CSSR for

Culturally Sensitive Social Robotics for Africa

D4.2.1 Person Detection and Localization

Due date: **10/01/2024**
Submission Date: **10/01/2024**

Start date of project: **01/07/2023** Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **Natasha Mutangana** Revision: **1.0**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Contents

Executive Summary

This report, Deliverable D4.2.1 Person Detection and Localization, details the development process of the person detection software module for the Culturally Sensitive Social Robotics for Africa (CSSR4Africa) project. This module aimed to detect and localize people in the robot's field of view, determine their position in both the image and the Cartesian head frame of reference, and compute the bounding box surrounding the person.

1 Introduction

One of the factors that contribute to effective human-robot interaction is spatial interaction. It is important that when humans and robots interact, the right amount of space is set between them. For this to happen, both agents need to be able to localize where the other agent is. Task 4.2.1: Person Detection and Localization, one of the tasks in the Culturally Sensitive Social Robotics project, aimed to solve this problem. This task was concerned with analyzing sensor data to detect and localize people with respect to the robot. The main objective of this task was to develop a software module that detects and localizes people in the robot's field of view. If a person is detected, their position is determined in an image frame of reference and the robot's Cartesian head frame of reference. Knowledge of the position of the detected person combined with cultural knowledge will enable the robot to pay attention to people and interact with them in a culturally sensitive manner by respecting social and cultural norms for proxemics. For example, when a robot is navigating in an environment with people present, it will maintain a respectful distance from them when approaching them.

This report, Deliverable D4.2.1: Person Detection and Localization, details the development process of a software module for task 4.2.1: Person Detection and Localization. The report will include the requirements of the module, the module specification, the algorithms used to design the module, a user manual, and recommendations for future work.

2 Requirements Definition

Task 4.2.1 was aimed at developing a software module capable of detecting and localizing people within a robot's field of view. This section outlines the specific functional requirements for this module.

1. **Accurate Detection and Localization**
   A fundamental requirement of the person detection module is the ability to detect human presence with high precision. This detection must minimize both false positives and false negatives to ensure reliability. Once a person is detected, their position should be accurately determined in both an image frame of reference and the robot's Cartesian head frame of reference. In scenarios where multiple individuals are present, the module should have the capability to detect and localize each person individually. For each detected individual, the module will compute a bounding box and assign a unique label (e.g., "Person 1"), which will be displayed on the output image.

2. **Coherence in detection and localization over time**
   To ensure coherence in detection and localization over time, each detected person is labeled in the image (e.g., "Person 1"), and the same label is assigned to that person in subsequent images. The label and the bounding box are color-coded, with different colors assigned to different people, and the same color is given to the same person in each subsequent image. If that person is no longer detected in an image, then that label is reused. If that person reappears in a subsequent image, they will be given a new label. A person in one image is considered to be the same one in a previous image if the spatial displacement of the person is less than a given tolerance specified by a configuration parameter value. Each detection is assigned a confidence value between 0 and 1 to indicate the likelihood that the detection is not a false positive.

3 Module Design

For the objectives of Task 4.2.1: Person Detection and Localization to be met, a comprehensive review of different object detection algorithms was made to ensure that an appropriate algorithm is selected to accurately detect and localize people within the robot's field of view. Two algorithms that were used during the development of the person detection module are the Histogram of Oriented Gradients (HOG) and You Only Look Once (YOLO) algorithms. Both algorithms have their advantages and disadvantages.

The HOG algorithm, initially proposed by Dalal and Triggs [1], characterizes features such as object appearance or shape well by the distribution of local intensity gradients in the image [2] The HOG algorithm is known for its ability to capture local shape information in histograms, making it suitable for detecting and localizing people within the robot's field of view. It provides a robust method for feature extraction and has been extensively used for pedestrian detection, demonstrating its effectiveness in capturing discriminative content and reducing image spectrograms. However, it is important to note that while HOG is effective for feature extraction, it may have speed and real-time processing limitations, which are crucial factors to consider for the person detection and localization module.

On the other hand, the YOLO algorithm, introduced by Redmond et al., has gained significant importance in the computer vision community due to its real-time and accurate object detection capabilities [4]. It offers real-time object detection, making it a suitable algorithm for the person detection and localization module, especially in scenarios where real-time processing is essential for spatial interaction between humans and robots. Furthermore, YOLO learns very general representations of objects, indicating its ability to provide comprehensive object detection capabilities [3]. However, to achieve real-time object detection, the module must have access to enough computing resources, such as a Graphics Processing Unit (GPU).

Since both the HOG and YOLO algorithms offer different advantages for person detection and localization tasks, this module development incorporated both in the design process.

4 Module Specification

This section defines the functional characteristics of the person detection module. The module detects the presence of people in the robot's field of view and computes their position in an image frame of reference and the robot's head's Cartesian frame of reference. In addition, the region that the person occupies in the image is determined by computing the bounding box surrounding the person. If more than one person is present in the robot's field of view, all of them are detected and localized. The label and the bounding box are color-coded, with different colors being assigned to different people.

The input to the module is an RGB image obtained by subscribing to one of the robot's cameras' topic or one published by an RGB image driver, and a depth image obtained by subscribing to one of the robot's depth sensors' topic or one published by a depth image driver. The names of the topics to be used for each sensor are read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic's name.

The output is an RGB image and a depth image, with bounding boxes drawn around each detected person in both images and an array of records, one record for each person detected. The components of a record are the person's label, the 2D image coordinates denoting the centroid of the bounding box, the width and height of the bounding box, a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive, and the 3D coordinates that define the point that

corresponds to the centroid of the bounding box surrounding the person in the image. The output RGB image is published on a topic named **"/personDetection/rgb_image"**. The depth image is published on a topic named **"/personDetection/depth_image"**. The array of records is published on a topic named **"/personDetection/data"**. The output can be visualized by running the stub node.

The operation of the module is determined by parameters provided in a configuration file named **"personDetectionConfiguration.ini"** that contains a list of key-value pairs. One key-value pair specifies the platform on which the tests are to be run, i.e., the physical Pepper robot or the Pepper simulator (e.g., platform robot — platform simulator). One key-value pair specifies the RGB camera to be used (e.g., camera FrontCamera — camera StereoCamera). One key-value pair specifies the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). The last key-value pair specifies the filename of the file in which the simulator sensor and actuator topic names are stored (e.g., simulatorTopics simulatorTopics.dat).

5 User Manual

This section provides detailed step-by-step instructions on how to set up and run the person detection software on the Pepper physical robot or using drivers.

5.1 Prerequisites

The person detection module was developed using Ubuntu 18.04 ROS 1 Melodic. The following instructions assume you have the Ubuntu 18.04 operating system and the ROS 1 Melodic distribution setup. If you don't already have them, get and install the Ubuntu 18.04 image from here [5] and follow the installation guide here [6] to install ROS Melodic.

5.2 Create and Configure a ROS Workspace

```
$ mkdir -p $HOME/ros/workspace/src
$ cd $HOME/ros/workspace
$ catkin_make
```

Note: The person detection module requires Python 3 compatibility. To achieve Python 3 compatibility, the first catkin_make command in a clean catkin workspace must be:

```
$ catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3
```

This will configure catkin_make with Python 3 [7]. You may then proceed to use just catkin_make for subsequent builds.

In your current directory, you should now have a 'build' and 'devel' folder. Inside the 'devel' folder, you can see that there are now several setup.*sh files. Sourcing any of these files will overlay this workspace on top of your environment.

```
# Source your new setup.*sh file:
$ source devel/setup.bash

# Add the setup to your .bashrc file so that you don't have to do this
   every time you open a new terminal [8]
$ echo "source $HOME/ros/workspace/devel/setup.bash" >> \
$HOME/.bashrc
```

5.3 Install the Person Detection Software

```
# Move to the source directory of the workspace
$ cd $HOME/ros/workspace/src

# Clone the person detection software from GitHub
$ git clone https://github.com/NMutangana/personDetection.git

# Build the source files
$ cd .. && catkin_make
```

You Only Look Once (YOLO) is one of the state-of-the-art, real-time object detection algorithms used in the implementation of the person detection module. Darknet's implementation of YOLOv3 [9] was modified to fit the person detection task. Follow the instructions detailed below to install the modified implementation of Darknet's YOLO algorithm to meet the requirements of the person detection task:

```
$ cd $HOME/ros/workspace/src

$ git clone https://github.com/NMutangana/cssr4africa_darknet.git

$ cd cssr4africa_darknet

$ git fetch origin

$ git checkout person_detection_modifications

$ make

# Download the pre-trained weight file
$ wget https://pjreddie.com/media/files/yolov3.weights
```

Based on the module specification, the interface of the person detection module will use the physical Pepper robot as the primary driver to generate test data. To get access to data obtained from the physical robot, the development environment for the physical robot needs to be set up. To do that, follow the instructions detailed in Section 3.2 of the Software Installation Manual [8].

5.4 Running the Person Detection Software

The person detection software can be run using the physical pepper robot or drivers to generate test data. Since two algorithms were used in the design of the person detection module, when running the software, it is important to specify which algorithm you want to use.

5.4.1 Running the Person Detection Software on the Physical Pepper Robot

```
# Open a new terminal, and bring up the Pepper robot. This is assuming
   the robot is turned on.

$ roslaunch pepper_dcm_bringup pepper_bringup.launch \
robot_ip:=<robot_ip> roscore_ip:=<roscore_ip> \
network_interface:=<network_interface_name>
```

```
# Open another terminal and launch the NAOqi driver.
$ roslaunch naoqi_driver naoqi_driver.launch nao_ip:=<robot_ip> \
roscore_ip:=<roscore_ip> network_interface:=<network_interface_name>

# Run the person detection software
$ cd $HOME/ros/workspace/src/

# Open a new terminal
# If you want to test the person detection module using the Histogram of
   Oriented Gradients (HOG) algorithm
$ rosrun personDetection personDetectionHog

# If you want to test the person detection module using the You Only Look
    Once (YOLO) algorithm
$ rosrun personDetection personDetectionYolo

# Open a new terminal to visualize the outputs
$ rosrun personDetection personDetectionStub
```

5.4.2 Running the Person Detection Software Using Drivers

```
# Change the directory to the src of the ROS workspace
$ cd $HOME/ros/workspace/src/

# Open a new terminal, and execute the driver that generates RGB image
   test data
$ rosrun personDetection personDetectionRGBImageDriver

# Open a new terminal, and execute the driver that generates depth image
   test data
$ rosrun personDetection personDetectionDepthImageDriver

# Open a new terminal
# If you want to test the person detection module using the Histogram of
   Oriented Gradients (HOG) algorithm
$ rosrun personDetection personDetectionHog

# If you want to test the person detection module using the You Only Look
    Once (YOLO) algorithm
$ rosrun personDetection personDetectionYolo

# Open a new terminal to visualize the outputs
$ rosrun personDetection personDetectionStub
```

6 Conclusion and Recommendations for Future Work

This module was designed to detect the presence of people within the robot's field of view and to compute their positions in both an image frame of reference and the robot's head's Cartesian frame of reference. It uses RGB and depth images acquired either from the physical robot or a driver. The

region that a detected person occupies in the image is determined by computing the bounding box surrounding the person.

The algorithms implemented in this module have shown promising results, but there is still potential for enhancement and further research. From various tests conducted, the Histogram of Oriented Gradients (HOG) algorithm proved to be effective when the person is farther from the camera, with their full body visible under clear lighting conditions. This implementation is useful for detecting if a person is approaching the robot. However, it would not be efficient if the person was interacting with the robot at close range. Future work should focus on improving the HOG algorithm's performance in scenarios involving nearby individuals. On the other hand, the You Only Look Once (YOLO) algorithm outperformed HOG in accurately detecting people under all environmental conditions, whether the person is near, far, or in poorly lit areas. The primary limitation is that running YOLO on a CPU does not provide real-time results. It takes an average of 12 seconds to produce detection outcomes. Therefore, I would recommend that future research explore utilizing more advanced and faster processing resources when running YOLO. This will ensure that the person detection module delivers real-time results, which are crucial for the effective deployment of Culturally Sensitive Social Robots in real-world settings, such as serving as receptionists or guiding laboratory tours.

7 References

[1] Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE. doi:10.1109/cvpr.2005.177

[2] Singh, A., Shukla, V., Tiwari, S., & Biradar, S. (2015). Wavelet-based histogram of oriented gradients feature descriptors for classification of partially occluded objects. International Journal of Intelligent Systems and Applications, 7(3), 54-61. https://doi.org/10.5815/ijisa.2015.03.07

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788). doi:10.1109/cvpr.2016.91

[4] Zhang, X., Yang, W., Tang, X., & Liu, J. (2018). A fast learning method for accurate and robust lane detection using two-stage feature extraction with YOLO v3. Sensors, 18(12), 4308. https://doi.org/10.3390/s18124308

[5] "Ubuntu 18.04.5 LTS (Bionic Beaver)," releases.ubuntu.com. https://releases.ubuntu.com/18.04/

[6] "melodic/Installation/Ubuntu - ROS Wiki," wiki.ros.org.
https://wiki.ros.org/melodic/Installation/Ubuntu

[7] "ROS/Tutorials/InstallingandConfiguringROSEnvironment - ROS Wiki," wiki.ros.org.
https://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

[8] "Culturally Sensitive Social Robotics for Africa D3.3 Software Installation Manual."
https://cssr4africa.github.io/deliverables/CSSR4Africa_Deliverable_D3.3.pdf

[9] J. Redmon, "YOLO: Real-Time Object Detection," Pjreddie.com, 2012.
https://pjreddie.com/darknet/yolo/

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Natasha Mutangana, CMU-Africa
David Vernon, CMU-Africa

Document History

Version 1.0

First draft.
Natasha Mutangana.
10 January 2024.