Using Hand Gesture To Command BWI Segbots

Jamin Goo, Mehrdad Darraji
CS 309: Autonomous Intelligence Robotics (FRI I)
May 15, 2017

Abstract

The goal of the project was to allow the hand gesture to command Building Wide Intelligence (BWI) Segbots, specifically the human following command. The first phase of the project was to find out the existing open-source project on real time hand tracking and transferring it into a ROS node. The second phase of the project was to understand the previous attempts on hand following command and utilize them. Lastly, we worked on bridging the first two phases through hand gesture recognition based on a given signal. With this approach, we demonstrated that it is possible to command the Segbots with a hand gesture. Although the main goal of the project was achieved, there are still many limitations in this projects with possible future works. Various limitations and possibilities are discussed at the end of the paper.

1. Introduction

BWI project is an initiative to develop fully autonomous mobile robots that are useful platforms for Artificial Intelligence research and helpful for humans. Nonetheless, one of the imminent issue with the Segbots is the lack of convenient form of communication between the robot and humans. Currently, the operation of the Segbots is mostly depended on the human manual input through linux terminal commands, which makes the operation inaccessible by nontechnical people.

Furthermore, there has been ongoing attempts on improving the functionality of the robots; in fact, such improvements include: human following, voice commanding, etc. Yet, some of the built packages lack accessibility due to the communication issue as previously mentioned. To rectify the problem, we decided to implement a form of human body gesture. Among the possible human gestures, including facial, hand, leg, etc., we decided to pursue hand gesture for it is universal and would result in higher detection rate given the resolution and the posture of the Kinect sensor on the Segbots.

Lastly, for the purpose of this project, we needed an action node which can be called through the hand gesture. Thus, we decided to use the human following node, which is built by the former Freshman Research Initiative (FRI) students. To test the goal of the project, which is to implement the hand gesture recognition and call the appropriate action, we chose the human following node as our successful action goal. This attempt can help overcome the contemporary issues of the Segbots and achieve their main goal of helping people as well.

2. Background / Related Work

Hand gesture recognition is a widely researched topic among computer scientists. In the study examined by Xia Liu and Kikuo Fujimura [1], the authors use depth image to effectively

track the hand and classify the gestures. The use of depth image makes the hand image filtering in the real time easier because we only need to consider the images closest to the camera. This implementation is based on the assumption that our hands are always located in front of our body whenever we make a hand gesture. Along with the hand tracking, Liu and Fujimura focused on analyzing the hand shape and trajectory. On their algorithm, they break down a gesture into four parameters, which are shape, location, trajectory, and orientation. With these detailed classification, they achieve accurate and robust hand gesture recognition for both static and dynamic gestures with the error rate of less than five percents.

Apart from hand gesture recognition, there has been developments on human body recognition system since the advent of affordable 3D camera such as Kinect. To track down the body gesture, developers use OpenNI library which allows skeleton tracing. Currently, under ROS Wiki, there is a package called "rtcus_kinect_gestures", which allow users to create user-custom gestures [2]. While skeleton tracing approach provides easier way of implementing dynamic gestures of hand gesture, it is not widely accepted as a good technique for hand gesture recognition. According to Kurakin, skeleton tracing lacks the ability to track down hand specific gestures such as palm or finger motions [3]. He further points out in his paper that without including these motions, the system is not complete.

3. Technical Approach

The key idea of our gesture recognition system is that there are two people in a given space; one who issues a command, and the other who receives the published command. Also, this approach mainly focus on the static gestures. Our algorithm has the following components:

I. Hand detection and tracking
II. Hand gesture recognition
III. Segbot command

3.1 Hand Detection and Tracking

The hand tracking implementation is based on the Simen Andresen, Vegar Osthus, and Martin Stokkeland's open-source project on computer vision [4]. The implementation can be divided into two main part of hand detection and tracking. The detection algorithm is processed by providing the regions of interest to the program through the color profiling of the user's palm using the OpenCV library. It is crucial to color profile each user's palm since the palm color is not universal for all human beings. Thus, this approach provides higher detection rate compared to hard coding a color value of one hand. After the color profiling, it thresholds the hand from the background. The detection uses seven median points on the hand as binary image and sums them up to get a more accurate reading of the hand. A median blur filter is used on the image afterwards to smooth and remove any noise.

The next step is to create the convex points around the hand which will make the place of the hand more clear so that it can get the farthest points (i.e. the place of the fingers in relation to to the area). A flaw that comes with this approach is that there will be some points that are not the fingertips; however, the authors have a technique of only keeping the relevant points. A point will be dismissed if two conditions are met: the length between the current vertex and the

neighboring vertex is more than four tenth of the length of the whole hand, or the angle between the vertex and its two neighboring vertices is more than eighty degrees. Through the analyzation of the data, the authors were able to keep track of the fingertip positions, number of fingers, number of hands, and the area of the hand.

3.2 Hand Gesture Recognition

Andresen et al's approach through OpenCV was an optimal solution for our problem by taking into consideration the flaws of computer vision and trying to fix them. At this point, the problem of hand detection was solved, and we needed a way to make the Segbots act on the given command. The lighting in the research lab made it difficult to work with OpenCV and as a result an accurate hand detection was difficult at times; a fix that seemed to repair the issue was to use some type of validation to let the algorithm know when to detect a command (explained more deeply in next section). An open hand seemed optimal to let the robot know that it should follow the person in the front of it; the algorithm would detect five fingers and publish a message to the following node to let it know it can start its process of following. The second command is to let the robot know that it can stop following you and that this is as far as it should go; this command is given by two fingers. Now that there is good communication with the Segbots on what it should do, we have to show it what each command means specifically.

3.3 Validation:

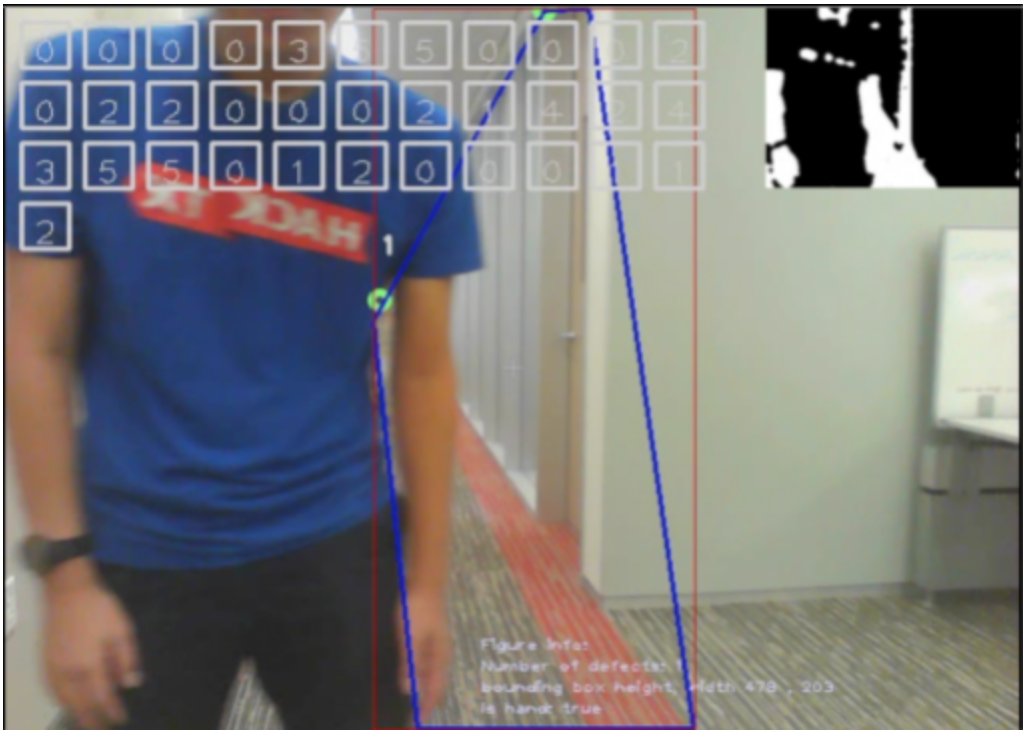

Figure 1. Wrong Hand Detection by Segbot While Moving

When the action node is called through the hand gesture, the segbot started following the person with the ongoing real time hand tracking. It was necessary for the program to keep the

hand tracking node as there is no promise on when the following node would be finished unless the person whom the robot follows makes the stop command. However, the tracking node detected the wrong hand or fingers, causing the current action to be cancelled sometimes without the consent of the user as seen from the figure above. Although we could have improved the image filtering, based on the lighting of environments, wrong hand detection would inevitably occur. Therefore, we decided to implement some form of validation process so that the hand gesture recognition node can be activated only after the successful validation.

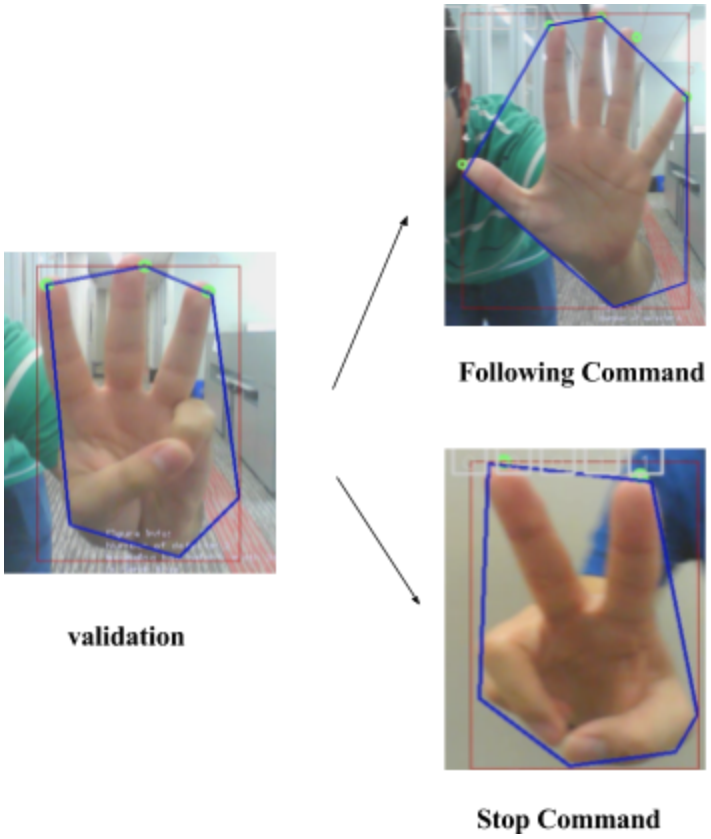

Figure 2. Overview of validation and hand gesture sequence

Our validation implementation is done by designating opening of three fingers as the start of validation process. Additionally, we set that any command has to be read within three time frames since the start of validation. Otherwise, the program would allow the publish of action node for an indefinite time. Thus, by limiting the validation process within three time frames, we were able to increase the robustness of the program as there were significantly less error where the Segbot abruptly stops while following due to wrong hand detection.

3.4 Commanding Segbots

The students from previous year have researched detecting and following people in front of the Segbot [5]. Their research focuses on the use of Point Cloud Library [6] by detecting point clouds or objects on the floor level. By color comparisons, it remembers the person that it has to follow based on the Kinect's frame of reference, and the angle/distance from the person. We

modified their nodes to subscribe to the hand gesture nodes, and it acts on a following command based on the messages that it receives. The node makes no action until it receives a command of five fingers, which at that time, it will do calculations from the point cloud and moves towards the person. It will keep on following the person until a stop message, two finger command, is received - at this point, no messages will be received, which makes the indication to the node that no new movements should be made till further instructions.

4. **Experiment and Evaluation**

The program's success was tested by simply making the hand gesture commands and checking whether the program responded correctly based on figure 2. To test that the program works seamlessly, we divided the test into smaller test cases. First, we tested whether appropriate signal is published based on the hand gesture input onto the screen. Second, we checked whether published signal is received by the subscriber. Until this point, we used GDB debugger to prevent any form of movement to only check the flow of signal. After successful checking of signaling, we eliminated the breakpoint and ran the program entirely. Fortunately, we were able to accomplish our main goal of making the robot understand the hand gesture and behave correspondingly. Yet, we conducted all of our trials on the corridor of Artificial Intelligence lab; there is a need to test the program in other surroundings to test whether the surrounding drastically affects the functionality. Regardless of the achievement, there are several limitations that needs to be addressed and how it should be re-worked in future iterations.

5. Limitation and Future Work

Most noticeably, there was high inconsistency in hand tracking. When the robot was at its stationary point, it had generally no problem understanding the gestures as long as we provide the color profiling of our palms. On the other hand, the program started providing inaccurate and inconsistent output as the robot started following a person. Such fluctuation resulted from the way that the hand tracking is designed. Through the color profiling, we set the program's regions of interest for every frame of image. Thus, when the robot is moving, it will still try to grab similar images that has similar HSV values with our palms. Therefore, even when we are not making any gestures, the program sometimes received wrong input. This problem was partially solved by implementing validation process, but not entirely. To rectify the problem, we would like to implement depth image approach for the hand image filtering. Rather than color profiling our palms, we could use kinect's depth image by grabbing the closest images from our body. This approach can help us achieve consistency through the received data.

Another issue that our project encompasses currently is the lack of hand gestures that the program can understand. As mentioned previously, the program can only understand two commands, one for following a person, and one for stop following. Hence, the program does not face much confusion in identifying and matching the input gesture. Therefore, there is a need to implement more gestures into the system while testing how well the program can understand these gestures, and recognizing some ways to make the robot effectively identify them. We could improve this by allowing dynamic gestures [7]. At the current stage, the program can only understand static gestures, which limits in the variations of gestures. Thus, such implementation

can allow a number of gestures, which can provide better form of communication between robot and people.

Lastly, the current implementation requires the presence of two people in order for the program to be functional. One for calling and making the commands, and the other for receiving the published commands. This way of approach was designed so that we can lower our reliance on machine for the development and debugging of the project. In this way, we were able to work anytime using the webcam from our personal laptop. While it allowed us to quickly reach the goal that we pursued, we would like to improve upon this by allowing only person to be present, meaning that a person can call and receive the action himself or herself. To address this limitation, we could subscribe the RGB data from the Kinect rather than RGB data from the webcam, which was used as our primary source of image. This would allow easier and more free form of communication between human and robot.

6. Conclusion

Although there are several limitations in our approach, we accomplished the core goal of original proposal by implementing a program to command BWI segbots using hand gestures. We felt the lack of completeness and the functionality of the program was due to unsuccessful early attempts on utilizing existing ROS packages and skeleton tracking using OpenNI. This led us to discover non-ROS project and we took the initiative of changing the entire computer vision project into a ROS package. Contrary to our initial perception of its difficulty, wrestling with ROS setup and configuration was definitely challenging, but the end result was rewarding and worth our long effort. In near future, we would like to implement some of the addressed future works to further improve the functionality.

Link to demo:
https://docs.google.com/presentation/d/13vorrt0HmuwcC8WUExaTIUqJwaOqkK6BXudd1eQgbNM/edit

Link to github repository: https://github.com/JGOOSH/Hand_Gesture_Command

7. References

[1] Liu, Xia, and Kikuo Fujimura. “Hand Gesture Recognition using Depth Data.” *IEEE International Conference on Automatic Face and Gesture Recognition*, 6th ser., 2004, 529-34.

[2] Iñigo-Blasco, Pablo, F. Díaz-del-Río, D. Cagigas-Muñiz, T. Murillo, S. Vicente-Díaz, J.L. Font-Calvo, and M.J. Domínguez-Morales. 2012. “Validation of Dynamic Human Body Recognition Algorithms based on Robot Programming by Demonstration methods using 3D cameras.”

[3] Kurakin, A., Z. Zhang, and Z. Liu. “A REAL TIME SYSTEM FOR DYNAMIC HAND GESTURE RECOGNITION WITH A DEPTH SENSOR.” *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, 1975-79.

[4] Andresen, Simen. “Hand Tracking and Recognition With OpenCV.” Entry posted August 12, 2013. Accessed May 10, 2017. http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/.

[5] Brenan, Michael, Saket Sadani, and Victoria Zhou. *Semi-Robust Detection and Following of Individuals*. N.p., 2016.

[6] Aldoma, Aitor, Zoltan Csaba Marton, and Federico Tombari. “Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation.” *IEEE Robotics & Automation Magazine*, September 2012, 80-91. Accessed May 10, 2017. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6299166&isnumber=6299141 .

[7] M. Van den Bergh *et al*., “Real-time 3D hand gesture interaction with a robot for understanding directions from humans,” *2011 RO-MAN*, Atlanta, GA, 2011, pp. 357-362.