OpenCV Person Follower

Hector Fraire Shailen Patel Michael Tirtowidjojo

Abstract

We worked on developing a system to allow a robot to follow a person through an environment using 2D images of the environment. This would allow the robot to move with a person and carry out tasks the robot is designed to do. In this paper we present the person detector for 2D images we implemented for the bwi robots as well as the method for following a person we developed. With this we show the possible uses for this method as well as the shortcomings of this method.

1. Introduction

There are many uses for a robot being able to follow a person in his or her daily tasks. Following a person can be used to have the robot carry items for someone looking to deliver many things throughout a building and can also be used to teach a robot a new path to follow to familiarize the robot with its surroundings. The task of having a robot follow someone proved to be interesting due to the many different possible ways to implement such a functionality. [Gockley, R. et. al] discusses two possible implementations: directly following a person, and following the path that the person took. Additionally, there are options in what kind of sensor to use with the kinect sensor using point clouds or any camera using 2-dimensional images being two of the main ways to detect a person.

2. Related work

Because our work dealt with such a broad concept, there are many projects that can be done that relates to our project. Originally, our plan was to save the paths that the robot takes in order to pull them up in the future so it can move places on demand. This could also prove to be an interesting project that would work hand in hand with ours. Another possibility to implement would be a tour guide robot that gives directions to those in a place like a museum. The robot would use the saved paths to either tell people how to get somewhere, or simply move to that desired location while the tourists follow the robot. There are also robots today that have a feature to listen to voice commands given by humans which the robot can then execute. The LS3 quadruped, made by Boston Dynamics, is a strong example of a project related to ours. Of course, however, it deals with many more advanced concepts than does our project. Movement and following capabilities are important features needed in many robots in the army. A project that uses similar person detecting capabilities as ours is the Human Detection for Robotic Urban Search and Rescue project conducted at CMU. This robot searches for humans using a wide variety of sensors to compensate for the unpredictable urban terrain.

3. Technical approach

In this section we will discuss the person detector we used as well as our implementation of a person follower. The detector works independently from the follower allowing it to be utilized in other projects that do not need the follower while the follower is dependent on the detector

3.1 Person Detection using OpenCV

For our person detector we decided to implement the OpenCV default 2D person detector that uses a histogram of orientated gradients (HOG) with support vector machines (SVM) as a ROS node. Our intention with this implementation of the OpenCV detector was to have a stand-alone node that could be ran independently allowing other researchers to use it if they are in need of a 2-dimensional detector. By designing it to

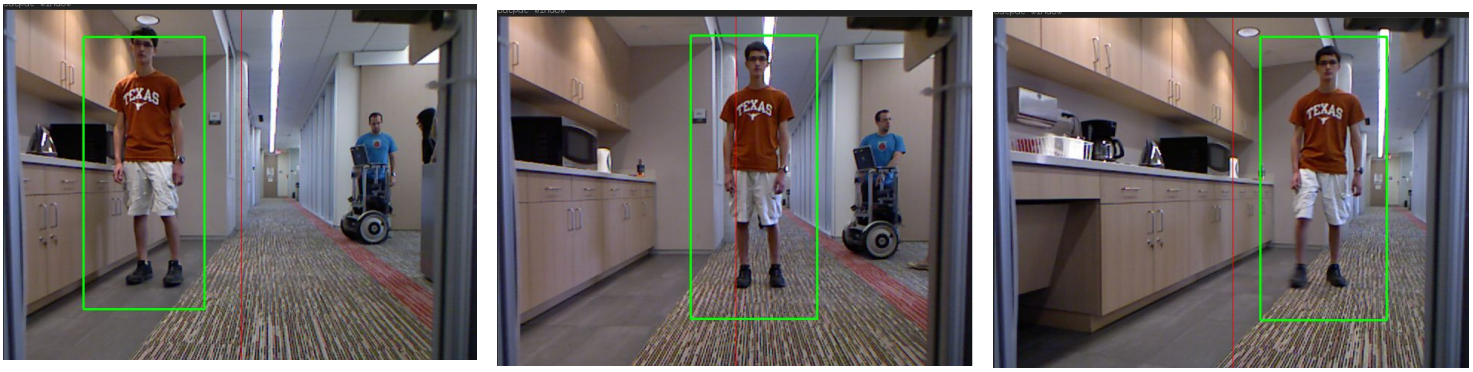

Figure 1: OpenCV person detector with the person to the left, center, and right of the robot. The detector shows a green bounding box around the person and a red line down the center x line of the image.

run independently of our follower, the detector can be used on the bwi robots as an alternative to the PCL person detector that uses the algorithm described by [Murano M. et. al] that is already on the robots therefore providing a 2D detector alternative to the existing 3D detector. Additionally, we created a custom message to be published from the detector. Called a HumanArea, this message contains another custom message, Human, and a geometry_msgs/Point which contains the x and y coordinates for the center of the image. The Human message contains a geometry_msgs/Point which contains the x and y coordinate for the center of the human as well as a float32 for the width of the human and a float32 for the height of the human. The reason for these custom messages will be further discussed in the implementation of the follower, however in summary the custom messages allowed us to more robustly implement the follower if the size of the image were to change.

3.2 Person Follower Implementation

With our follower we chose to assign a priority to the movements performed by the robot. We chose to implement a version of the direct following method proposed by

[Gockley, R. et. al]. Therefore, we determined that having the robot rotate to orientate itself toward its target should have higher priority over the robot moving forward to the target. Thus, our decision block for the robot's movement is modeled by:

```
IF person left of image center THEN
    Turn left
ELSE IF person right of image center THEN
    Turn right
ELSE
    Move forward
ENDIF
```

3.2.1 Turning

We based our turning off of the center of the person and the center of the image. Therefore, it was necessary for us to implement a custom message that provided the center of the image in addition to the center of the person. When the center of the person was farther left that a certain threshold from the center of the image we had the robot determine that it needed to turn left with a matching if statement to determine if it needed to turn right. For determining how much to turn we had two different possible approaches: calculate a more exact amount to turn based on the distance from the center or have the robot turn a small amount and then re-evaluate if it needed to turn more. We decided against calculating a more exact amount to turn as we decided it would be more robust to have the robot continuously update which direction it should turn taking into account the fact that a person may have moved back in the opposite direction the robot was originally turning. We initially implemented the second method having the robot turn a small amount each time. However, this was often slow when the robot had to turn a greater distance. In the end, we implemented a hybrid method for turning where we had the robot only turn a small amount each time while increasing the amount it turned when the distance from the center was greater. While this approach does not calculate a distance to turn as suggested by the first method, it takes into account the idea behind the first method that if the target is farther from the center the robot should turn more.

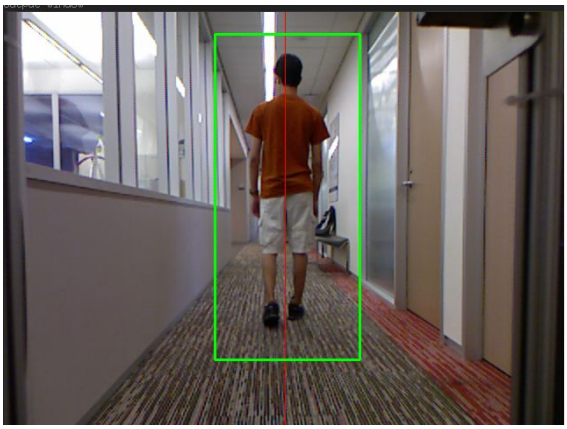

Figure 2: Robot moving forward with its target directly in the center of the image.

3.2.2 Forward Movement

Controlling how much the robot moved forward presented two main options. We could either get depth data from a point cloud using the kinect sensor or we could use a heuristic based on the height of the person being

followed. However, we decided to only use the 2D image instead of iterating through point clouds for reasons further discussed in Section 4.2. Thus, we utilized the height of the person to create a heuristic to determine whether or not the robot need to continue to move forward. This provided a the possibility for two independent methods or a hybrid method similar to the one in Section 3.2.1 used for turning the robot that would allow the robot to move forward more if the height of the person was smaller. As with turning, we implemented the hybrid method having the robot move a greater distance when it is farther away from the target and a shorter distance when it is closer. Moving forward is similar to the turning for the robot in the sense that it moved small amounts each time in the direction of the person rather than planning a goal to the person then executing that goal. The reason for this is the same as the reason for it with turning as we wanted the robot to be able to constantly readjust itself and check if it need to turn more before continuing to move forward rather than planning a goal to a location where the person is likely to not be once the goal has been completed.

Figure 3: Robot first orienting itself toward its target (left), then moving toward its target (right).

3.3 Sound

The final implementation to our person follower is the robot saying the movement it is going to perform. With this, the robot says “left,” “right,” or “forward given for each respective direction and announces “done” when it has reached its target. As our implementation constantly sends commands to turn or move to the robot, it would have been impractical to have the robot announce its movement each time a goal was sent as it would be constantly repeating itself. Therefore, it was necessary to include a flag that kept track of the last movement performed by the robot. With this, the robot only says the movement if the movement changed such as the robot turning left after moving forward. With the sounds, the robot is able to communicate to the person it is following what it intends to do allowing the person to keep track of the robot to ensure it is still tracking him or her correctly without needing visual contact with the robot.

4. Experiments / Evaluation

The experiments consisted of isolating functionalities of the robot, e.g. rotating and moving forward, and testing them separately. After several trials of testing the two functions separately, we tested both running concurrently. Once our group successfully

implemented the functionalities of detecting and following a person, we began testing it by having a person stand in front of the detector and running the person-follower node.

4.1 Rotation Experiments

To test the rotation and angling of the robot, we removed the “moving forward” functionality from the code to isolate and perfect the rotation. One person would stand at a slight angle to the front of the robot at the closest distance to which the kinect sensor could detect a person, which is roughly 1-2 meters. We would have trials to observe if the robot would actually rotate to face the person. If it did face the person and stop, then the person would take incremental steps sideways to observe the robot’s rotation angle and rotation speed in following the person’s new position. The main issue we faced with the turning was determining what threshold from the center of the image to turn should be and how much the robot should turn. Initially we used half of the width of the person as the threshold and only turned the robot when the person was outside of the center of the image +/- half of the width of the person. This ended up being too wide of a threshold for the robot to be considered facing the person so we reduced the threshold to a static amount of 50 pixels. This allowed the robot to more accurately determine when it needed to turn. Once that was fixed, the next issue our group faced was how much the robot should be turning. The problem we faced was if the robot only needed to turn a small amount, with the fixed amount to turn the robot would over-rotate to where the follower would decide it needed to turn back in the other direction. By using the hybrid method presented in Section 3.2.1, we were able to have the follower chose smaller amounts to turn preventing the robot from over-rotating. Once this issue was resolved, a person from our group could stand at an angle to the kinect sensor, and the robot would turn itself to perfectly face in the direction of the person it detected with no issue. Then the person could step sideways in short or long steps, and as long as the person remained in frame of the kinect sensor, the robot would follow and face the person with the appropriate rotation speed and angle, and it stopped once it faced the person.

4.2 Forward Motion Experiments

Once we perfected the rotation of the robot, we started experimenting with the forward motion of the robot. Initially, we removed the rotation functionality from the code in order to only focus on the forward motion. Our first idea was to use the depth of the person begin detected in order to calculate how far the robot should move forward. For example, if the depth (distance of the person from the kinect sensor) was 3 meters, then the robot would move forward ~2.5 meters. However, this proved to be challenging in that we had to implement code that iterates through the point cloud and retrieves the depth of the person. To avoid having to iterate through the point cloud, we decided that working in 2D was a better idea, and we came up with the idea to use a height heuristic instead as detailed in Section 3.2.2. The experiments consisted of standing directly in front of the robot (with no angle) at a distance of roughly 1 to 2 meters and observing if the robot would move forward. The experiments did not show too many problems with the forward motion, apart from having to tweak the code to change the distance from the person in which the robot stopped and the speed of the robot when moving forward.

The overall motion worked, and these tweaks were just semantics. After the forward motion worked for a stationary person, we experimented with a moving person. A person would stand at a starting position of roughly 1 to 2 meters from the robot and then would start stepping away slowly. Our group noted that, even at top speed, the robot only has the capability of following a slowly-moving person. The person can not walk away at a natural speed for the risk of the kinect sensor losing visibility. That's just a limitation of the person-detector, the kinect sensor, and the top speed of the physical robot while having the robot perform its usual obstacle avoidance. Given these limitations, the experiments show that the robot does task our group set out to achieve: following a person.

4.3 Experimenting With The Finished Product

Once all the code was written, we experimented with the robot with all functionalities enabled, which included rotation, forward motion, and sound. For the first few experiments, a person would stand at an angle to the robot, about 1 to 2 meters away, and we would observe what the robot did. Since we had already perfected the rotation and forward motion separately, putting them together went smoothly. One problem we observed during these experiments is that the robot would sometimes move forward before angling itself to the correct position to face the person. We prioritized rotation before forward motion in the code and fixed that bug. Apart from that, the robot did everything as expected. It would rotate to face the detected person if needed and move toward the person if far enough away. We also enabled the sound component during these trials of testing the motion, and the robot seemed to say "left", "right", and "forward" in the correct times when it was moving left, right, or forward. There were some compile errors in our code initially when implementing sound, but once we successfully compiled it, there were no bugs in it and it worked perfectly. Overall, the robot's capabilities were tested extensively, as single components and when working together as a whole, so it can be conclusively said that the robot successfully follows a person.

5. Conclusion / Future work

In the end the robot was able to successfully detect a person and orientate itself and move toward the detected person. The project had two main limitations that could serve as the focus of future projects: the classification scheme of the detector and the speed of the robot. With the OpenCV detector, an issue that presented itself was the detector would not always detect the person depending on the person's surrounds and how the person was standing. For general standing and walking motions, the detector was capable of accurately detecting a person. However, if the person were to hold their arms in certain ways or stand too close to the sensor the detector would not be able to detect the person. The other limitation faced was the speed that the robot followed the person. Since we wanted the robot to perform its normal obstacle avoidance, the robot would only move at a speed that was slower than a normal walking speed for a person. By directly controlling the robot's speed rather than using the existing node to move the robot, we could have the robot follow at a more acceptable speed. However, this would remove the obstacle avoidance for the robot.

6. References

R. Gockley, J. Forlizzi and R. Simmons, "Natural person-following behavior for social robots," *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, Arlington, VA, 2007, pp. 17-24.