Recognizing Military Gestures:

Developing a Gesture Recognition Interface

Jonathan Lebron

March 22, 2013

Abstract

The field of robotics presents a unique opportunity to design new technologies that can collaborate with humans to solve interesting problems. This is especially important in cases where a task is too difficult or dangerous, such as war. This project serves to bridge a gap within the field of Human-Robot Interaction by presenting a robust gesture recognition interface that can recognize gestures and translate these into actions for a robot to perform. The system uses the Xbox Kinect and Willow Garage's Robot Operating System (ROS) to recognize a set of seven military gestures. My system accurately recognizes complex sequences of gestures and performs a preset action for each gesture.

1 Introduction

Robots exist in a very small space in our world today. Despite the great potential they possess, they are not widely used throughout the world. We must aim is to find better applications for their capabilities and eventually implement them to perform everyday tasks. Nevertheless, one pressing issue we must address is making our interactions with these machines as efficient and natural as possible. The field which focuses on improving our communication with robots is called Human-Robot Interaction (HRI). My research looks to contribute to the field of HRI by presenting a robust gesture recognition interface. The motivation behind this research is the need for robotic systems to operate in less than ideal conditions where speech is not a viable or optimal solution (e.g. loud environments, gunfire exchange). Natural gestures, like hand movements and pointing, are one of the many ways we communicate as humans and by implementing an interface that resembles human-to-human interaction we are bridging a communication gap and making HRI more natural and intuitive. Nevertheless, gestures are inherently complicated and indeterminate; therefore, I have chosen to focus on military hand signals because they are a set of gestures with specific meanings. Unlike gestures used in everyday settings, military gestures map 1-to-1, for example there is only one gesture that signals soldiers to freeze. This allows me to maintain a controlled research environment. For the purpose of military tasks this gesture recognition interface is very useful for soldiers looking to communicate with the various robot systems that are currently being deployed in field ops [9]. The question I seek to answer with this research is: How do we give a robot a sequence of natural gestures to interpret and act upon? A sequence in the context of this project will be described as set of instructions given to the robot by the user to be executed in succession.

Outline Section 2 gives an overview of previous work and the current look of HRI. Sections 3 and 4 go over the progression of my system, from initial approach to final implementation. Section 5 presents experimental results and, finally, section 6 presents future work.

2 Background and Related Work

2.1 Image Manipulation

Gesture recognition has relied heavily on the advancement of image manipulation. Most systems today use some form of image manipulation to recognize patterns and establish relationships to specific gestures. William T. Freeman and Michal Roth [3] go over orientation histograms and methods of recognizing gestures based on image manipulation. This research shows some early issues related to image manipulation such as lighting and varying positions. Freeman and Roth used histograms of local orientations so their classifications would be independent of lighting changes and position. Image manipulation is critical to the improvement of gesture recognition as it allows us to focus on specific features that we want to track.

2.2 Multimodal Systems

More recently research has geared towards the implementation of multimodal systems; these are systems that take in more than one form of input and intelligently fuse these to better interpret a user's intentions. These systems tend to be very successful because they use more than one modality (e.g. speech and gestures) and intelligently combine these to better understand the user. Rogalla *et al* [6] developed a multimodal system using an event management system to delegate commands to a robot based on the current state of the environment and the user's speech and gesture input. The event management system transforms user input into actions and fuse incoming events that are related. The system was very successful, correctly classifying 95.9% of gestures. Another multimodal system was presented by R. Stiefelhagen *et al* [7], in which various methods are discussed for detecting gestures and combining these with speech commands. An interesting feature of this system is that it takes into account the users head pose, as this is a huge indicator of communication. Stiefelhagen explored HRI in the context of a kitchen scenario and used head pose estimation and point gesture recognition to successfully communication commands to a robot. Another multimodal system was explored in [5].

Figure 1: Microsoft's Xbox Kinect.

2.3 The Xbox Kinect

Microsoft inadvertently revolutionized robotics with its Xbox Kinect hardware (Figure 1). It provides a very cheap and easy to use solution for depth imaging and 3d point cloud data. K. K. Biswas [2] uses this hardware to detect various gestures. His method uses depth information from the Kinect and image manipulation to isolate the user from the background. Once this preprocessing occurs, the algorithm then focuses on regions of interest and creates a histogram of the current gesture. Finally a multiclass SVM is trained to recognize the gestures based on the histograms. The Kinect has also been used to develop a control-free interface for accessing medical images [4] and general human detection [8].

3 First Approach

This project began as a multimodal system before moving specifically into the gesture recognition domain. I initially planned to combine gesture and speech information to control a robot as was done in the work of Rogalla [6] and Stiefelhagen [7] however I soon realized this was beyond the scope of my thesis. Given a time constraint of 10 weeks and the complexity of the event management system necessary to control such a system, I moved away from a multimodal system to a strictly gesture-based system. What ultimately led to this decision was Cornell's research on military gesture recognition, which I based my project on.

When I was still in the beginning stages of my research on multimodal systems, I looked into Carnegie Mellon University's CMU Sphinx speech recognition toolkit. This is a cross-platform open source toolkit to develop speech recognition systems. The advantage of multimodal system over a unimodal system is that it is more robust because it analyzes information from more than one modality; so if say the gesture recognition

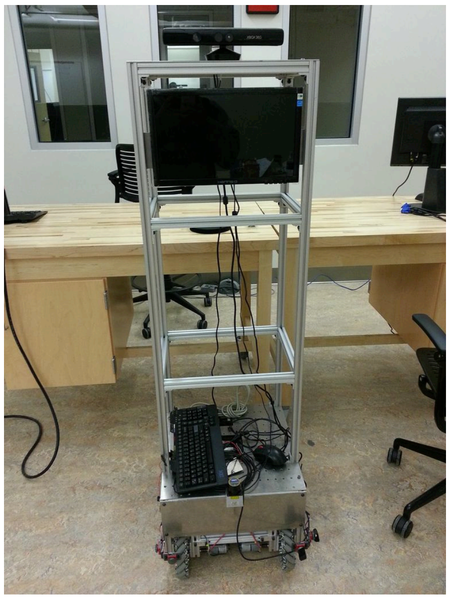

Figure 2: my gesture recognition interface was integrated into Union College's giraffe robot.

were to fail or generate erroneous data, the speech recognition would ccorrect the outlier and generate the correct output. A multimodal system would have been a better solution, but based on time and complexity it was out of scope for this project.

After deciding to stay away from multimodal system, I began looking into the MIT Kinect Hand Detection ROS package for more accurate gesture recognition. This package tracks the left and right hand of the user based on point cloud data. I also planned to use the pi face tracker package to track head pose and eyes similar to [7]. I ended up using a similar approach to Cornell's Personal Robotics Laboratory, which used the Xbox Kinect to track the user skeleton and used a classifier to recognize arm gestures. The following section explains the final implementation of my system and how it works.

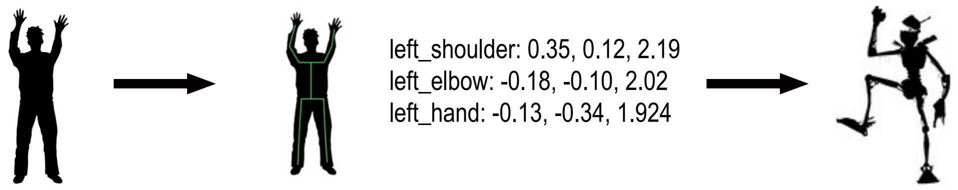
left_shoulder: 0.35, 0.12, 2.19
left_elbow: -0.18, -0.10, 2.02
left_hand: -0.13, -0.34, 1.924

Figure 3: Diagram of each step taken to perform a gesture recognition.

4 Methods and Design

4.1 Overview

I implemented my system on Union College's Giraffe robot seen in Figure 2. The system is composed of four phases: user input, recognition, classification and output. The user input phase begins when the user stands in front of the robot and performs a gesture. This is followed by the recognition phase where the Kinect reads in the X, Y, Z coordinates of the left arm of the user. We look specifically at the left arm because this is the arm used in the military to signal commands to soldiers. During this same phase these coordinates are transformed into angles. I explain what these angles represent and how they are generated in Section 4.3.2. The next phase is classification. The angles from the recognition phase run through a classifier that determines the gesture being performed and stores this gesture in a sequence. These first three phases repeat until the execute gesture is performed. Once the system receives the execute gesture, the output iterates through the sequence and sends commands to the robot for it to move. Figure 3 shows a high-level interpretation of this control flow.

4.2 Gestures

This system can recognize a set of seven military gestures. The gestures are STOP, FREEZE, LISTEN, RIFLE, COVER, STOP and ENEMY. As of now the system only allows the user to perform static gestures, but in Section 6 I present ways to implement dynamic gestures. The RIFLE gesture is arbitrarily used as an execute command to tell the system when to begin the output phase. An image of each gesture is provided at the end of this paper.

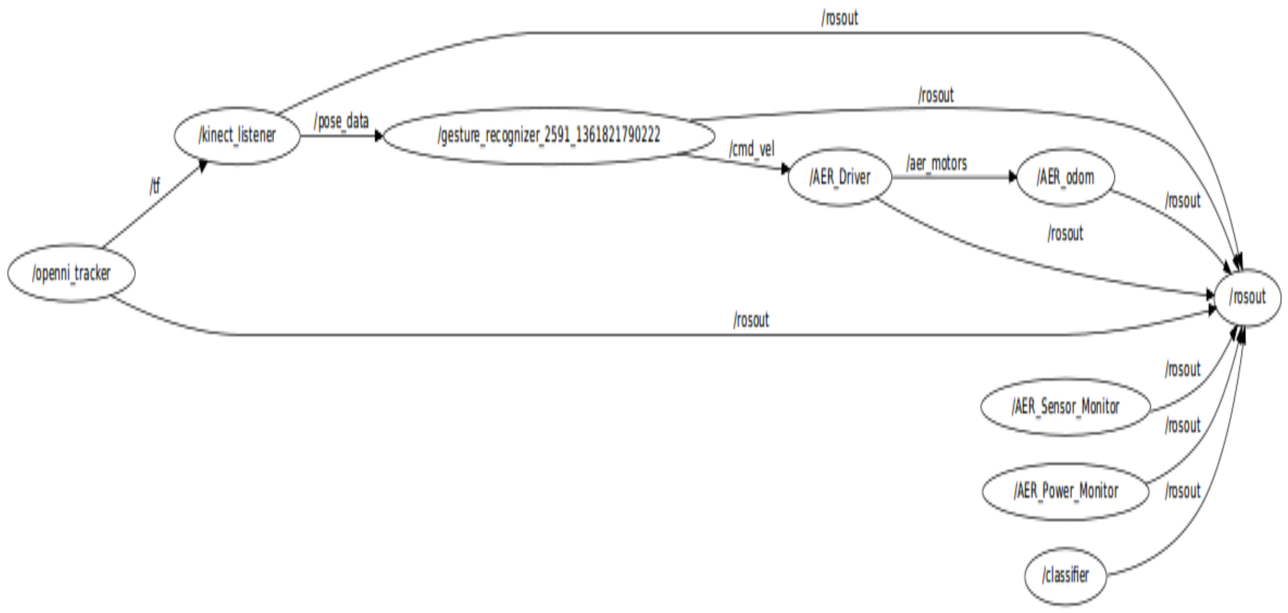

Figure 4: Graph showing the relationship of the nodes in the system

4.3 Nodes

The software design is composed of six main nodes inside the Robot Operating System (ROS). ROS is a platform developed by Willow Garage to standardize programming in robotics and make the process more efficient. ROS operates using nodes and messages. Nodes are executable files and messages are data; Nodes interact by publishing and subscribing to messages from other nodes. There are six main nodes operating in my ROS environment and they are *openni tracker*, *kinect listener*, *gesture recognizer*, *classifier*, *add gesture* and *AER Driver*. Figure 4 displays a graph generated by ROS of all the nodes in the system.

4.3.1 Openni Tracker

The *openni tracker* node is provided by OpenNI and it uses the Kinect hardware to track the user's skeleton (Figure 5). Using the Kinect's depth information, the node publishes frames containing X, Y, Z coordinates of different parts of the users body. Here is a sample output of openni tracker:

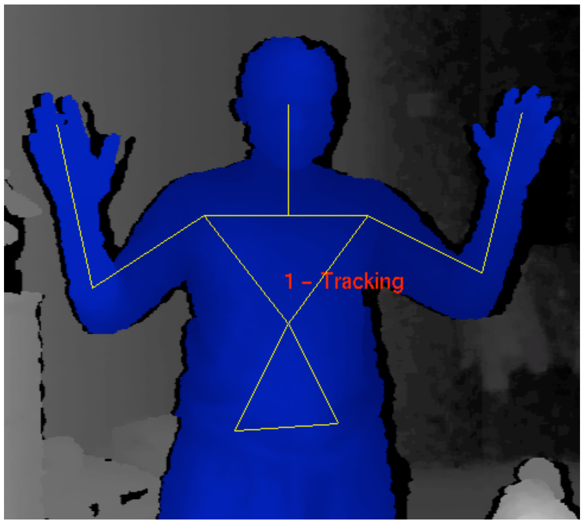

Figure 5: Visual of user's skeleton being tracked by the *openni tracker* node provided by OpenNI.

```
head: -0.208168, 0.362528, 2.247084 | 0.111806, 0.014298, 0.041540, 0.992759

neck: -0.190124, 0.141407, 2.196289 | 0.111806, 0.014298, 0.041540, 0.992759

torso: -0.173044, -0.091958, 2.143523 | 0.110194, 0.014691, 0.037494, 0.993094

left_shoulder: -0.358859, 0.12825, 2.1998 | 0.165864, 0.72583, -0.4794, -0.464566

left_elbow: -0.186129, -0.10265, 2.0264 | -0.325417, -0.581837, 0.49569, 0.556648

left_hand: -0.133285, -0.394494, 1.924446 | 0.00000, 0.00000, 0.00000, 1.00000
```

If you look closely the data is broken up into two parts, three values on the left and four values on the right. The values on the left are X, Y, Z coordinates, the values on the right represent rotation. My system only takes into account the values on the left and this data is handled by the *kinect listener* node.

4.3.2 Kinect Listener

The *kinect listener* transforms the X, Y, Z coordinates from *openni tracker* into the angles (θ_{hand}, ϕ_{hand}, ϕ_{elbow}, θ_{elbow}). This is done by using an algorithm to convert the coordinates from Cartesian to Spherical.

The angles $\theta_{hand}/\phi_{hand}$ represent the position of the hand in respect to the elbow and the angles $\phi_{elbow}/\theta_{elbow}$ represent the position of the elbow in respect to the shoulder. The reason for using these angles instead of the X, Y, Z coordinates provided by the kinect is so we can accurately classify gestures independent of the position and size of the user.

4.3.3 Gesture Recognizer

Essentially, the *gesture recognizer* node is a delegate node. It subscribes to the data published by the kinect listener and decides what to do with it. *Gesture recognizer* distributes the pose data generated in *kinect listener* to either the *classifier* node to classify the gesture, or to the *add new gesture* node to be stored in a training dataset. When *gesture recognizer* send data to *classifier* it receives a response message containing the gesture the user has performed, and based on this message it generates a message to tell the robot to move. These messages are composed in the form of cmd_vel ROS messages which take an X, Y, Z parameter.

4.3.4 Classifier

The *classifier* node takes in handPhi, handTheta, elbowPhi, elbowTheta and uses this information to determine the gesture that is being performed by the user. The algorithm is a C4.5 decision tree created using the WEKA [1] machine learning suite. This node returns its output back to *gesture recognizer*. Here is an example output of C4.5 decision tree generated by WEKA:

```
handTheta <= 0.48988
|   elbowTheta <= 0.865855:  rifle (150.0)
|   elbowTheta > 0.865855
|   |   elbowTheta <= 1.029816:  listen (150.0)
|   |   elbowTheta > 1.029816:  freeze (150.0)
handTheta > 0.48988
|   handTheta <= 1.536201
|   |   elbowTheta <= 1.029816:  cover (150.0)
|    |    elbowTheta > 1.029816
|    |    |    elbowTheta <= 1.770026:  abreast (150.0)
|    |    |    elbowTheta > 1.770026:  enemy (150.0)
|    handTheta > 1.536201
|    |    handTheta <= 2.562422:  stop (150.0)
|    |    handTheta > 2.562422:  antigesture (150.0)
```

4.3.5 Data Collection / Add New Gesture

I developed a data collection node as a means to efficiently collect data and format it. This node takes subscribes to *gesture recognizer* to receive pose data messages and then formats and stores this data into a master CSV file containing all the training data for our system. This node collects 150 frames of pose data and adds it to the training dataset. Currently this node can collect data from the user and add it to the training dataset, however the decision tree in *classifier* has to be manually updated for the new data to take effect. Once I had this node running I thought it would be interesting to be able to have the option to add new data to the training set on the fly. From this idea came the *add new gesture* node. Although it was not originally part of the design of the system, I decided to add this feature because it would allow users to customize their experience with the interface and use gestures that are most useful to them. As of now *add new gesture* is not fully implemented because of difficulties using the WEKA suite inside of python.

4.4 Handling Errors

With a system that requires robot action, it is important to handle erroneous data so that the correct action is performed. My system handles this by conducting multiple recognitions before any command is sent to the robot. When the same gesture is recognized in sequence the system is more confident that it is the correct gesture. For the gesture to be added to the final sequence it has to be correctly recognized 10 times consecutively. After a gesture is recognized, it is stored in an array of gestures. The system also handles errors for gestures that are incorrectly recognized when the user is not performing a gesture. This is accomplished by introducing the anti-gesture to the list of gestures that can be recognized. The anti-gesture is recognized when the user has their hands to their side, however it is not added to the sequence when recognized. This prevents the system from incorrectly recognizing gestures where the users hands are below the waist (e.g. STOP).

5 Results

As a result of this project, I have developed a gesture recognition interface that can successfully identify a set of 7 military gestures with a very high accuracy. This system is capable of handling a boundless sequence and execute an action for each gesture in that sequence. THe accuracy with which the system can recognize gestures is 82%. The system does very well in live demos however the accuracy is lower than expected because the system fails to recognize the LISTEN gesture correctly; it tends to confuse LISTEN with COVER. All other gestures are recognized at a clip of +93% aside from the LISTEN gesture. I provide suggestions as to how my system can achieve a higher accuracy in the following section. From here we can expand on the system and experiment with other domains outside the military (e.g. social robotics, medicine, etc.).

6 Future Work

Much work can be done to improve the overall effectiveness of the system. We can start with enhancing the *openni tracker* node. Before *openni tracker* can begin to collect data from the kinect it must first go through a calibration phase where the user has to maintain a Psi pose for a few seconds. This is not characteristic of a natural interaction and the system would definitely work better if it was able to omit this step and allow for fluid interaction. Another improvement would be handling of dynamic gestures. One way we can do this is to partition a dynamic gesture into static gestures and have our system recognize these as individual gestures. We can account for the dynamic gesture by treating the static sub-gestures as its own sequence and verifying that each gesture recognized is associated with the dynamic gesture. The *classifier* can also be modified to use a more sophisticated algorithm such as a State Vector Machine with feature vectors. This would allow us to recognize a wider range of gestures and with high accuracy. Finally, this system can be coupled with a speech recognizer to form a multimodal system for robust Human-Robot interaction.

References

[1] Weka webpage. http://weka.wikispaces.com.

[2] KK Biswas and Saurav Kumar Basu. Gesture recognition using microsoft kinect®. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 100–103. IEEE, 2011.

[3] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International Workshop on Automatic Face and Gesture Recognition*, volume 12, pages 296–301, 1995.

[4] L. Gallo, A.P. Placitelli, and M. Ciampi. Controller-free exploration of medical image data: Experiencing the kinect. In *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*, pages 1–6, June.

[5] Dennis Perzanowski, Alan C Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a multimodal human-robot interface. *Intelligent Systems, IEEE*, 16(1):16–21, 2001.

[6] O. Rogalla, M. Ehrenmann, R. Zllner, R. Becher, and R. Dillmann. Using gesture and speech control for commanding a robot assistant. In *IEEE INTERNATIONAL WORKSHOP ON ROBOT AND HUMAN INTERACTIVE COMMUNICATION, PISCATAWAY*, pages 454–459. IEEE Press, 2002.

[7] R Stiefelhagen, C Fugen, R Gieselmann, H Holzapfel, K Nickel, and A Waibel. Natural human-robot interaction using speech, head pose and gestures. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2422–2427. IEEE, 2004.

[8] Lu Xia, Chia-Chih Chen, and J.K. Aggarwal. Human detection using depth information by kinect. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15–22, June.

[9] Brian M Yamauchi. Packbot: A versatile platform for military robotics. In *Defense and Security*, pages 228–237. International Society for Optics and Photonics, 2004.

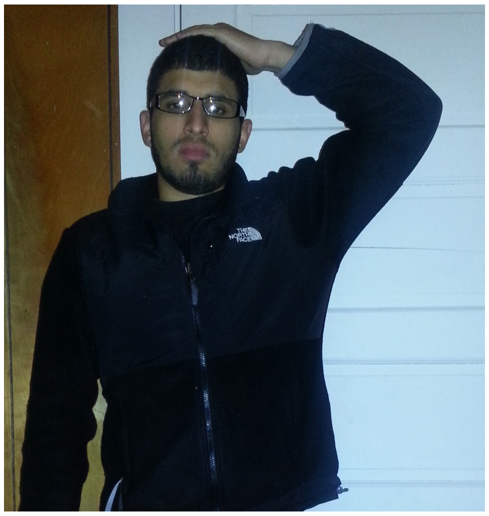

Figure 6: COVER gesture

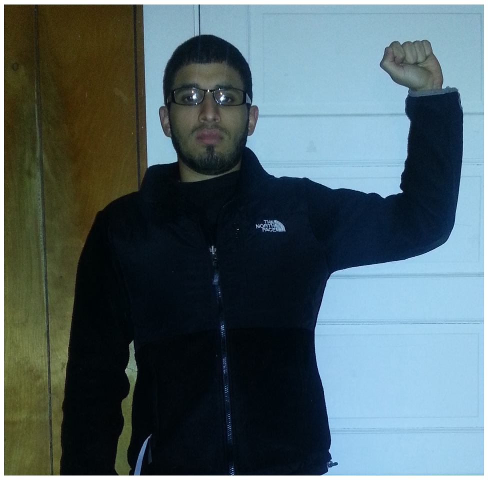

Figure 7: FREEZE gesture

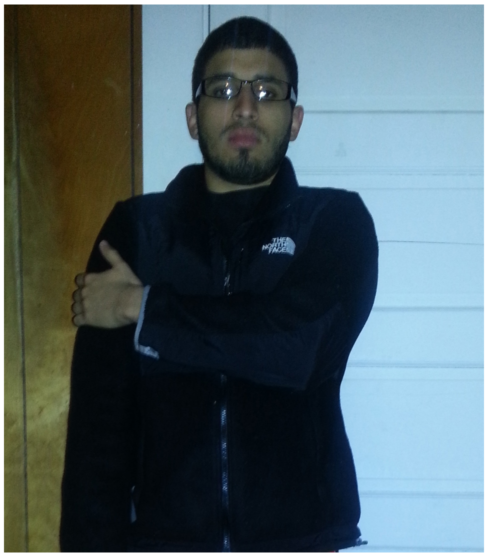

Figure 8: ENEMY gesture

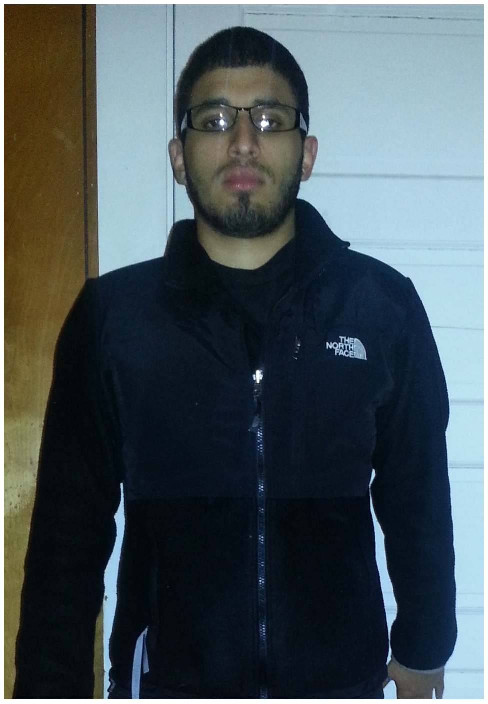

Figure 9: STOP gesture

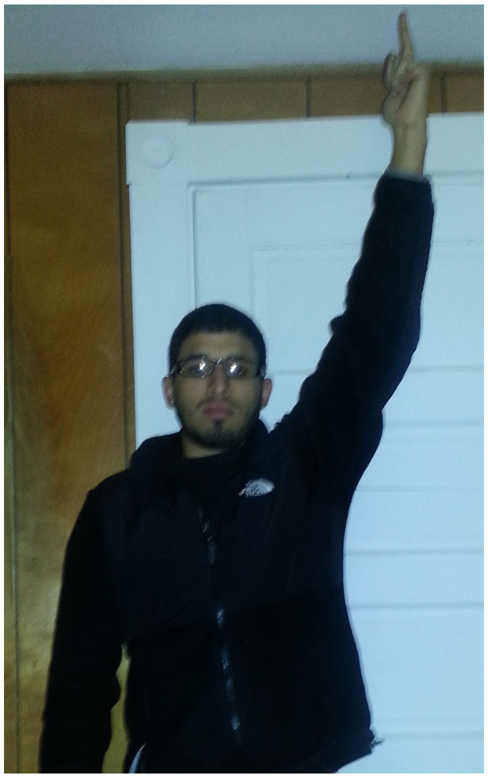

Figure 10: RIFLE gesture

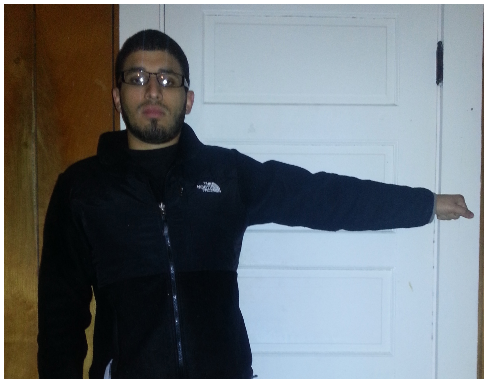

Figure 11: ABREAST gesture

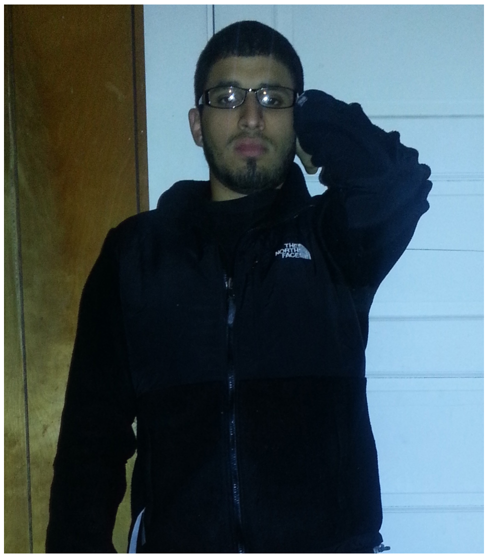

Figure 12: LISTEN gesture