Mr.HANDI - Human and digit identificaiton

Kenon Fachon(kjf43) Chris McNally(csm44) Hee Jung Ryu(hr99)

Abstract—**Mr.HANDI is a robot that follows a person around whenever he makes the corresponding hand gesture to do so. Mr.HANDI uses the Rovio robot system as its hardware side, and ROS as the software side. Our goal is to make the Rovio recognize two separate hand gestures: follow and stop. When the follow hand gesture is given, the Rovio will continue to track the human who gave the gesture and adjust its heading to keep them at the center of its vision. On recognition of the stop gesture, the Rovio will cease forward movement. The robot was capable of detecting a human and moving toward it.**

I. Introduction

THE interface between technology and its users is becoming increasingly complex as the technology becomes more advanced. A user interface that allows a person to use a robot with little or no technical training could be useful in many situations. Hand gestures are very easily taught to users, even when there is a language barrier. Commands given through such gestures could be quickly and easily taught to virtually anyone, opening up the accessibility of the system and making it easier to integrate into an environment.

The ability to interact autonomously with humans is another valuable trait for technology. Detecting or tracking a human allows a robot system to be aware of users in its space and, depending on the system's purpose, increase safety and/or function.

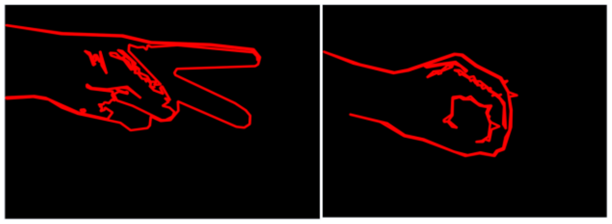
Our aim was to combine these two traits into a robot system capable of being commanded through hand gestures to track and follow a human in its visual range. The Rovio, the robot on which the system was built, has one camera available for visual sensing. Using this camera, we captured a feed of pictures and used them for the hand identification and human tracking functions. The results of these functions determined the direction and type of movement for the Rovio.

II. Hand Identification

The filtering process for hand images involves several steps. First, a Gaussian filter is applied to the image to help remove any noise. Then, a canny edge detection filter is used to find the edges in the image. A dilation filter is used to help fix any "breaks" in edges. The idea behind this is that due to noise and other errors, edges that are connected in the original image may not be after the canny filter is applied; the dilation filter is used to address this problem. Finally, we use openCV's function cgFindContours to obtain the most relevant edges from the data. The most difficult aspect of this part of our project was the selection of parameters for the various filters. We tried several different threshold values for the canny edge detection, as well as different kernel sizes for both the Gaussian and Canney filters. We also attempted to use other filters as well, such as a median filter and openCV's erode filter. Ultimately, the method we used was a good tradeoff between accuracy and computational complexity.

Chris McNally is with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14850. (csm44@cornell.edu)

Hee Jung Ryu and Kenon Fachon are with the Department of Computer Science, Cornell University, Ithaca, NY 14850. ({hr99,kjf43}@cornell.edu)



(a) (b)

Fig 1. Filtered hand gestures for (a) go and (b) stop

Our intention for hand classification was to use the filtered images' hu invariants for SVM training and classification. Hu invariants are calculated using the moments of an image, and thus do not change if an image is transformed in some way, such as a size or rotational transformation. Our hope was that this would allow our classifier to recognize a given hand signal regardless of it's position or rotation within the image. Our SVM used a polynomial kernel, and we had separate code for performing cross-validation on the SVM.

III. Human Tracking

The human tracking algorithm went through two iterations. The first iteration attempted to find a human face in the source image, and then use the Lucus-Kanad Method to predict movement through optical flow. This original version was very fast at tracking once the human was detected and very robust when tested on a webcam attached to a computer. However, when tried on Rovio which involves longer image receiving time, the algorithm could track the person of the interest anymore because the images received from Rovio had so much big time difference between images that the prediction of the optical flow in order to track the human was not accurate anymore. In addition, the amount of computation/the length of the algorithm slowed down the human detecting/tracking in a great time length.

In the second version of the human detecting/tracking algorithm, a focus was on calculation speed and compatibility with the Rovio's relatively slow stream of images. A second classifier haar-trained to detect upper body detection was added to compensate for this loss of robustness. After the addition of this other classifier, Rovio could detect face/human better even in the setting where the human's face is half or not visible in Rovio's monocular eye.

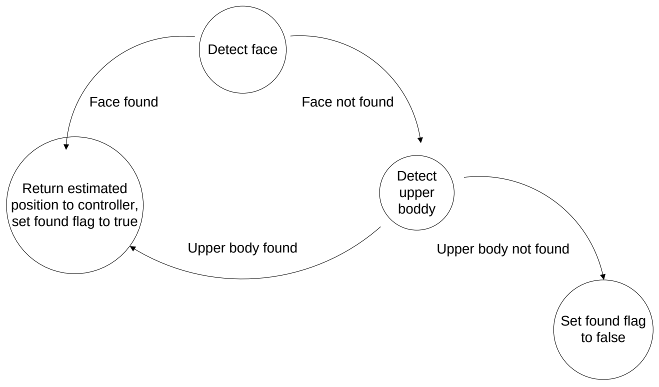


Fig 2. Program structure for improved human detection

In addition, the major reason of using haar training and cascade classifier is that they are quick in performance and compact in size of the trained classifier even when the actual train data size could be over 5000. In fact, the face detection trained data is based on about 5000 data images. In order to run the classifier, it takes only about a few lines of code and the time less than one or two seconds. Moreover, when training, haar training function has the way to generate unlimited number of more training data images from a limited amount of existing training data without having had to manual go and fetch more training data images.

The second revision of the human detection algorithm assumed two conditions: a target is visible in the Rovio's visual range, and a target is visually the largest object in the image received from Rovio. Some of experiments such as trial numbers 2 and 14 in the Fig. 4 on the next page were in fact tested on the conditions that did not necessarily satisfied the above conditions resulting in lower performance of accurately detecting the human target. However, other experiments such as trial number 14 were done under the circumstance that fulfills the aforementioned conditions thus resulting is much better accurate performance. In conclusion, when the two condition requirements are met, the algorithm can perform even better than 82.4% of accuracy thus having the two conditions satisfied is strongly recommended for using the algorithm on Rovio.

IV. Control

The control of the robot is handled through ROS and issuing HTML requests to the Rovio. The HTML requests are handled by a module provided by the CS4758 staff, called Interface. The controller written specifically for Mr.HANDI subscribed to the image message publisher in the interface module, and published a geometry twist message to the interface to issue movement commands to the Rovio. The image message was converted to an openCV format and passed to the hand classifier and human trackers. Each function returned values to indicate the next command state of the robot, whether a human was detected, and the estimated position of the human.

Using these values, the controller calculated the next command that should be issued to the Rovio, and set the current state as the next state predicted by the hand classifier. The human's estimated position was then compared to threshold values signifying the human was to the Rovio's left, in front of the Rovio, or to the Rovio's right. The instructions to be issued to the Rovio were then dependent on which command state the hand classifier predicted the Rovio should be in during the current iteration.

In the "go" state, if the human was determined to be in the left portion of the image, the Rovio was issued the command to travel left and forward, with the intention of bring the Rovio both closer to the human and center the human in the Rovio's vision. If the human was found to be in the center, the Rovio was issued a command to move forward. Finally, a human found in the right section of vision cause a command to move forward and right to be issued.

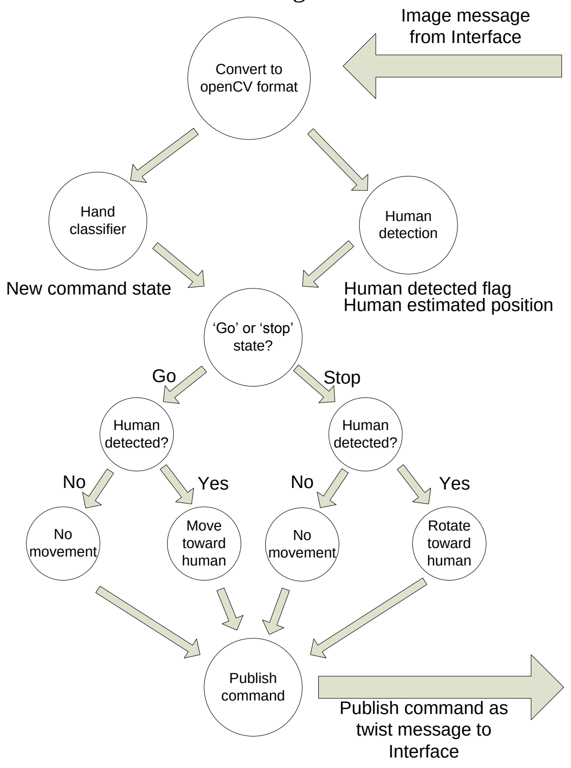


Fig 3. Program structure for controller

Two options were considered for the case when in the "go" command state, no human was detected: turn in a random direction for a full circle until a human was acquired again, or remain in place until a human was acquired. The first option would have provided a chance to seek out a human that was previously not in vision range. However, if the human tracking function merely lost lock on the human for a few images, then the Rovio could be turning away from the human it is suppose to be tracking. The ‘remain still’ option represented less utility, but offered better stability in the case when the human tracker was incorrect. The ‘remain still’ option was chosen for this reason.

In the "stop" state, the Rovio made the same determination as the "go" state for the position of the human, but instead of issuing forward commands, the Rovio was issued commands to rotate in place to keep the human centered. This kept the vision focus on the human in case a hand gesture command was given from the human being tracked.

V. RESULTS

A. Hand Identification

The various image filtering methods seemed to work well, as did the method of breaking down the image into key components using image segmentation. We were able to consistently produce data that was simpler than the original image, while clearly maintaining the shape of the hand. The SVM was much more difficult to implement; we began by working with openCV's SVM implementation, but stopped when we learned that this code was new and probably not reliable. We then switched to using the libsvm implementation of support vector machines. This was more effective than openCV in that the code actually worked; unfortunately, we were still unable to achieve statistically meaningful results with our SVM. This could have been due to a number of reasons. the size of our training set may not have been large enough. It could also be that hu invariants are not the best indicator of hand shape. Although the SVM's parameters (such as the selection of a kernel) may have also played a role in the issues we had, it seems unlikely given the SVM's performance.

B. Human Tracking

For the trial numbers 2, 7, and 12, the rovio moved to a completely wrong direction both because the rovio slipped bit by bit to a wrong direction due to the limitation of the control command we can make with the current version of the rovio and because rovio started to detect something else bigger after losing the person from its vision. Other than those three times, Rovio rarely missed the human and when it did miss him/her it redetected the person only one or two frames of images later. In these trials, the Rovio correctly identified the target human 82% of the time.

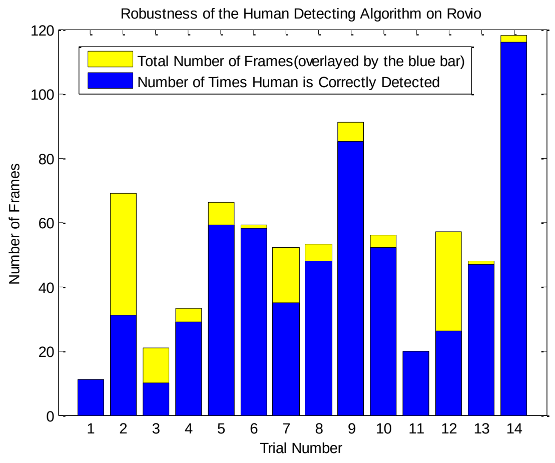


Fig 4. Accuracy of human detection

C. Control

The controller was able to direct the Rovio toward a detected human fairly reliably. If a human position was passed to the controller, the controller would move the Rovio toward it. We ran into some issues with overshooting when moving toward a human. Figure 5 shows the difference between the tracked human’s position and the center of the Rovio’s vision. This value tended to oscillate as the Rovio moved toward the human because of overcorrection. We attempted to correct for this with a wider forward section to reduce oscillations, but several constraints with our control over the Rovio interfered. The update rate of the Rovio was about an image a second, with images sometimes arriving fairly close together, and other times on the order of seconds apart. This made predicting how far the Rovio would go with one command difficult. The controller was changed to stop Rovio movement when no image was received to help reduce these oscillations. A video of the final result of the combined controller and human tracking algorithm was submitted to the course staff of CS4758.

Altering the speed of the Rovio was also not an option, our control over movement was either moving or halted in the specified direction. If speed was available as an option, a PID controller may have helped increase accuracy of keeping a human centered in the Rovio's vision.

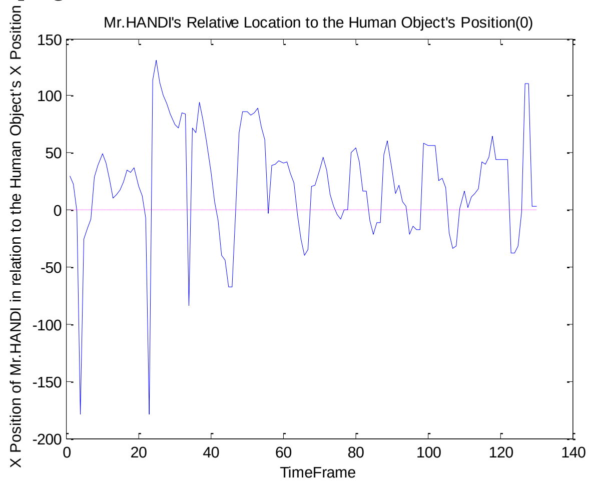


Fig 5. Dist. of human fron center of vision

VI. Conclusions

For this project, we aimed to create a robot system capable of recognizing hand gestures as a control input, and humans as goals to move towards. Both algorithms were first tested in a separate environment for performance, then integrated into ROS and Rovio. The human tracking algorithm proved reliable in the separate environment, but when first integrated into ROS/Rovio ran into issues with the slower update rate. A modified algorithm addressed these issues and provided more accurate detection. Control over the Rovio was successful, but limited by the available commands that could be passed to the Rovio, and by the Rovio’s method of movement.

Reference

[1] P. Viola, M. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*. 2001.

[2] Zhichao Chen, Stanley T. Birchfield. *Person Following with a Mobile Robot Using Binocular Feature-Based Tracking*. Submitted to IEEE/RSJ Internationl Conference 2007.

[3] Rainer Lienhart and Jochen Maydt. *An Extended Set of Haar-like Features for Rapid Object Detection*. Submitted to ICIP 2002.