ACTA IMEKO
ISSN: 2221-870X
December 2019, Volume 8, Number 4, 3 - 8

Mobile manipulator control through gesture recognition using IMUs and Online Lazy Neighbourhood Graph search

Padmaja Kulkarni[1], Boris Illing[1], Bastian Gaspers[1], Bernd Brüggemann[1], Dirk Schulz[1]

[1] *Fraunhofer FKIE, Fraunhoferstraße 20, 53343 Wachtberg*

ABSTRACT
Gesture-based control potentially eliminates the need for wearisome physical controls and facilitates easy interaction between a human and a robot. At the same time, it is intuitive and enables a natural means of control. In this paper, we present and evaluate a framework for gesture recognition using four wearable Inertial Measurement Units (IMUs) to indirectly control a mobile robot. Six gestures involving different hand and arm motions are defined. A novel algorithm based on an Online Lazy Neighborhood Graph (OLNG) search is used to recognise and classify the gestures online. A software framework is developed to control a robotic platform through integrating our gesture recognition algorithm with a Robot Operating System (ROS), which is in turn used to trigger predefined robot behaviours. Experiments show that the framework is able to correctly detect and classify six different gestures in real time with average success rates of 81.61 % and 81.67 %, while keeping the false-positive rate low by designing and using only 126 training samples.

Section: RESEARCH PAPER

Keywords: IMU-based gesture recognition; Online Lazy Neighbourhood Graph; OLNG; robot control

Citation: Padmaja Vivek Kulkarni, Boris Illing, Bastian Gaspers, Bernd Brüggemann, Dirk Schulz, Mobile manipulator control through gesture recognition using IMUs and Online Lazy Neighborhood Graph search, Acta IMEKO, vol. 8, no. 4, article 2, December 2019, identifier: IMEKO-ACTA-08 (2019)-04-02

Editor: Yvan Baudoin, International CBRNE Institute, Belgium

Received November 22, 2018; **In final form** April 2, 2019; **Published** December 2019

Corresponding author: Padmaja Kulkarni, e-mail: padmaja.kulkarni@fkie.fraunhofer.de

1. INTRODUCTION

For robots to be able to work in unstructured environments, areas dangerous to humans, or disaster sites, human intelligence is still vital. In such cases, the teleoperation of robots could be a solution. With recent advancements in robotics, the complexity of using robots has also increased. Nevertheless, currently used technology limits the majority of man-machine interfaces to text or GUI-based interfaces and joysticks. Such types of control can become cumbersome in the case of, for example, robots with a heavy control box or high degrees of freedom. Often, working in disaster areas could be stressful for an operator. Hence, alternate and intuitive control paradigms need to be developed.

With recent progress in the field of hardware and software, focus has been given to easy interfaces wherein non-expert users can efficiently use the system. In order for new users to be able to use otherwise complex systems, gesture-based control seems particularly useful as it can be very intuitive [1], [2]. Such a control can potentially eliminate the need for physical interfaces and enables richer human-robot interaction [3].

In well-established approaches for gesture recognition, vision-based gesture control is well researched, but the setup time and dependency on controlled environmental conditions, like lighting, make teleoperation unsuitable for disaster areas. On the other hand, Hoffmann et al. [4] developed an Inertial Measurement Unit (IMU)-based control for a robot manipulator, which does not need any infrastructure. They transformed human arm motions into corresponding robotic manipulator motions using five IMUs attached to the sleeve of a wearable jacket. They showed that teleoperation performed in this way is very efficient and intuitive [4]. However, this direct control method cannot be used to trigger some predefined manipulator motion or to trigger robot-based motions.

This paper presents an extension of the work done by Hoffmann et al. [4] in the area of wearable IMUs. We present a framework based on an Online Lazy Neighbourhood Graph (OLNG) search, which can identify and classify dynamic gestures in real time and can be used to trigger predefined robot motions. The main contribution of this work is the implementation and evaluation of an OLNG search-based algorithm for gesture recognition and robotics application. Additionally, we develop a software architecture that allows the integration of robotic platforms with our gesture recognition algorithm to trigger robot motions corresponding to the gestures with Robot Operating System (ROS) middleware. Prior to this OLNG search, the

algorithm was primarily used in the area of computer graphics [5].

2. RELATED WORK

Gesture recognition has applications in a wide range of areas, such as human-computer interaction, sign language recognition, gaming, household device control, and robot control [6]. Most approaches in the field of gesture recognition are based on vision, IMU, and Electromyography (EMG) signals [7], [8], [9]. Furthermore, depending on the gesture types, gesture recognition techniques can be divided into static and dynamic gesture recognition. Dynamic gestures are more difficult to recognise due to their temporal variations. However, they ascertain more natural interaction and are more practical [10], [11].

For IMU-based approaches, many use glove-mounted sensors. Mummadi et al. [12] used an IMU-based glove for real-time sign language recognition. They used various machine learning algorithms, such as Support Vector Machines, Naive Bayes, Multi-Layer Perceptron, and Random Forest, to classify the gestures. Wu et al. [13] used a data glove with perception and Hidden Markov Models (HMMs) to classify hand gestures. However, they only tested the results on simulated data. Georgi et al. [8] coupled IMU-based motion with EMG muscle activity to recognise hand and finger gestures. They used HMMs for the gesture recognition and obtained 74.3 % in accuracy with different users. Shin et al. [14] developed a system based on IMU and EMG sensors for controlling a mobile robot. They employed HMMs as the underlying algorithm for gesture recognition. However, these methods either only classified static gestures with the hand and fingers [8], [12], [14] or needed a huge database (about 1000 samples for each kind) [12].

In the domain of commercial products, the Myo armband by Thalmic Labs uses EMG signals along with IMUs to detect up to five different gestures and motions of the arm. Various learning approaches like Support Vector Machines [15], k Nearest Neighbour (kNN), and Dynamic Time Warping (DTW) [16] are used for gesture recognition with up to 86 % accuracy. However, a set of only five pre-defined gestures is supported.

For vision-based gesture recognition, Microsoft Kinect is widely utilised. OpenNI or Kinect SDK software are used for motion tracking [17]. For gesture identification, algorithms like DTW, HMMs, or Artificial Neural Networks (ANNs) are implemented. For an overview thereof, see [18]. Amin et al. [19] developed a vision-based technique to identify hand gestures using Principal Component Analysis (PCA) and Gabor representation. Yu. et al. [20] devised a control architecture for gesture-based control of UAVs using the ASUS Xtion camera. They defined nine gestures and used ROS for integrating gesture recognition with UAVs. They achieve a recognition rate of over 85 % in an indoor setting and with static gestures. Lai et al. [21] developed a hand gesture recognition system using the Microsoft Kinect camera with OpenCV libraries. They obtained an accuracy of about 95 %. However, vision-based approaches have limitations like occlusion and are vulnerable to bad performance due to ambient lighting and background changes.

Considering the limitations of vision sensors and the lack of commercially available robust sensors, IMU-based gesture control suits our control requirements.

3. APPROACH

In order to control a mobile robot with gesture recognition, a two-step procedure is necessary. The first step involves recognition of the gesture and the second involves designing the control architecture to trigger pre-defined robot behaviours. Accordingly, the approach description is divided into two sections. The first one explains the underlying algorithm we developed for gesture recognition and the second section describes the complete system, including the robot control framework.

3.1. Gesture recognition

We assume that the IMU readings are available in the form of vectors at a discrete time interval $(\dots, \vec{\alpha}_{t-2}, \vec{\alpha}_{t-1}, \vec{\alpha}_t, \dots)$, where $\vec{\alpha}$ is a vector of Euler angles and *t* is time. This vector is referred to as input vector in this paper. An underlying training database consists of Euler angles obtained from the four IMUs. A 12-dimensional vector forms a data point in the database. For building the database, sequences of such vectors labelled with the corresponding gesture names are saved while the gestures are being performed. Every vector in the database has a unique index *i*, which is later used for the identification of a particular vector.

For sequence matching, a window of *m* input vectors is defined. The distance between the current input vector and each vector in the training database is calculated, and from this set of distances, the indices of the *k* nearest vectors (kNNs) and their spatial distances from the input vector are obtained and stored using Fast Library for Approximate Nearest Neighbours (FLANN) in real time [22]. FLANN allows a fast nearest neighbours search by combining two existing approaches: i) *k*-means trees and ii) randomised *kd*-trees. Additionally, FLANN provides automatic selection of the algorithm and configuration of parameters for a given dataset and the desired precision for the task.

A matrix is created for the kNNs in the training database and for a window of *m* input vectors. Considering these $k \times m$ vectors as nodes, a graph can be created, as shown in Figure 1. For each input vector, its *k* corresponding graph nodes are then sorted in ascending order of their database index *i*. For the sequence matching, nodes are chosen in such a way that their database indices are in strictly ascending order. A path is a sequence connected from the first vector in the window to the last vector.

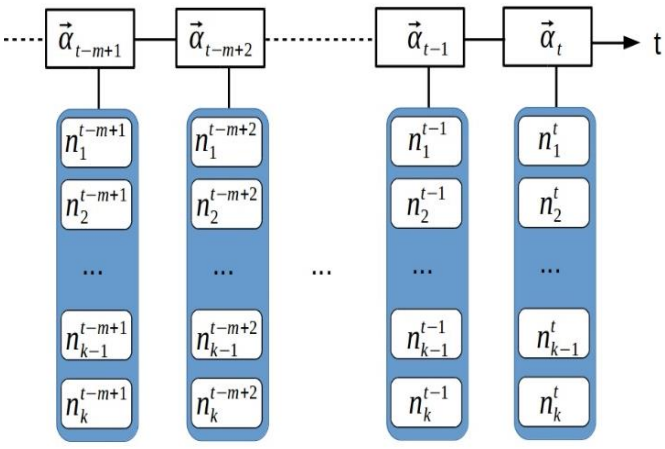

Figure 1. $k \times m$ nodes used for the construction of the OLNG. *k* is the number of neighbours in the window of the latest *m* sensor readings [5].

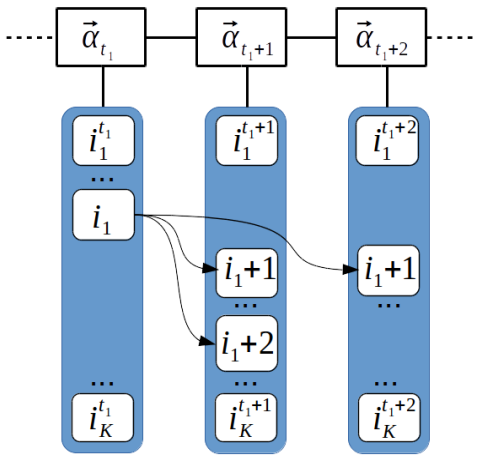

Figure 2. Path finding using OLNG search with *k* nearest neighbours The nodes of valid continuous paths are connected. The first valid path shown in the figure skips a column and hence it would be penalized [5].

Figure 2 shows the possible sequences for three input vectors. If no valid neighbour index for a particular input vector exists, then the neighbours of the next input vector are considered for the path. In that case, an extra path cost is added for skipping one input vector. In this way, all possible paths are listed, and the best path among them is chosen based on its minimum cost. OLNG search offers extremely fast sequence matching and is suitable for real-time applications due to its linear complexity [5].

For gesture recognition, the following procedure is applied.

1) kNNs to the current input vector are calculated using FLANN for the whole database.
2) OLNG is then used to find a matching and valid sequence for a window of m input vectors.
3) The best path found is saved along with its cost and sequence-matching length for comparison.
4) An arm and hand movement is considered to be a gesture if the sequence matching length is more than a certain threshold value and the cost of the path is less than another threshold.

Figure 3. A user wearing the jacket with IMUs attached to it and holding a serial device. The green button is used to start the gesture recognition and the red button is used to stop recognizing gestures.

Gesture recognition restarts when the robot signals the completion of a motion. All needed thresholds were chosen based on initial informal experiments and were tweaked and validated during our evaluation.

In the next section, we present the overall system architecture including the robot control.

3.2. Robot control architecture

The control architecture is such that the user controls with a serial device in their hand whether or not the gesture recognition is triggered. A user with this serial device is shown in Figure 3. The serial device has two buttons, coloured red and green, to start and stop the gesture recognition respectively. A flowchart of the control scheme used for achieving gesture-based control of a mobile manipulator is shown in Figure 4.

A user starts the gesture recognition by pressing a green button on the serial device. This enables data publishing with IMUs. The gesture recognition algorithm reads the IMU data and extracts movements in the specified window size, including the latest data. Then the algorithm checks if the input data corresponds to a gesture. If 'yes', then a pre-defined robot

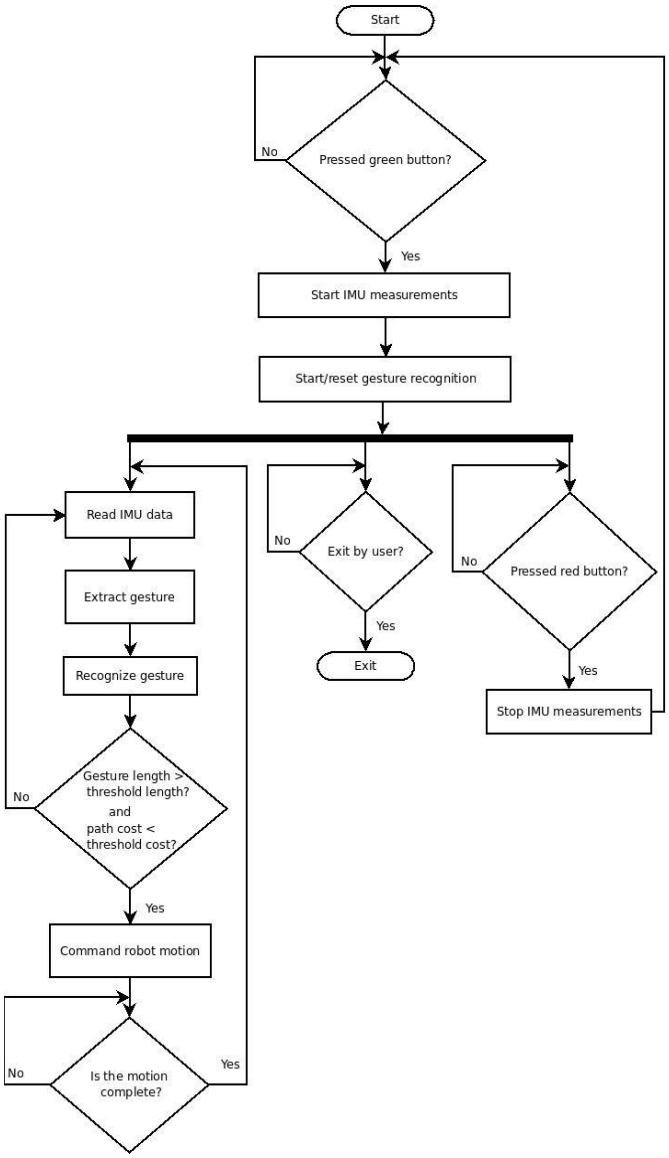

Figure 4. Complete system flowchart. After starting the gesture recognition, the stop signal for the gesture recognition and system exit is continuously checked in different threads using ROS.

motion is triggered. Otherwise, the input gesture window is modified to include the latest IMU data frame, and this new input window is checked for gesture recognition. Each recognised gesture triggers a specific motion of the robot. During the robot motion, no gestures are detected.

At the same time, a check for an exit signal from the user or a red button press is performed continuously. When the red button on the serial device is pressed, the IMU measurement is stopped, and the system waits for a green button signal from the user to start the gesture recognition process again. Until the green button is pressed, the user's motion has no effect on the gesture recognition system. When the system exit is signalled, the system needs to be restarted.

The programming was done in C++ using a Linux operating system and ROS as a middleware. ROS is a software architecture that supports robotic platforms and includes various libraries to plan and execute robot motions. For generating and commanding motions to the robot MoveIt! motion planning framework is used. Different software programs called *nodes* communicate in ROS with published *messages*. Using these messages, we establish a link between *gesture recognition node* and the *robot MoveIt! Interface*. MoveIt! is a ROS-integrated software architecture that provides collision-free trajectory generation for a robotic manipulator. These trajectories in turn can be used for various tasks, including picking up and placing objects while making sure that the geometric and hardware constraints of the robot are not violated [23].

4. EXPERIMENTAL SETUP

We validated our OLNG search-based algorithm by testing it for online gesture recognition. Xsens MTw sensors were used for the motion capturing and fo building a database. The sensor is a 9-axis IMU consisting of a 3-axis gyroscope, acceleration sensor, and magnetometer, and it includes an extended Kalman filter. Internally, the sensors operate at 75 Hz, but such a high rate does not offer much additional information during arm movements. Hence, we used a frequency of 15 Hz to update the input. Four such sensors were mounted on a jacket at the humerus, radius, hand, and finger.

The training database used consisted of 21 motions of each gesture from three different users for a total of 126 samples. The test database for obtaining the threshold cost values consisted of ten different motions of each gesture plus ten random arm movements. Once the threshold cost value was obtained, the algorithm was tested with a different dataset consisting of 20 motions of each gesture. For OLNG search we considered k = 40 nearest neighbours, a window of m = 10 vectors, and a path length of 9. These parameters were tweaked and fixed based on our initial informal experiments. Given these fixed parameters, the path cost was varied from 0 to 4 in steps of 0.05 square radians.

We defined six gestures to test our algorithm: hurry, drive, gather, disperse, forward, and down. These are based on internationally used hand signals. Figure 5 shows the gestures used. For the gesture 'Hurry', the user performs an up and down motion of the arm with a closed fist above the shoulder. For the gesture 'Drive', a closed fist left-to-right motion is performed. For the gesture 'Gather', the arm rotates in a circular way above the shoulder. For the gesture 'Disperse', the arm is moved up and down from the elbow keeping the hand straight. For the gesture 'Forward', the arm is stretched downwards and with the hand facing outside, the forward motion of the arm is performed. For the gesture 'Down', the hand is moved down from the wrist, keeping the rest of the arm steady.

To apply robot motions, a telemax manipulator from Telerob was used. The robot with the telemax arm is shown in Figure 6. The robotic arm has seven degrees of freedom, including one telescopic joint. The gestures defined are used to control the robot manipulator to perform the behaviours listed in Table 1.

Figure 5. (a)-(f) show the six different gestures defined in our experiment.

Figure 6. The robot hardware used for demonstrating the gesture-based control.

Table 1. The gestures and corresponding robot motions.

Gestures	Robot behaviour
Drive	Drive slightly forward.
Hurry	Increase the speed of the robot base.
Gather	Lookout position for the robot manipulator.
Disperse	Fold arm into transport position.
Forward	Extend arm forward for inspection.
Down	Decrease the speed of the robot base.

Table 2. Areas under the ROC curve for the gestures with variable threshold path costs.

Gestures	Robot behaviour
Drive	Drive slightly forward.
Hurry	Increase the speed of the robot base.
Gather	Lookout position for the robot manipulator.
Disperse	Fold arm into transport position.
Forward	Extend arm forward for inspection.
Down	Decrease the speed of the robot base.

5. RESULTS

The Receiver Operating Characteristic (ROC) curves for all six gestures with varying path costs are shown in Figure 7. The areas under these curves are defined in Table 2.
It can be observed that these areas are significantly above 0.5 (value for a random guess); hence, we conclude that the algorithm we used is able to distinguish the gestures well. Based on the ROC curves, the threshold path cost was chosen to be

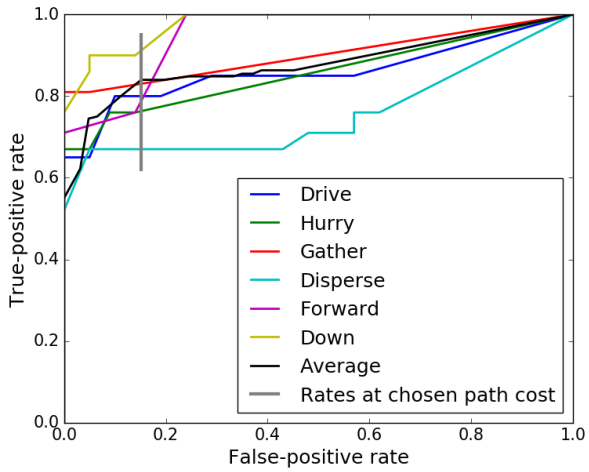

Figure 7. ROC curve for all gestures with variable threshold path costs. A path cost corresponding to high true-positives and low false-positives is chosen.

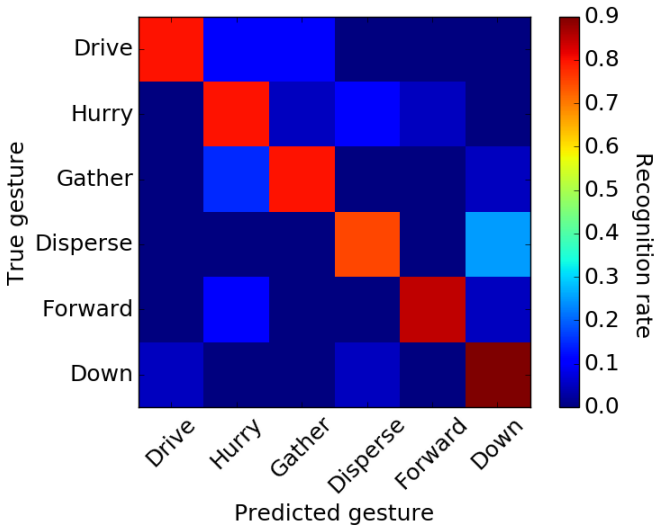

Figure 8. Confusion matrix for gestures with the chosen threshold path cost.

1.80 square radians with 81.61% true-positive rate and a false-positive rate of 15.12 %. By including random arm movements in the test database, all thresholds were also validated to only recognise a gesture when one was performed.

The confusion matrix corresponding to the chosen cost is shown in Figure 8. 'Down' was considered the best at the chosen cost, with a 90 % prediction rate. It can be observed that the gesture 'Disperse' is more difficult to recognise, as it shares similarity with the whole gesture 'Down'. Hence, the recognition rate of the gesture 'Disperse' was observed to be 75 %, the lowest among all other gestures. The same partial similarity is observed between the gestures 'Gather' and 'Hurry', which leads to the non-symmetrical confusion matrix. Nevertheless, we were still able to obtain a recognition rate of 80 % for both gestures.

The recognition rates for all the gestures are given in Table 3. It can be observed that the gesture recognition rate is above 75 % for all the gestures.

6. CONCLUSION AND FUTURE WORK

A novel algorithm based on OLNG search was implemented and evaluated for the application of gesture recognition in real time and used for mobile manipulator control. Experiments were performed to validate our approach for gesture recognition. Areas under the ROC curve for all gestures were observed to be higher than the random-guess value of 0.5. The highest area obtained was 0.97, and the lowest was 0.75, proving the viability of the algorithm. The results showed that we could obtain a gesture recognition rate of above 75 % for all six gestures, with a maximum recognition rate of 90 % at the chosen cost. The

Table 3. Gesture recognition rates with the chosen threshold path cost.

Gestures	Gesture recognition rates
Drive	80.0%
Hurry	80.0%
Gather	80.0%
Disperse	75.0%
Forward	85.0%
Down	90.0%

average recognition rate obtained in the later tests was 81.67 % compared to the initial rate of 81.61 % computed from the ROC curve, while keeping the false-positive rate low by design. A video showing our gesture recognition algorithm is available online at https://youtu.be/uILnMSN46dI.

As a part of this work, we also developed a software framework to trigger predefined robot motions based on a detected gesture. This framework was integrated with our gesture recognition in ROS middleware. The developed architecture is robot independent; hence, it facilitates the integration of various robotic platforms. The developed framework also allows easy definition of new gestures and their addition to the existing database.

In the future, we aim to expand our training database by adding correctly recognised gestures to it. We would also like to extend our algorithm to match parts of the start, middle, and end of a gesture to counter similarities between different gestures. This will enable us to further minimise false-positives and increase the robustness of the algorithm.

For additional efficiency of finding the nearest neighbours, nanoflann [24], an optimised version of FLANN, will be integrated into the algorithm in the future. From a software architecture point of view, we would also like to include user feedback in the loop before sending commands to the robot based on a detected gesture. This will enable us to avoid robot motions due to wrongly detected or wrongly classified gestures, making the control more robust to both the false-negatives of the algorithm and human errors.

REFERENCES

[1] E. Coronado, J. Villalobos, B. Bruno, F. Mastrogiovanni, Gesture-based robot control: Design challenges and evaluation with humans, Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2017, Singapore, pp. 2761-2767.

[2] B. Brüggemann, B. Gaspers, A. Ciossek, J. Pellenz, N. Kroll, Comparison of different control methods for mobile manipulation using standardized tests, Proc. of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, Linkoping, Sweden, pp. 1-2.

[3] S. Iba, J. M. V. Weghe, C. J. J. Paredis, P. K. Khosla, An architecture for gesture-based control of mobile robots, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1999, Kyongju, South Korea, vol. 2, pp. 851-857.

[4] J. Hoffmann, B. Brüggemann, B. Krüger, Automatic calibration of a motion capture system based on inertial sensors for tele-manipulation, Proc. of the International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2010, Funchal, Portugal, pp. 121-128.

[5] J. Tautges, Reconstruction of human motions based on low-dimensional control signals [Dissertation] University of Bonn, Bonn, Germany, 2012.

[6] Y. Chen, B. Luo, Y. Chen, G. Liang, X. Wu, A real-time dynamic hand gesture recognition system using kinect sensor, Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, Zhuhai, China, pp. 2026-2030.

[7] D. C. Kavarthapu, K. Mitra, Hand gesture sequence recognition using Inertial Motion Units (IMUs), Proc. of the Asian Conference on Pattern Recognition (ACPR), 2017, Nanjing, China.

[8] M. Georgi, C. Amma, T. Schultz, Recognizing hand and finger gestures with IMU based motion and EMG based muscle activity sensing, Proc. of the International Conference on Bio-inspired Systems and Signal Processing (BIOSIGNALS), 2015, Lisbon, Portugal, pp. 99-108.

[9] S. M. Mankar, S. A. Chhabria, Review on hand gesture based mobile control application, International Conference on Pervasive Computing (ICPC), 2015, Pune, India, pp. 1-2.

[10] L. Chen, F. Wang, H. Deng, K. Ji, A survey on hand gesture recognition, Proc. of the International Conference on Computer Sciences and Applications (CSA), 2013, Wuhan, China, pp. 313-316.

[11] J. S. Sonkusare, N. B. Chopade, R. Sor, S. L. Tade, A review on hand gesture recognition system, Proc. of the International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2015, Pune, India, pp. 790-794.

[12] C. K. Mummadi, F. P. P. Leo, K. D. Verma, S. Kasireddy, P. Scholl, J. Kempfle, K. V. Laerhoven, Real-time and embedded detection of hand gestures with an IMU-based glove, Informatics 5 (2018) pp. 28-45.

[13] J. Wu, W. Gao, Y. Song, W. Liu, B. Pang, A simple sign language recognition system based on data glove, Proc. of the International Conference on Signal Processing (ICSP), 1998, Beijing, China, vol. 2, pp. 1257-1260.

[14] S. Shin, D. Kim, Y. Seo, Controlling mobile robot using IMU and EMG sensor-based gesture recognition, Proc. of the International Conference on Broadband and Wireless Computing Communication and Applications (BWCCA), 2014, Guangdong, China, pp. 554-557.

[15] G. Pomboza-Junez, J. H. Terriza, Hand gesture recognition based on sEMG signals using Support Vector Machines, Proc. of the IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin), 2016, Berlin, Germany, pp. 174-178.

[16] M. Benalcázar, C. Motoche, J. Zea, A. Jaramillo, C. Anchundia, P. Zambrano, M. Segura-Morales, F. Palacios, M. Pérez, Real-time hand gesture recognition using the Myo armband and muscle activity detection, Proc. of the IEEE Second Ecuador Technical Chapters Meeting (ETCM), 2017, Salinas, USA, pp. 1-6.

[17] J. Suarez, R. R. Murphy, Hand gesture recognition with depth images: a review, Proc. of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2012, Paris, France, pp. 411-417.

[18] B. Nandwana, S. Tazi, S. Trivedi, D. Kumar, S. K. Vipparthi, A survey paper on hand gesture recognition, Proc. of the International Conference on Communication Systems and Network Technologies (CSNT), 2017, Nagpur, India, pp. 147-152.

[19] M. A. Amin, H. Yan, Sign language finger alphabet recognition from Gabor-PCA representation of hand gestures, International Conference on Machine Learning and Cybernetics (ICMLC), 2007, Hong Kong, China, pp. 2218-2223.

[20] Y. Yu, X. Wang, Z. Zhong, Y. Zhang, ROS-based UAV control using hand gesture recognition, Proc. of the Chinese Control and Decision Conference (CCDC), 2017, Chongqing, China, pp. 6795-6799.

[21] H. Y. Lai, H. J. Lai, Real-time dynamic hand gesture recognition, Proc. of the International Symposium on Computer, Consumer and Control (IS3C), 2014, Taichung, Taiwan, pp. 658-661.

[22] M. Muja, D. G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, Proc. of the International Conference on Computer Vision Theory and Applications (VISAPP), 2009, Lisbon, Portugal, vol. 2, pp. 331-340.

[23] S. Chitta, I. Sucan, S. Cousins, MoveIt! [ROS Topics], IEEE Robotics & Automation Magazine 19(1) (2012) pp. 18-19.

[24] Nanoflann, https://github.com/jlblancoc/nanoflann [Accessed on: 20.11.2018]