a Lenovo Thinkpad Carbon X1 laptop detailed in Table 1. Both on-board and remote computers are similar laptop computers.

Table 1. On-board computer used in experiments.

Memory	Processor	Graphics	OS Architecture	Hard Disk	Ubuntu Distribution
15,2 GiB	Intel® CoreTM i5-835OU CPU @ 1.70GHz × 8	Intel® UHD Graphics 620 (KBL GT2)	64-bit	503,0 GB	Ubuntu 18.04.5 LTS

5.1 Face Recognition

When applying face recognition, only face images in “people” folder are recognized. Thus, this folder should include all the face images that are going to be recognized. All the images should be in *.png* format and file named according the person as showed in Fig. 4.

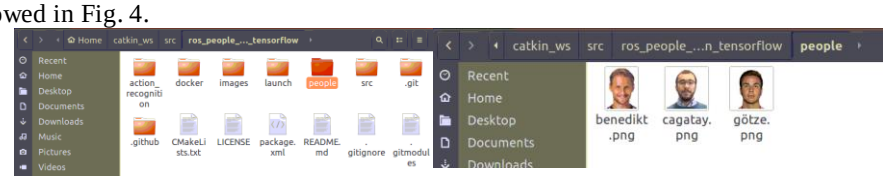


Fig. 4. Face recognition database

Adding new images into this folder database is simple. To make recognition as accurate as possible, white background of the image gives the best results for the recognition. In this way, framing outline between the faces and background are more accurate reducing the delays in recognition.

5.2 Testing the System

In order to test the system functionality, the following preliminaries should be done. In the first, we bring up Real Sense sensor by running the following launch file

```
$ roslaunch realsense2_camera rs_camera.launch
```

By opening the second terminal, we set up the recognition. There are options to start all at once or separately in their own terminals for object detection and face recognition. To launch everything, run the following launch file in terminal

```
$ roslaunch cob_people_object_detection_tensorflow alltogether.launch
```

Launching only object detection

```
$ roslaunch cob_people_object_detection_tensorflow
  cob_people_object_detection_tensorflow.launch
```

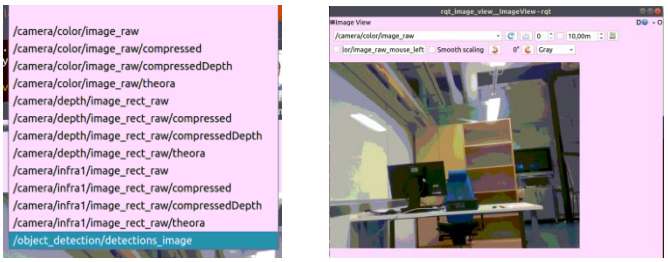
Launching only face recognizer

```
$ roslaunch cob_people_object_detection_tensorflow
  cob_face_recognizer.launch
```

By opening rqt_image_view it is possible to visually monitor and follow all recognition operations

```
$ rosrun rqt_image_view rqt_image_view
```

After running the *rqt_image_view* command, a new window opens showing video stream from a Real Sense camera.



When choosing "/object_detection/detections_image" instead of "/camera/color/image_raw" from the left upper corner menu bar, all recognitions should appear in the view as shown below.

6 Results and Discussion

The aim of this research was to find solutions for issues that can arise of TensorFlow open-source object-/face recognition and sensors software installation for ROS robot platform. To prove these issues, experiments conducted with TurtleBot2 robot was carried out. Practical and detailed solutions for each installation issue were given step

by step. These instructions are kept hands-on type answering for the most crucial case-driven issues when setting up ROS and TensorFlow communication. Available source codes and software packages are utilized, such as several ROS packages from Github public repository, reducing the complexity of further programming.

At its whole, the entire system is proved functional even though getting all hardware working seamlessly with ROS platform remain quite complex. Fig. 5 shows the TurtleBot2 physical structure after sensors are fitted on the robot. Kinect sensor is located on the middle shelf; Real Sense and LiDAR are on the top shelf.

Fig. 5. Robot Platform

In the following Fig. 6 left hand side upper corner is the window of object detection, right hand side upper corner shows face recognition window and on the lower right-hand side is video stream from Kinect. The indoor environment shown in figure is previously mapped by using TurtleBot2's *gmapping* algorithm.

Fig. 7 shows the CPU load when `bringup minimal.launch`, `rviz_launchers view_navigation.launch`, `teleop keyboard_teleop.launch`, `rs_camera.launch` and `tensorflow alltogether.launch` are all launched together.

Fig. 6 Recognition

Percentages 205 % and 189 % in Figure 7 are both TensorFlow programs causing high CPU load. To solve the aforementioned problem, main solutions would be:

- To extend computation capacity of on-board computer (especially RAM disk space)
- To make sure that all images are in applicable image format
- Same outlines and white background color in all images located in people folder

The motivation for using TensorFlow with ROS is the need for object detections and face recognition implemented in ROS-based moving robot. Primary goal is an open-source robotic platform for all users that have an interest to develop the proposed system platform further. In the near future, we continue by combining both Kinect and A2 LiDAR topics into a one *rviz* window producing a better *gmapping* outcome. While still experimental, the entire system environment is fully functional, as demonstrated by the previous experiments.

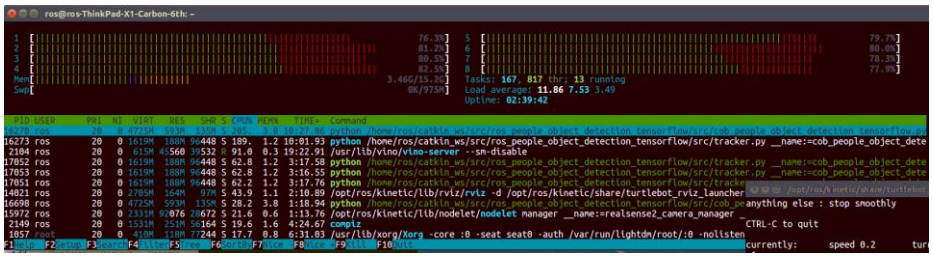


Fig. 7 CPU load

References

1. Kouzehgar, M., Tamilselvam, Y. K., Heredia, M. V., Elara, M. R.: Self-reconfigurable façade-cleaning robot equipped with deep-learning-based crack detection based on convolutional neural networks. Automation in Construction, 108 (2019)
2. Tongloy, T., Chuwongin, S., Jaksukam, K., Chousangsuntorn, C., Boonsang, S.: Asynchro-nous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks. In IEEE 2017 2nd International Conference on Robotics and Automation Engineering (ICRAE), pp. 68-72 (December 2017)
3. Millan-Romera, J. A., Perez-Leon, H., Castillejo-Calle, A., Maza, I., Ollero, A.: ROS-MAGNA, a ROS-based framework for the definition and management of multi-UAS co-op-erative missions. In IEEE 2019 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1477-1486 (June 2019)
4. Reddy, P. P.: Driverless Car: Software Modelling and Design Using Python and Tensorflow, No. 1446, EasyChair (2019)
5. Gaifullin, R., Ivanou, M., Gazizov, R.: Natural Human-Robot Interaction Toolkit. In International Conference on Human Interaction and Emerging Technologies. Springer, Cham., pp. 196-200 (August 2019)
6. ROS People Object Detection & Action Recognition Tensorflow, https://github.com/cagbal/ros_people_object_detection_tensorflow, last accessed 2020/10/30
7. Cervera, E., Del Pobil, A. P.: Roslab: sharing ros code interactively with docker and jupyterlab. IEEE Robotics & Automation Magazine, 26(3), 64-69 (2019)
8. Make Your TurtleBot2 run on ROS Melodic (Ubuntu 18.04), https://github.com/gaunthan/Turtlebot2-On-Melodic, last accessed 2020/10/30
9. Akhmetzyanov, A., Yagfarov, R., Gafurov, S., Ostanin, M., Klimchik, A.: Exploration of Underinvestigated Indoor Environment Based on Mobile Robot and Mixed Reality. In International Conference on Human Interaction and Emerging Technologies. Springer, Cham., pp. 317-322 (August 2019)