Adaptive Gesture Recognition System for Robotic Control using Surface EMG Sensors

Benjamin Marcheix
Sciences Fondamentales et Appliquées
Université de Poitiers
Poitiers France
benjamin.marcheix@etu.univ-poitiers.fr

Bryan Gardiner
School of Computing, Engineering & Intelligent Systems
Ulster University
L'Derry, Northern Ireland
b.gardiner@ulster.ac.uk

Sonya Coleman
School of Computing, Engineering & Intelligent Systems
Ulster University
L'Derry, Northern Ireland
sa.coleman@ulster.ac.uk

Abstract—Traditionally, Electromyography (EMG) technology was used primarily in the medical domain for the investigation of neuromuscular normalities. However, the development of cheap surface EMG armbands have made this high-end technology commonly accessible to a much wider community. For example, providing gesture interfaces for gaming or controlling peripheral hardware devices. So far, research within this field has typically used complex machine learning classifiers, substantially large databases and long learning/training phases to develop applications-based approaches.

In this paper, a novel algorithm is presented based on one shot learning, i.e. requiring only one example per gesture, therefore substantially reducing the training time and database size. To assess the reliability and the usefulness of the developed system, the accuracy of the algorithm has been compared with classic machine learning approaches providing comparable accuracy. Additionally, the algorithm is successfully demonstrated via a robotic control experiment using various gestures for mobile platform and manipulator control.

Keywords—electromyography, adaptative gesture recognition, robotic control

I. Introduction

In recent years, the field of gesture recognition using surface EMG sensors has attracted a growing amount of interest due to the devices particularly low cost and the numerous applications in different kinds of fields such as robotic, sign language recognition and medical applications. In order to minimize crosstalk, it has been demonstrated in [1] that it is possible to use classic machine learning classifiers to accurately recognize gestures using data produced by EMG sensor armbands, using a SVM algorithm to classify gestures for controlling home devices. Additionally, in [2] an accuracy of 93% was obtained for basic gestures classification using a back-propagation neural network with supervised learning. The work that describes the most gesture recognition using surface EMG state of the art is in [8], which reaches an overall accuracy of 95% for user independent gesture classification using a SVM algorithm. They also describe their algorithm performances according to gesture groups sizes.

Moreover, Morais et al. [5] describe a way to use the Myo^{tm} armband from Thalamiclabs^{tm} in robotic applications using both the armband's EMG and accelerometer sensors to control the PeopleBot robot. However, their algorithm was only able to recognise a basic gesture set provided by Thalamiclabs^{tm} to control this robot. More generally, current systems still lack a high level of versatility and often require a lot of time for gesture control set up.

Fortunately, some one-shot learning classifier systems have been created such as the one developed by Li et al. [12]. This algorithm is a classifier that uses a Bayesian implementation of a probabilistic approach to recognise object categories patterns in images. Another one-shot learning approach is proposed by Vinyals et al. [13], which consists of a neural network adapted and trained to perform one-shot learning for classification. They found encouraging results using the Omniglot [16] and ImageNet [17] networks compared with the state of the art. Although these articles concern one-shot learning approaches, they are not related to robotic applications. Gardiner et al. [14] describes a minimalistic learning approach where they have implemented a novel approach to control code generation using a NARMAX modelling methodology. Although their application is specific to robot motor control, their system's training phase does not use one-shot learning.

To address this shortfall, this paper describes a new one-shot learning approach, similar to those described above, for surface EMG sensory data and hand and wrist gesture recognition. We provide a solution that is easily configurable, highly accurate and can readily interface with multiple robotic platforms. The remainder of this paper is organized as follows. The technical aspects of the developed algorithm are explained in Section II. The system has been tested and evaluated compared to existing similar algorithms in Section III, where the developed algorithm has been adapted for different robot control scenarios. Conclusion and future work are presented in Section IV.

II. Methodology

A. Environment

To develop the system, it was chosen to use the Robot Operating System (ROS) to make the project easily adaptable to any robot that utilises ROS. The algorithm has been developed using Python because it suits for system prototyping and it is supported by ROS.

The following research has been realised on an Ubuntu 18.04, 64 bit, running on a virtual machine. The Melodic Morenia version of ROS has been used. The virtual machine used a 4 core CPU (1.8GHz) and 11Gb of RAM memory (DDR3) with a Nvidia MX150 GPU.

The Myotm armband provided by Thalamiclabstm was used in these experiments to record EMG signals produced by forearm muscles. The armband is composed of 8 EMG sensors, evenly distributed around the interior of the armband, a gyroscope and an accelerometer. As shown in Fig. 1, the Myotm is positioned on the upper portion of the forearm to record both forearm and finger muscle contractions. For further details about the relationship between finger gestures and forearm muscles activity, please see [15].

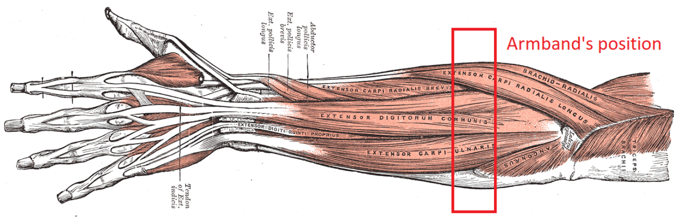

Fig. 1. Forearm muscle organisation and Myotm position.

For a more in-depth description of the capabilities of the Myotm armband, please see the work of Seema Rawat et al., [11].

B. System Technical Description

Fig. 2 depicts the overall ROS network architecture of the proposed system. Firstly, to use the Myo armband for custom gesture recognition, it is necessary to access the raw sensory data from the eight EMG sensors present around the interior of the Myo armband. The raw sensory data should be accessed in real time and should be accessible via a node within the ROS network. To achieve this, we use the ROS package [7] which manages the Bluetooth protocol between the host PC and the armband. This is represented as the "/myo_raw" ROS node in Fig. 2.

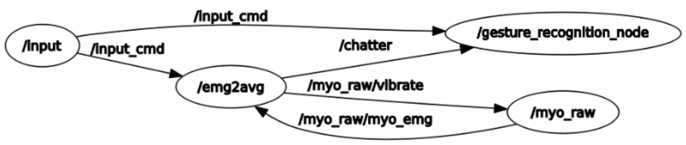

Fig. 2. Developed ROS package RQT graph.

Next, it is necessary to use the captured raw data to identify combinations of finger, hand and wrist movements to successfully identify both basic and complex gestures. The proposed algorithm to achieve this encompasses two phases; the learning phase (represented as the /emg2avg node in Fig. 2) and the real-time gesture recognition phase (represented as the /gesture_recognition_node in Fig. 2). The learning phase is where a one-shot learning approach will be developed to associate sensory outputs to custom gestures. This is conducted as follows. For each gesture, 100 samples are captured for each of the 8 EMG sensors at intervals of 20ms, across a 2 second window (Step 1 of Fig. 3). Each set of samples for each EMG sensor is then smoothed to eradicate any outlier sensor readings, using:

$$s_i = \frac{1}{n}\sum_{j=1}^{n} x_j \qquad i = 1, \ldots, 8 \tag{1}$$

where s depicts each EMG sensor, n is the number of samples and x is each sensor value. The 8 averaged values are then added to the database associated to that specific gesture (Step 2 of Fig. 3). Finally, the system enters a pause state where it waits for another gesture to be added (Step 3 of Fig. 3.)

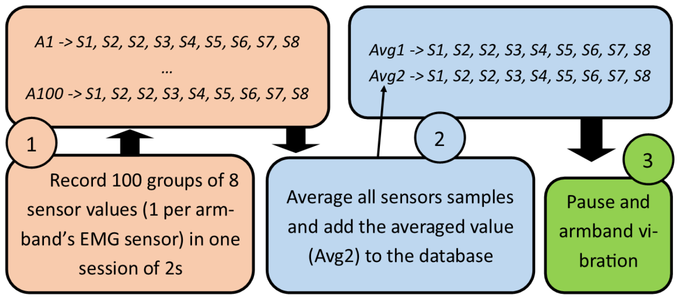
A1 -> S1, S2, S2, S3, S4, S5, S6, S7, S8
...
A100 -> S1, S2, S2, S3, S4, S5, S6, S7, S8
Avg1 -> S1, S2, S2, S3, S4, S5, S6, S7, S8
Avg2 -> S1, S2, S2, S3, S4, S5, S6, S7, S8
1
Record 100 groups of 8 sensor values (1 per armband's EMG sensor) in one session of 2s
2
Average all sensors samples and add the averaged value (Avg2) to the database
3
Pause and armband vibration

Fig. 3. Learning process for one gesture scheme.

This process is repeated for each gesture that is to be added to the gesture database, automatically linked to a gesture index between 0 and the maximum number of gestures, so when recalled, can be readily associated to a robot command. The storage of these values in the gesture database is shown in the top left section of Fig. 4, where each gesture (5 gestures in this example) is labelled as A1, A2, A3, A4 and A5.

The second phase describes the real time gesture classification process. During this phase, the algorithm will compare the incoming raw data flow with existing gestures stored in the database during the learning phase. Here, the best matched gesture is determined and selected as the result of the classification. This is represented in Fig. 4. The incoming raw data (A6) incorporates 8 EMG sensor values, which can be successively compared with all the existing gestures available in the database, calculating a matching score (M_{score}) for each. The matching score is calculated using the following equation:

$$M_{score} = \sum_{n=1}^{8} |I_{n\times1} - D_{n\times1}| \tag{2}$$

where I is the incoming data and D is the existing data. The lower the matching score, the closer the incoming data are to the database values. The gesture corresponding to the lowest matching score is then selected and published in the ROS topic "/rt_cmd" at a rate of 50Hz.

C. Accuracy Tests

In order to determine the system performance, a set of experiments was conducted using a number of participants completing a number of gestures. The gestures varied in complexity and are grouped into six gesture sets, as presented in Fig. 5. To ensure the system is tested to its full capability, gesture set E was introduced, which is suspected to be more difficult for the system to classify than the other gesture sets as the gestures included in set E consist of minimal muscle movement differences between gestures, making it very difficult to decipher between the sensory values.

Initially, five participants were selected to perform a set of gestures to determine the algorithm's accuracy. Each participant first completed the one-shot learning phase by making each of the gestures, which is subsequently stored in the gesture database. Next, each participant was asked to randomly reproduce each previously trained gesture 10 times and the algorithm's predictions were recorded. Each prediction has been categorized as true positive if the classifier recognized the gesture correctly and as true negative if the system recognized another gesture from the true one. These operations were then repeated for each of the six different gesture sets. To avoid muscle fatigue, ample rest periods were taken during testing for all participants involved.

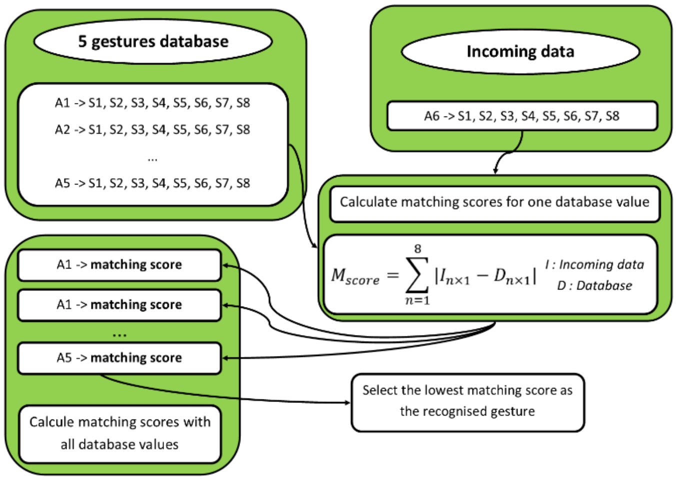
5 gestures database
A1 -> S1, S2, S3, S4, S5, S6, S7, S8
A2 -> S1, S2, S3, S4, S5, S6, S7, S8
...
A5 -> S1, S2, S3, S4, S5, S6, S7, S8
Incoming data
A6 -> S1, S2, S3, S4, S5, S6, S7, S8
Calculate matching scores for one database value
$M_{score} = \sum_{n=1}^{8} |I_{n\times1} - D_{n\times1}|$
I : Incoming data
D : Database
A1 -> matching score
A1 -> matching score
...
A5 -> matching score
Calcule matching scores with all database values
Select the lowest matching score as the recognised gesture

Fig. 4. Developed algorithm for real time classification phase.

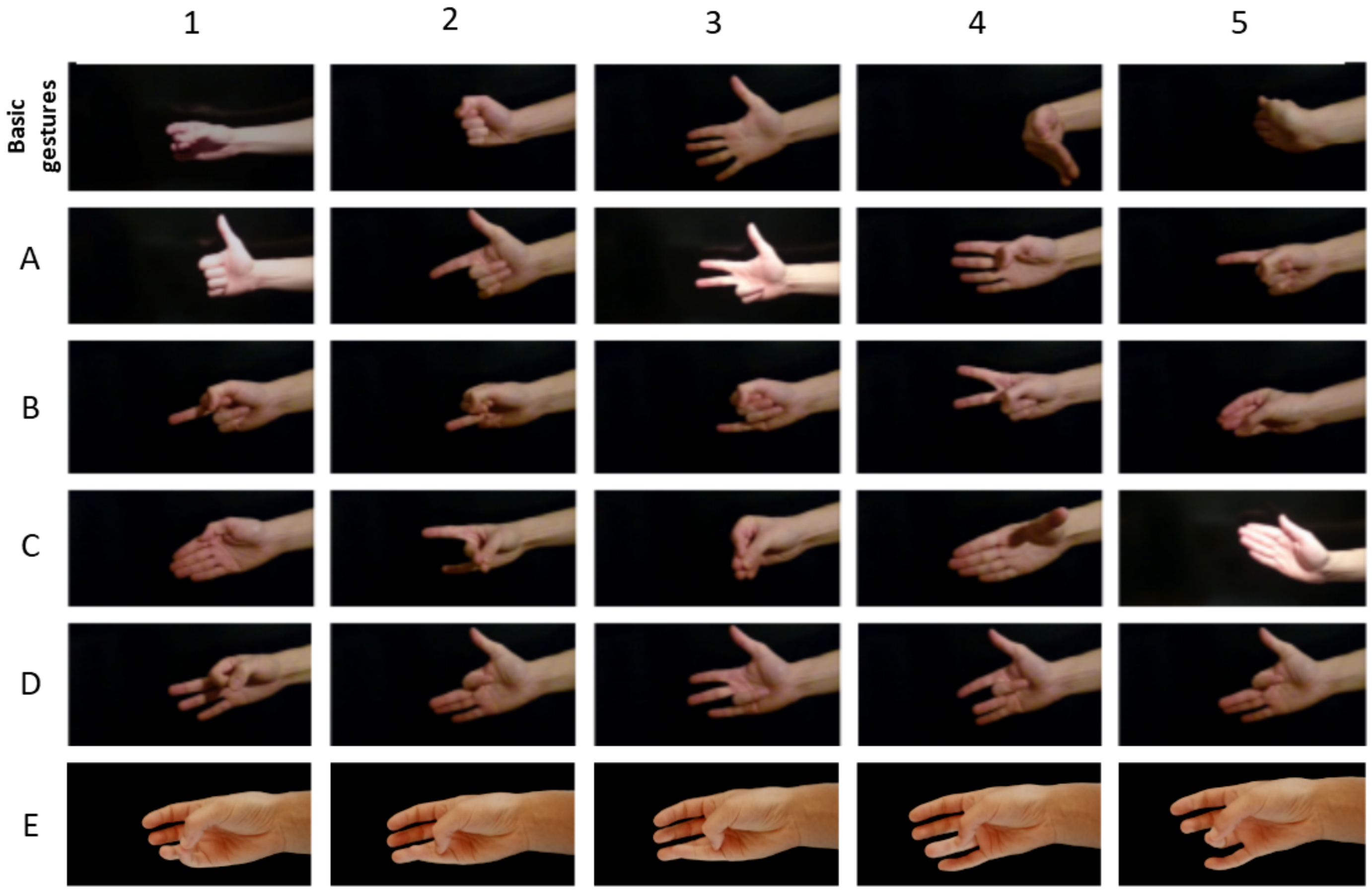

Fig. 5. Gesture sets used in algorithm accuracy testing. Basic gestures and sets A-D were also used for tests conducted in [8]

D. Robotic Experiment

To demonstrate the versatility and the wide range of different robots the system can adapt to, the proposed algorithm is utilized to simulate the control of two different robot platforms using Gazebo. The Turtlebot3 was used to demonstrate the systems capability for controlling mobile robotic platforms and the Tiago robot was used to demonstrate the capability of the system for controlling manipulator arm platforms.

As depicted in Fig. 6, a link node ("/rt2turtle") has been created to convert the classifier predictions to twist[1] messages published into the "/cmd_vel" topic which is subscribed by the robot simulation to trigger movements. This conversion node is important as it allows the input mapping between gesture groups and robot's movements. For example, when the first gesture is recognized by the classifier, it will send the value 0 to the "/rt2turtle" node which will convert it to a specific command that depends on the mapping.

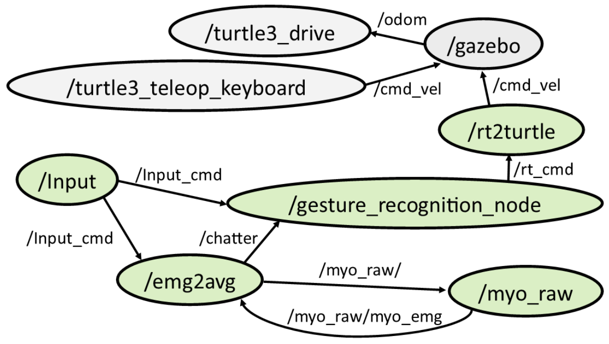

Fig. 6. Integration of the developed ROS package (green nodes) for control of the Turtlebot3 robot platform (grey nodes).

For the Turtlebot3, three different gesture sets have been mapped to control the robot's movements. First, gestures from the set of basic gestures (Fig. 8) were mapped to control the Turtlebot3 robot. Next, gestures from the gesture set A in Fig. 5 were used to control the robot. Finally, the last control mapping used different finger contractions when the participant's hand was lightly resting on an anti-stress ball (Fig. 7); this provided a more controlled finger movement when conducting each gesture.

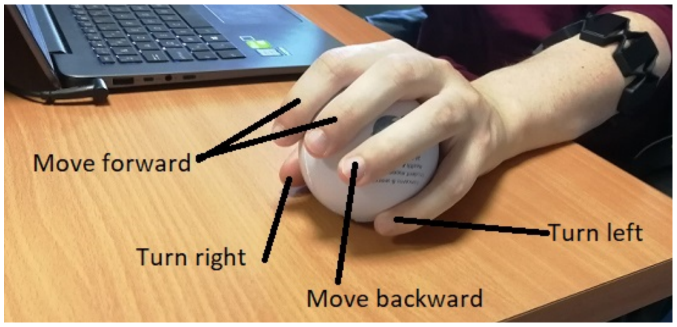

Fig. 7. Gesture mapping to control the Turtlebot3 movements with anti-stress ball.

To adequately evaluate the gesture recognition system, a path navigation experiment was set up to test each of the three gesture mappings described above and compare these with the use of keyboard controls to manoeuvre the robot.

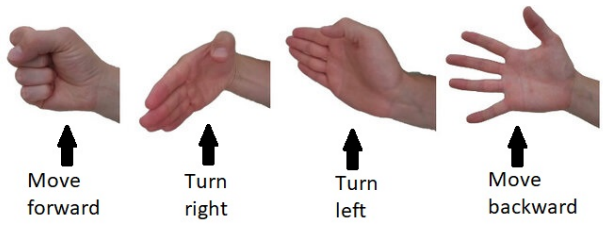

Fig. 8. Gesture mapping to control the Turtlebot3 movements with basic gestures group.

The aim of this experiment was to control the Turtlebot3 robot to traverse a specific path as shown in Fig. 9, and to do this in the most efficient time possible. It will then be possible to compare the developed system efficiency using different gesture sets with the classic keyboard teleoperation. The tests were conducted using five different participants where each had five minutes to get familiar with each of the four different command groups before beginning the experiment.

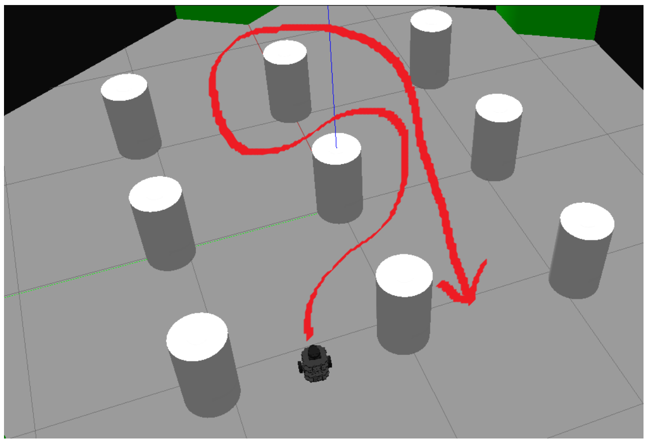

Fig. 9. Path the subjects were asked to follow.

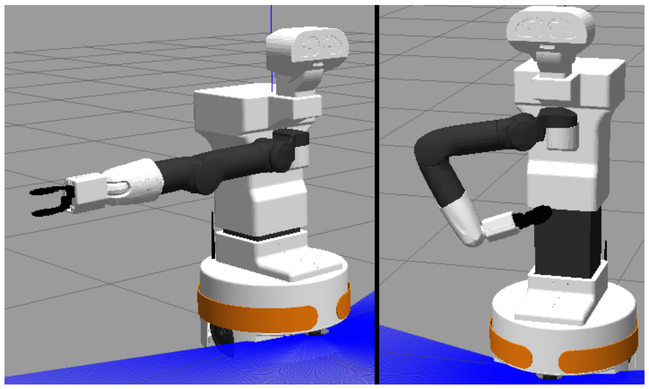

Fig. 10. Tiago robot articulated arm movements.

III. Results

A. Accuracy Tests

Fig. 11 shows the accuracy results of the proposed gesture recognition system and the associated standard deviation for each gesture group. The result average is 98.8% (Sd: 1.69%)

[1] Very common ROS message type

for successful prediction of the basic gesture set and 88.3% (Sd: 3.42%) for the gesture sets A to D in Fig. 5. For gesture set E, which is much more complex, the successful prediction average is 61.6% (Sd: 16.21%).

As a comparison, in the study in [8], a maximum accuracy of 95.64% (sd: 5.45%) was found with a gesture set very similar to the basic gesture set in Fig. 5. With this kind of gesture set, the proposed algorithm produces a more accurate result, obtaining 98.8% (Sd: 1.69%) accuracy.

The gesture set E recognition accuracy average is 61.6% (Sd: 16.21%) which is acceptable for this kind of gesture series due to the particularly high similarity of all the group E gestures. If the accuracy of a completely random classifier with a group size of 5 was considered, the rate of correct prediction would be 20%. Therefore, even with the demonstrated reduction of accuracy for the particularly complex gestures in set E, the proposed approach is still producing an accuracy that is 41.6% higher than if a random prediction would have been made. Also, it should be noted, as shown in Fig. 11, gesture set E has a very high standard deviation (16.21%) due to significant differences between all subjects.

The deviation of accuracy between the set of basic gestures, gesture sets A to D and gesture set E is likely to be associated with the difference in difficulty between these gesture sets and the variance of muscle movement required between each of the gestures within any particular gesture set. For example, the basic gesture set is considered easiest for the system to classify because of the very wide differences between the gestures in this set. When comparing this to gesture set E, the minimal muscle movement between each gesture within this set makes it much more difficult for the classifier to correctly make the correct gesture classification.

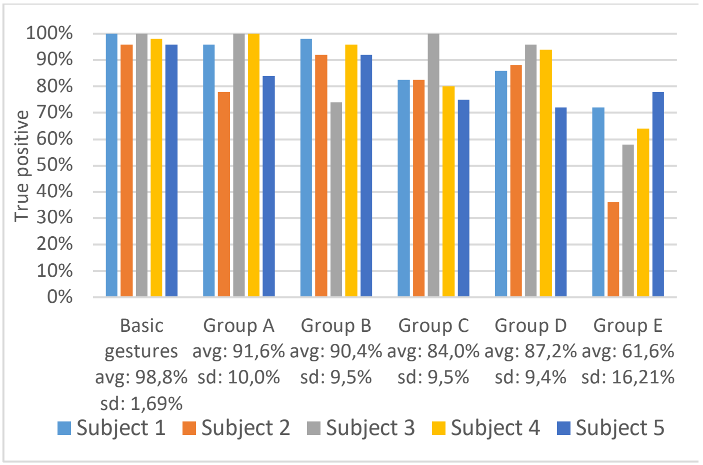

Fig. 11. True positive rate in function of the gesture group.

B. Robotic Experiments

Fig. 12 shows the results of the path navigation experiment with the Turtlebot3 robot. The average time participants took to finish the path (expressed in seconds) for each of the four gesture sets is provided.

It can be seen from Fig. 12 that some gesture sets such as the anti-stress ball (Fig. 7) and the basic gesture set (Fig. 8) seem to suit the control of the Turtlebot3 better than gesture set A. This could be due to the participants finding these gesture sets to be more intuitive for robotic control. It is easier to remember that a side finger gesture is associated with a turning robot movement with the anti-stress ball than a several fingers gestures as used in gesture set A.

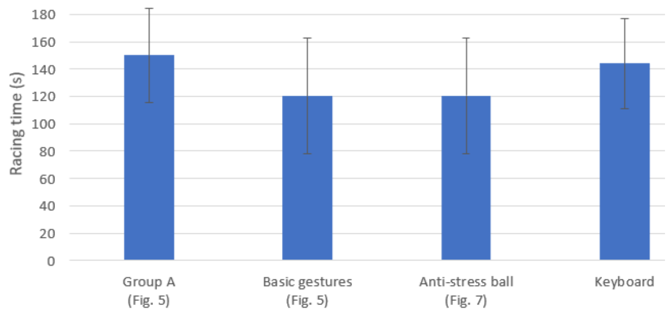

Fig. 12. Path traversal time utilising different mapped gesture sets.

It is also interesting to compare the keyboard control group with the gesture sets. There is on average a gain of 23.70s (Sd: 16.44%) when using the two most suitable gesture groups (basic gestures and anti-stress ball) when compared with using the keyboard, which is a very significant difference.

IV. Conclusion

A new versatile one-shot learning approach to classify both finger and wrist gestures is discussed. The main objective of the proposed strategy is to create a gesture recognition system that is easy to train (one-shot learning), does not require large database storage, is computationally efficient, yet still accurate when compared with alternative classic machine learning classifiers (e.g. SVM). It has been demonstrated that the developed algorithm is versatile in both the variety of gestures the system can detect and the capability to control various robotic platforms. Although some physical limitations were found as shown in the accuracy results, the one-shot learning approach offers a wide range of gestures that can be used to control many robotic platforms, using computationally inexpensive training and real-time classification. This enables the proposed gesture recognition system to run on a minimalistic computing platform such as the credit card sized Raspberry-Pi^{tm}, further enhancing the usability of this system for embedded robotic platforms.

References

[1] ImageNet – Stanford Vision Lab, Stanford University, Princeton University, May 2019. Available: http://www.image-net.org.

[2] B. Gardiner, S. Coleman, T.M. Mcginnity and H. He, "Robot control code generation by task demonstration in a dynamic environment," Robotics and Autonomous Systems, 2012, vol. 60, no 12, p. 1508-1519.

[3] B. Marcheix, “Complete ros code,” June 2019. Available : github.com/BenjaminMar/myo_code.

[4] F. Kerber, M. Puhl, and A. Krüger, “User-independent real-time hand gesture recognition based on surface electromyography,” 19th International Conference on Human-Computer Interaction with Mobile Devices and Services. ACM, 2017. p. 36.

[5] G. Morais, L. Neves and A. Masiero, “Application of myo armband system to control a robot interface,” BIOSIGNALS. 2016. p. 227-231.

[6] G. Pomboza-Junez and J. Terriza, “Hand gesture recognition based on sEMG signals using Support Vector Machines,” IEEE 6th International Conference on Consumer Electronics-Berlin (ICCE-Berlin). IEEE, 2016. p. 174-178.

[7] He, H., McGinnity, T. M., Coleman, S., & Gardiner, B. Linguistic decision making for robot route learning. IEEE transactions on neural networks and learning systems, 2014, vol 25(1), p. 203-21.

[8] Abreu, J. G., Teixeira, J. M., Figueiredo, L. S., & Teichrieb, V. (2016, June). Evaluating sign language recognition using the myo armband. In 2016 IEEE XVIII Symposium on Virtual and Augmented Reality (SVR), pp. 64-70.

[9] L. Fei-fei, R. Fergus, and P. Perona, “One-shot learning of object categories,” IEEE transactions on pattern analysis and machine intelligence, 2006, vol. 28, no 4, p. 594-611.

[10] L. Meier, “Ros tutorials,” May 2019. Available: wiki.ros.org/ROS/Tutorials.

[11] O. Jones, “Ros tutorials,” December 2018. teachmeanatomy.info/upper-limb/muscles/anterior-forearm.

[12] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu and D. Wierstra, “Matching networks for one shot learning,” Advances in neural information processing systems. 2016. p. 3630-3638.

[13] S. Ager. “Omniglot website,” November 2019. Available : www.omniglot.com.

[14] S. He, C. Yang, M. Wang, L. Cheng and Z. Hu, “Hand gesture recognition using MYO armband,” Chinese Automation Congress (CAC). IEEE, 2017. p. 4850-4855.

[15] S. Rawat, S. Vats, and P. Kumar, “Evaluating and exploring the MYO ARMBAND,” 2016 International Conference System Modeling & Advancement in Research Trends (SMART). IEEE, 2016. p. 115-120.

[16] S. Spasojević, T. Ilić, I. Stojković, V. Potkonjak, A. Rodić and J. Santos-Victor, “Quantitative assessment of the arm/hand movements in Parkinson’s disease using a wireless armband device,” Frontiers in neurology, 2017, vol. 8, p. 388.

[17] TJ Watson, “Running ros_myo and examples,” May 2019. Available : github.com/roboTJ101.

[18] U. Demirel, “Creating a generic hand and finger gesture recognizer by using forearm muscle activity signals,” PhD thesis, Middle east technical university, 2017.