Robocup@Home Education 2021 Technical Paper

UBC Open Robotics

Contents

1 Introduction

The Robocup@Home Education challenge is comprised of three tasks - Carry my Luggage, Find My Mates, and Receptionist - which our team has broken down into several software challenges which then required integration with the hardware - a Turtlebot2 with a PhantomX Pincher robot arm.

2 Person Tracking

The Person Tracker module uses the Deep SORT algorithm and a modified tiny-YOLOv4 model trained on the COCO dataset (that only contains the

"person" label) to identify any humans in the frame of either a video or a live webcam. Once a person is identified, a bounding box is drawn around them alongside an identifier number; this number is used to distinguish different individuals.

The Person Tracker has a mAP (mean average precision) of 95.553%. In testing, the tracker is able to accurately track an individual for up to 20 minutes without a significant drop in accuracy. The individual is also able to leave the frame for up to 25 seconds before the tracker will no longer be able to match people entering the frame to their previous ids. The tracker runs at an average of 20 FPS on a Ryzen 5 3600 CPU. The primary flaw of the tracker is the effect of low framerates on its performance; under these conditions the tracker struggles to re-detect or even perform basic tracking if the individual is too fast or the algorithm takes too long to process a frame.

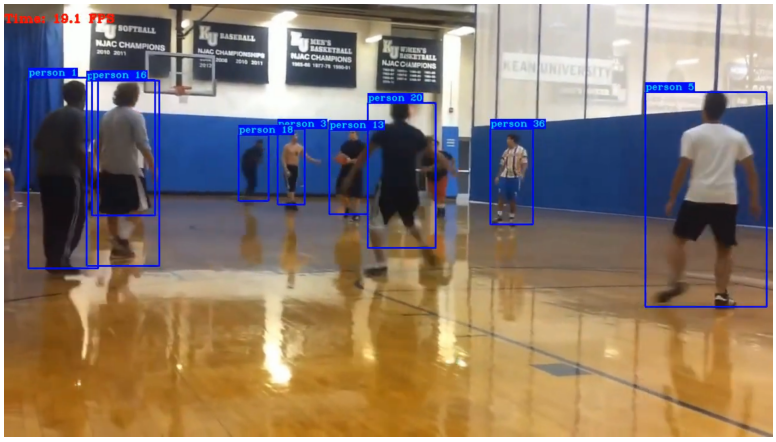


Figure 1: Example of Person Tracking in a Basketball Scene. The model successfully keeps track of each person's id, even if they move out of frame for a small duration.

3 Pose Recognition

The Pose Recognition module combines person tracking with a classification model to perform pose recognition on people in photos, a video, or live webcam footage. The person tracker identifies people in the input and creates a bounding box for them, which is then passed into the model. The classification model uses image classification to distinguish between people standing and sitting. The person tracker uses YOLOv3 and the model was

trained on a filtered version of the COCO dataset containing only images with the "person" label.

(a) Sitting Pose (b) Standing Pose

Figure 2: Pose Recognition on video of person sitting down.

Performance-wise, the classification model is quite accurate. From testing, the only erroneous predictions came from images in which the target's legs were not at all visible in the image. The module runs at a maximum of 8 FPS for live webcam input, and processes approximately 10 images per second.

4 Object Detection

Object detection is an important functionality the pipeline needs to have. We separated the task into the different types of object detection so we could develop them in parallel. These are furniture detection - which allows the system to describe a person's location relative to furniture around them, and bad detection - which is required for the Carry My Luggage task.

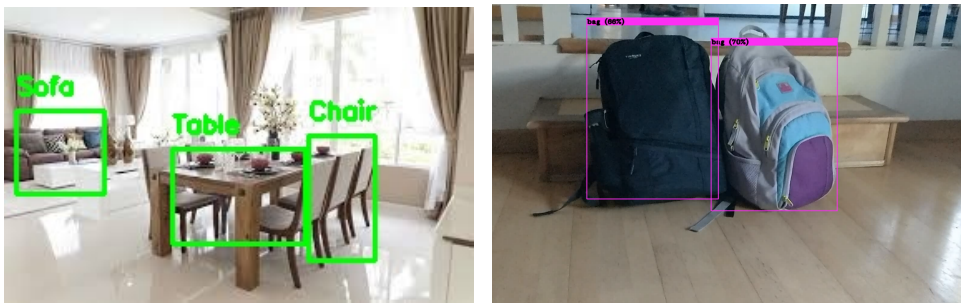
4.1 Furniture Detection

The Furniture Detection module can be used to detect three classes of furniture in an input image, namely, chair, table and sofa. It has been trained using a tiny-YOLOv3 model. The dataset used for training/testing purposes consists of roughly 1500 images (500 per class) with a mix of simple, easy to identify images taken from datasets on Kaggle.com, and slightly more complex images taken from the COCO dataset. Upon selection, the images were manually labeled with bounding boxes around the objects of interest.

The final mean Average Precision (mAP) of the trained model was 82% with a test loss of 0.71. The model accurately detects objects in images at a decent speed, but has trouble detecting objects in complex images (for example, many chairs instead of only one, or a piece of furniture partially obstructed by another object).

4.2 Bag Detection

Similar to furniture detection, bag detection is an object detector which aims to locate bags given a camera image as input. Based on the tiny YOLOv4 architecture, it was trained using the darknet framework on the MS COCO database, which was filtered to include only relevant classes. Training was performed on an RTX3070 GPU for around 7 hours. The best mAP@0.50 was 31%, which seems quite low, however in practice this mAP was satisfactory. The low mAP is suspected to come from the immense variety of images present in the COCO dataset, several of which are edge cases and difficult to detect.



(a) Furniture Detection (b) Bag Detection

Figure 3: Object Detection. a) Furniture detection example. b) Bag detection working despite low mAP during training.

5 Speech Recognition

The Speech Recognition module uses Vosk, an offline open source toolkit, to enable speech recognition. It offers functions to transcribe text from audio captured by a microphone in real-time, and uses a lightweight LibriSpeech model to allow speech-to-text transcription with a low word error rate of 9.85% and latency of 0.15s. For our project, we configure the module to initiate speech recognition upon using desired wake words.

For the purposes of the tasks we are tackling, this module has to extract entities from the transcripts. This requires us to parse and tag the recognized words against a model to precisely identify desired entities. In our case, we trained a language processing model based on NLTK's Stanford NER classifier on a training set corpus for named entities (obtained from Kaggle and modified to our needs). This is used to extract names of people from natural speech responses. We also trained another model with a small set of training examples (approximately 500) about drinks and beverages to allow us to identify any drink mentioned in a natural speech response.

Lastly, to allow our robot to interact with the end users, we use the pyttsx3 library to enable text-to-speech. This library provides us the ability to use offline text-to-speech as well as provides configuration settings to adjust the robots voice, volume and speech rate.

Although this module performs well and achieves worthwhile results, we are still challenged by a few issues. This includes the inability to recognize uncommon names, and occasionally falsely predict drinks due to the use of a crude entity extraction model produced from a very small set of training examples.

6 Age and Gender Prediction

6.1 Age Prediction

The age and gender prediction model is based on the paper "Age and Gender Classification using Convolutional Neural Networks" by Gil Levi and Tal Hassner. The age prediction returns the top 2 predictions among eight age groups, '(0, 2)','(4, 6)','(8, 12)','(15, 20)','(25, 32)','(38, 43)','(48, 53)','(60, 100)'. The output is in the form of 'Guess @ 1 age group with prob of __' and 'Guess @ 2 age group with prob of __'. These details are specific but not enough for our task.

We noticed that the age group labels contain gaps between each one (there is no 13 or 14). Since predicting the subject's age using only integers

can be inaccurate in some cases, we decided to add a string description to accompany the prediction output. We describe someone with the prediction of age (8,12) as a 'teenager', that way if the subject is actually 13 or 15, the term 'teenager' can cover the slightly mislabeled cases, as well as when the subject's age falls within the gaps. Accordingly, we assigned 'toddler, kid, teenager, young adult, adult, middle age, and elderly' to each age group.

In order to rank the subjects' ages, we implemented a new parameter called age_est_2. At first, we planned to rank the age group and then its top prob separately to get the oldest person. It is obviously slow. Later we tried to compare age_est = upper_level_age * prob, which is the first parameter we've come up with. We noticed a problem with this parameter existing in such cases: subject A was labeled (25, 32) with prob = 0.57, B was labeled (15, 20) with prob = 0.94. A should be ranked older than B, but instead the age_est of A = 32 * 0.57=18.23 and age_est of B = 20 *0.94=18.75. Parameter age_est allows subjects with higher prob in lower age groups to be ranked first, and that's not what we want.

Age_est_2 = guess@1 age * prob + guess@2 age * prob. Parameter age_est_2 solved the cases like subjects AB mentioned above. To get the best ranking results, ideally, we can keep adding all the (guess * prob) till the 8th guess, but age_est_2 is already accurate enough for our tasks. In general, the age prediction has slightly lower accuracy in middle-age groups (38, 43) prediction, they tend to get mislabeled into younger age groups. Other age groups' predictions are accurate.

6.2 Gender Prediction

Gender prediction outputs the higher probability of the two (Male, Female). A very important factor in gender prediction is hair length. For female subjects, if there is no hair or dark material placed in the area close to the face and directly below the ear, it can be easily misclassified into males. Situations like this include when females tie up the hair to the back. For males it's the counter effect: if male subjects have longer hair and it's below the ear, it can be misclassified into females. Interestingly, makeup and beards are not among the important factors for classification.

The accuracy of the prediction depends on the quality and the lighting of the input. Both predictions run on image input, using TensorFlow 1.15 and a CNN model. The robot camera does not have a very high resolution, which could bring down the accuracy. We will train the model with low-resolution datasets for improvements.

(a) (b)

Figure 4: Age and Gender Prediction. a) Guess 1 is the most probable age range given, and the gender prediction is labelled with M for Male and F for Female b) Example of how age_est_2 can improve age prediction

7 Arm Navigation

The Arm Navigation module uses a ROS package called MoveIt! to control the arm's navigation and object manipulation properties. Using its powerful capabilities, we were able to program the arm to attain poses such as resting, carrying, dropping off and waving. One component of MoveIt! that we found very useful was its ability to compute inverse kinematics (IK). When given the desired pose for the arm, we need a way to calculate the specific coordinates for each joint (ie. the IK), which becomes exceedingly complicated as the number of joints increases. The MoveIt! package takes care of this by computing the complex, low-level IK calculations, using the Kinematics and Dynamics Library from the Orocos Project. The implementation uses a numerical method for the computation.

The MoveIt! Interface itself offers a simple, step-by-step way to plan and test arm poses. First, we build a description of our robot's arm by defining its joints, gripper and linking components. Then, by specifying certain positions for each joint, we can visually create pre-planned poses for the arm, as seen below in Figure 1. MoveIt!'s backend would then attempt to calculate the IK for us and give us the solution if the pose we had defined was possible given the arm's physical constraints. The joint coordinates for each pose were then stored into a configuration file.

The gripper state was published to the gripper_state topic. To determine

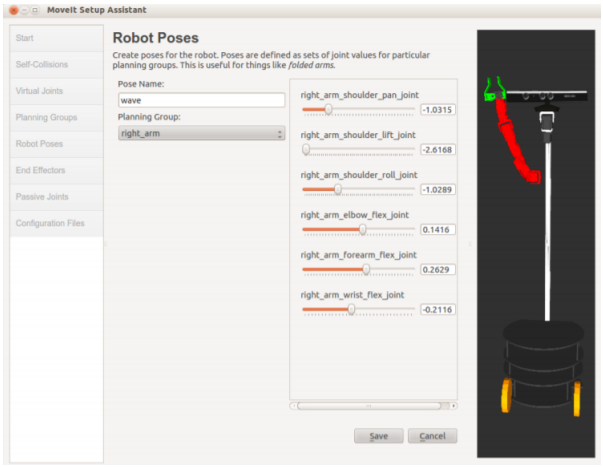


Figure 5: Pre-planned waving pose

which pose to move to, the arm navigation node subscribed to the arm_pose topic.

8 SLAM Navigation and Localization

Our SLAM Navigation and Localization module was done using the turtlebot_navigation stack from ROS. The stack gets sensor information from our robot's astra camera as a sensor, which gets published over sensor_msgs/LaserScan or sensor_msgs/PointCloud messages. For the mapping, odometry information was published using tf and nav_msgs/Odometry message. Here we make use of the geometry/msg Twist to facilitate the movement of the robot in the real world. The base controller has a node that subscribes to the "cmd_vel" node that converts velocity into commands for the motor. A person is followed by repeatedly getting the depth and angle information of the person with respect to the robot and sending repeated movement commands accordingly. Movement is adjusted using a p controller where we consider the robot to be the plant and with a desired distance between person and a desired angle, 0 degrees, we repeatedly check the error and use this error in depth and angle to adjust the movement. Navigating to a desired location in the map is performed using the move_base node. The move_base node links together a global and local planner to accomplish its global navigation task.

(a) PID controller

(b) Mapping using Lidar scan

Figure 6: Navigation. a) Using a PID controller to steer autonomously towards targets b) Using a map constructed from Lidar-like data to go to target locations

9 Integration

The integration of all software modules with the hardware components were facilitated using ROS (Robotic Operating System). ROS's APIs were used to simplify the amount of code needed to be implemented to control the Turtlebot2 which we are using for the Robocup@Home Education competition. For example, as mentioned earlier, the MoveIt! Motion Planning Framework, embedded in ROS could be easily used for robot control, 3d perception, and collision checking. Other tools such as Rviz and Gazebo also makes it easy to visualise and simulate the robot's operation, which is especially useful during the current global pandemic

Our project's integration revolved around incorporating three important design decisions.

Firstly, we created a Turtlebot_wrapper which wraps all nodes having to do with the robot. By creating this we are able to offer higher level functionality such as 'follow_person'. By setting the state of the turtlebot wrapper to 'follow' the wrapper will automatically subscribe to the necessary nodes: for example, PersonTracking mentioned above and the callback will send appropriate geometry_msgs.msgs.Twist messages in accordance with the PID controller we implemented.

Secondly, we incorporated all modules that we developed into either ROS

nodes / services or importable packages. Trained weights are stored according to a standardized file structure, so that these applications can be easily deployed to any system. In an effort to make integration frictionless and modularized, we ensured that any individual module had clear endpoints and well-defined inputs and outputs (commonly in JSON format). For example, we ensured that all object detectors that we developed which output bounding boxes all used the same format for the boxes: x_center, y_center, width, height. In addition, all these dimensions are normalised between 0 and 1 in relation to the size of the image used. So, even if the image needed to be resized for some application, these parameters could still be easily used.

Finally, by realising the first two items, we are then able to write a set of completely decoupled scripts for running through the Robocup@HomeEducation tasks. Subtasks such as text to speech can be imported when necessary and other more frequently repeated subtasks such as person following can be run as a rosnode, which can be polled or subscribed to.

Throughout all this, ROS is used as the platform which handles message queues, timing and interaction with hardware drivers. By interfacing across ros topics we were also able to use modules written using different versions of Python or Python libraries.