Semi-Robust Detection and Following of Individuals

Michael Brenan, Saket Sadani, Victoria Zhou

Abstract

Currently, the robots involved in the Building Wide Intelligence (BWI) program do not have the ability to follow a given individual. To rectify this tragic fact, we introduce and describe our implementation of a PCL-based person detection, tracking, and following program known as *Creeper*. The system is capable of continuously detecting and tracking individuals in point clouds (provided by depth-enabled cameras), and following specific targets despite some unreliable readings and brief interruptions from other individuals.

Background & Introduction

The primary goal of this project was to implement a program on the BWI segbot machines which could detect and follow individuals using the available sensors on the machine. Of course, the fundamental act of attempting to "follow" an individual is nontrivial, and can be split into three distinct aspects: *detection*, *tracking*, and *movement*.

Person Detection, which is simply the process of finding people/ individuals in a frame of sensory data, is a classic field of research and one of the oldest problems in computer vision. The numerous solutions, nearly all based on machine learning methods such as *Support Vector Machines* ("SVM") and *Neural Networks*, differ largely on the features and dimensionality of data used for detection. While relatively robust solutions have existed in the two-dimensional domain for many years (such as component-based detection in [1]), many new interesting point cloud approaches have arose to take advantage of the abundance of information provided by depth-enabled cameras, such as Point Cloud Library's *people detector* in [2].

Person Tracking, a natural extension of the detection problem, involves *tracking* individuals across sensory input frames (in essence, by attributing each detected person in each frame an identity which may be shared across frames). As the problem arose at nearly the same time as detection, the solution space is also rich; at a basic level, however, most solutions operate on *similarity metrics*, such as distance between detections, color content of detections, or other identifying features in order to track individuals or objects over time, examples of which are seen in [3].

Movement is simply the orienting and movement of the robot towards the tracked individual (though the act of doing that can certainly be anything but simple).

Due to the smaller scope of our report and project, we specifically focus on the aforementioned problems in the three-dimensional domain using the Point Cloud Library (or "PCL"). To this end, we develop and explain the *Creeper* program, which is responsible for

collecting point cloud data from the Microsoft Kinect sensor [4], performing person detection and tracking using operations from PCL, and sending movement goals to the BWI robot's movement goal system (which also handles low-level movement and obstacle avoidance). As in many robotic projects, our system is based on the Robot Operating System (ROS), and our discussion presupposes knowledge of basic ROS topics such as topics and nodes.

Methodology & Implementation

Much like the problem itself, our implementation is split into three distinct and separate phases: detection, tracking, and movement, where information from the previous phase is fed into the next phase. We explain each in detail:

Person Detection - PCL "people" classifier

Due to the limited time constraints of the project, we make use of *background_person_detector*, a lightweight ROS node from the BWI Experimental package which thinly wraps PCL's *people* detector/classifier. PCL's detector operates by attempting to find a "floor plane", and then detecting distinct point clouds (or "objects") on top of the floor plane; each object is passed to a SVM (Support Vector Machine) which has been trained on thousands of samples of standing people, and which marks "matching" objects as people [5]. The background person detector node simply repeatedly calls PCL's detector on point cloud data provided by a Microsoft Kinect mounted on the robot each frame, and then publishes both poses and point clouds for the detected humans at a frequency of about 2 Hz.

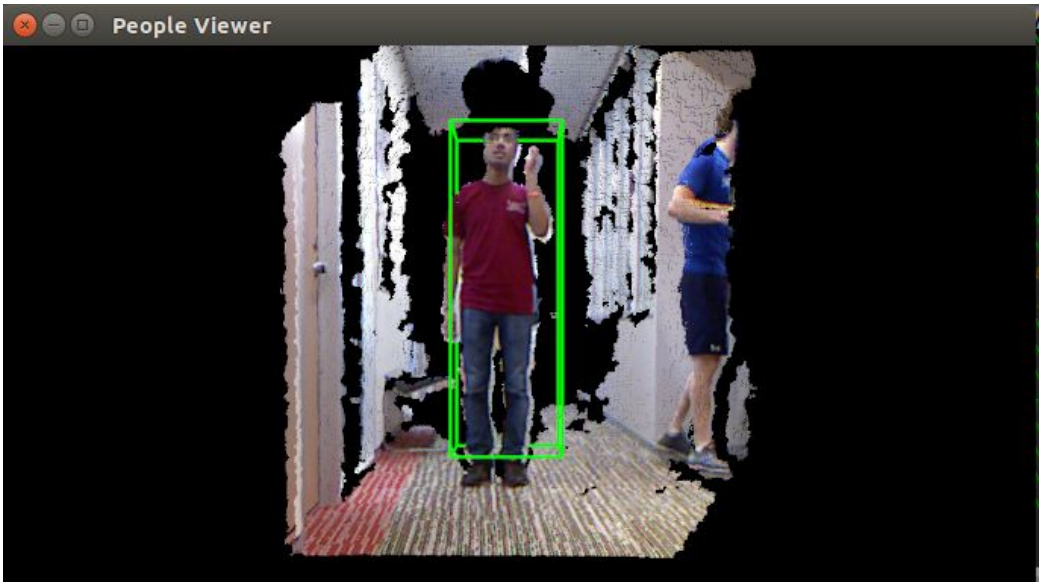

Figure 1: Example of *background_person _detector* detecting an individual. A green bounding box is drawn around person.

Person Tracking - Trivial Classification & Color Classification

Person Tracking is implemented on top of person detection, and takes advantage of the point clouds associated with each person. The two solutions provided with our implementation are the *Trivial* classifier and the *Color* classifier.

- The *Trivial* classifier is, as the name implies, the minimum viable classifier, and simply identifies every input human cloud as a single person. This classifier works quite well when only a single person is present to be followed, but can quickly become nondeterministic when multiple people enter the scene.
- The *Color* classifier is a slightly more sophisticated classifier which uses the average color of the input human clouds to differentiate between people, giving a decent heuristic for differentiating between people with reasonably different clothing, hair, or skin color. The previous average color of each person cloud is stored, and then the best match is determined by comparing the average color of the current input cloud to all of the average colors of the people being tracked. The closest match (by Euclidian distance) is chosen, as long as the difference is within a specified threshold; if the distance is not within the threshold, the cloud is assumed to be a new person and a new classifier and person with that color is created. Of course, the classifier is very sensitive to background noise and to false positives in detection.

Movement - Rotation & Following

The third aspect (and the one people actually see), is to have the robot actually act to follow the individuals which the program is tracking. For our purposes, the actual person to follow depends on the classifier being used - it is simply the first person detected in the frame if using the Trivial classifier, and it is the first person originally detected when the program started if using the Color classifier.

Movement operates in a tight loop, and repeats the same set of checks each iteration:

- Check if anyone has been detected in the first place. If not, then rotate to the left (in an attempt to find someone), and return to the beginning of the loop.
- Check if any *new* information (eg, information which the robot hasn't acted on) is available for the person being followed. If no information has been available for a short time, the detector has likely lost track of the person, so wait a brief time and then begin rotating to search for the person.
- Assuming we have new information for the robot to act on, determine the person's relative location from the frame of reference of the camera in terms of distance and angle (using some relatively straightforward trigonometry), and then send a move base goal to the robot to carry out.

Evaluations

Fortunately, we achieved the basic goal of the project: the robot can follow an individual relatively seamlessly, even if the individual temporarily goes out of sight or goes around sharp corners. Testing done on the 3rd floor of the GDC where the robot followed one individual (without other individuals showing up) all around were generally quite successful; however, the color tracking had serious problems with data noisiness which made the proposed multi-individual tracking infeasible. And, as in any AI project, there are a list of limitations and shortcomings to be address in possible future iterations:

Limitations

There are several limitations of the presented solution, regarding PCL perception, robot obstacle avoidance, and robot movement:

PCL perception

PCL perception is computationally expensive and therefore very slow (i.e. frames are updated every 2HZ), which can cause a notable lag in response time when the robot is first trying to track an individual (though fortunately, the effect is not nearly as prevalent once the robot starts moving towards the person directly).

In addition, PCL's person detector is not completely robust. If the individual is too far away, it only detects a person at certain orientations. For example, if the individual is standing far enough and if any significant portion of the individual is black (i.e. hair, jacket, pants, etc.), the scanner will fail to detect the black portions, and by extension, the person. Far more interestingly, *background_person_detector* will consider whiteboards to be a person as well.

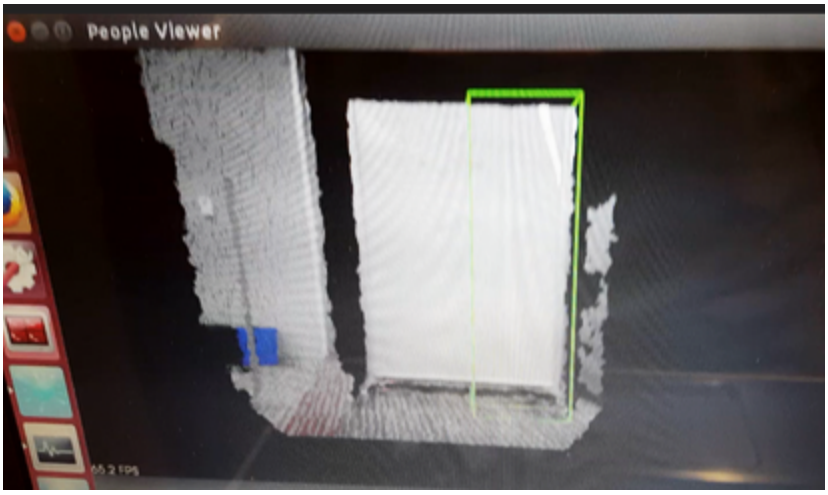

Figure 2: The green bounding box, indicates that a person has been found; however, in this case, a whiteboard has been determined to be a "person".

Robot obstacle avoidance

The robot's obstacle avoidance protocol holds precedence over our program, making movements often times jerky and causing occasional random rotations as the robot scans its surroundings. In addition, when robot gets too close to a person, it will turn slightly as to "avoid" the person altogether, resulting in random swerves at close range.

Robot Movement

The robot's (default) movement is quite slow so the person being followed can not walk too fast or else *Creeper* will no longer be able to detect the individual. However, this can potentially be changed by changing the default speed when sending goals to the robot's */base_link*.

Future Work

Better differentiation between multiple people

Instead of relying on dominant point cloud color to classify people, we could instead actually track the movement of the figures from frame to frame. This would not rely on any feature (eg, color) of a given individual, but just on the assumption that each individual will stay in relatively the same position from frame to frame (i.e. not jump across the room). Tracking movement from frame to frame could be done through the combined techniques of using the Harris detector[6] to find interest points (which, in addition, should not rely on having entire person in frame) and using SURF (Speed Up Robust Features) to find correspondence between the descriptors found using Harris detector across several frames. [7]

Use less computationally expensive method of finding people (using openCV in a useful way)

OpenCV already has a library with pre-trained HOG and SVM detector (pyimage). Combining this tool with techniques, such as triangle similarity, to calculate distance from a given 2D image, it should be possible to use openCV to track and follow individuals. In addition, openCV is less computationally expensive as it only has to process two-dimensional data in comparison to PCL's three-dimensional data, making it a potential candidate for a revamped detector. Even though openCV does not give depth data, as it is only processing 2D images, there are techniques to track object movements [8] and determine object's distance from the camera using triangle similarity [9], which we hope to look into.

Conclusion

While certainly not perfect, we accomplished the core goals of our original proposal, and successfully implemented a program for detecting, tracking, and following individuals using 3D point cloud data as well as PCL utilities. Wrestling with ROS documentation, configuration, and setup was certainly challenging (and was probably harder than the actual algorithms we used!), but the end result was quite nice to see and certainly worth the effort.

Acknowledgements

Thanks Jivko for general advice and Shih-Yun for guiding us through the initial stages of our project (telling us OpenCV wouldn't work as well without significant extra effort).
Also, thank you to the mentors for opening up the lab to us and patiently answering our questions.

References

[1] Mohan, A., Papageorgiou, C., & Poggio, T. (2001). Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *23*(4), 349-361.

[2] Rusu, Radu Bogdan, and Steve Cousins. "3d is here: Point cloud library (pcl)." Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011

[3] Comaniciu, D., Ramesh, V., & Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on (Vol. 2, pp. 142-149). IEEE.

[4] Han, J., Shao, L., Xu, D., & Shotton, J. (2013). Enhanced computer vision with microsoft kinect sensor: A review. *Cybernetics, IEEE Transactions on*,*43*(5), 1318-1334.

[5] Aldoma, A., Marton, Z. C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., ... & Vincze, M. (2012). Point cloud library. IEEE Robotics & Automation Magazine, 1070(9932/12).

[6] Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In Alvey vision conference (Vol. 15, p. 50).

[7] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In Computer vision–ECCV 2006 (pp. 404-417). Springer Berlin Heidelberg.

[8] Rosebrock, A. (2015). OpenCV Track Object Movement - PyImageSearch. Retrieved May 09, 2016, from http://www.pyimagesearch.com/2015/09/21/opencv-track-object-movement/

[9] Rosebrock, A. (2015). Find distance from camera to object using Python and OpenCV. Retrieved May 09, 2016, from http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/