3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)
Proceedings of Engineering & Technology (PET)
pp. 233-238
Copyright IPCO-2016

Hand Gesture Recognition Using Leap Motion Controller for Recognition of Arabic Sign Language

Bassem Khelil[#1], Hamid Amiri[#2]

[#]*University of Tunis El Manar, Electrical Engineering Department, National Engineering School of Tunis*
SITI-LAB, Tunis, Tunisia
[1]khelil.bassem@gmail.com
[2]hamidlamiri@gmail.com

Abstract— **Introduction of novel acquisition devices, such as Leap Motion Controller (LMC) and the Microsoft Kinect system, allows to give a precise informative description of the hand pose, which can be exploited for accurate gesture recognition, namely hand gesture recognition of sign language. In this paper, we propose a novel method of pattern recognition to recognize symbols of the Arabic Sign Language (ArSL). Our proposal is based on the sparse data provided by the LMC sensor. The scheme extracts meaningful characteristics from the data, such as angles between fingers, to achieve a high-accuracy, which uses a classifier to decide which gesture is being performed. We show that our approach allows to recognize 28 static hand gestures of ArSL for letter "alif"-"yah" and digits 0-9 successfully. An experiment study of our approach is addressed and we show that recognition rate could be improved.**

***Keywords*— Hand Gesture Recognition, Leap Motion Controller, SVM, Arabic Sign Language.**

I. Introduction

A sign language is a communication method for deaf people. By using a sign language as an input interface to Information and Communications Technology (ICT) devices, it possible for becomes impaired people to hear, something which is hard to perform by using conventional keyboard or touch pad. A sign language uses visual information associated to fingers, hand and arm acts. At the same time, several fingers acts are used with a part of the face such as line of sight and mouth. Fingerspelling could signify one of the Arabic alphabet 28 letters in the form of fingers.

In the current research for Sign Language Recognition (SLR), the image recognition of colored images, depth images and hand shapes are used in [1]. Since it must be taken with colored gloves [1], the glove worn is not suitable. The image recognition requires long calculation time to detect the hand and the fingers. Therefore, it takes relatively a long interval to attain the final recognition result. In the case of the recognition with Kinect sensor [2], a large space is required for skeletal tracking. It is hard to recognize the fingerspelling anywhere with Kinect sensor. Therefore, SLR is required using a smart device like leap motion controller, which can easily recognize the shape of fingers or hands anywhere.

In this project, we propose a hand gesture recognition approach using LMC ([3]-[4]). This latter has skeletal tracking that recognizes the framework of fingers to obtain a highly accurate several data such as the index finger, the position of finger bones and the degree of the thumb. In addition, the use of LMC allows recognizing 28 static hand gestures of ArSL for letters "alif"-"yah" and digits 0-9 successfully in real time.

This paper is organized around seven sections: section 2 introduces the LMC. Section 3 presents the literature review of sign language recognition in details. The following section details our suggested gesture recognition system using LMC. Section 5 highlights the simulation results. This paper is enclosed by a conclusion and future perspectives.

II. Leap Motion Controller

The LMC is a compact device that can be connected to a PC using a USB. It uses InfraRed (IR) imaging to define the position of predefined objects in a limited space in real time. It can then sense hand and finger movements in the air above it, and these movements are recognized and translated into actions by the approach to be developed. The sensor software analyzes the objects detected in the device’s field of view. It recognizes hands, fingers, and tools, to permit reporting discrete positions, gestures, and motion [4]. The controller’s field of view is an inverted pyramid centered on the device, as represented by figure 1.The effective range of the controller extends from approximately 25 to 600 millimetres above the device. The controller itself is accessed and programmed through Application Programming Interfaces (APIs), with support for a variety of programming languages, ranging from C++ to Python and JavaScript. The positions of the recognized objects are acquired through these APIs. The Cartesian and spherical coordinate systems are used to describe the positions in the controller’s sensory space.

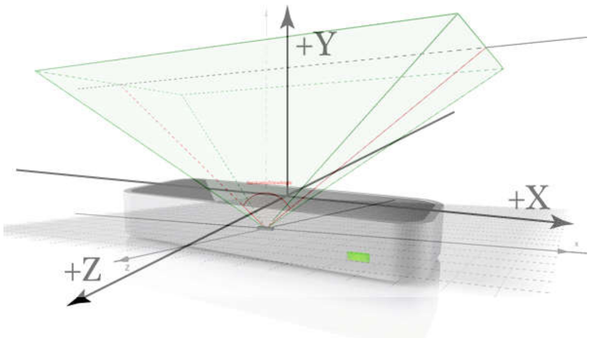


Fig. 1 Field of view and coordinate system of LMC

III. RELATED WORK

It has been proposed in the literature ([6]-[7]-[8]) a set of computer vision algorithms based on the color or intensity of images. Applications such as the camera mouse [5], tracking a certain keypoint in a stream of images to control a computer's pointing device while including the mouse, were based on RGB data. With the apparition of RGB-D (color images and depth maps synchronized) capture devices, using mainly the Microsoft Kinect sensor, the gesture recognition field had a great push forward. In this sense, a lot of works were addressed. For example, Sign language translation, representing a great challenge of Computer Vision, was tackled by [6]. In their work, X.chai et al used a 3D trajectory description of one sign language word and matched it against a gallery of trajectories. Another work of B.N. Estrela et al [7] used an RGB-D image from the Microsoft Kinect sensor to recognize the letters of the manual alphabet, known as fingerspelling.

The difficulty of hand gesture was discussed in the literature. In [9], [10] and [11], time of flight camera was used and [12] addressed the use of Microsoft Kinect sensor. A time of-flight camera computes the depth of an object by determining the time spent by the light to travel and come-back from the camera to the object of interest. These works used data from a point cloud and required further processing for hand detection before actually detecting gestures. ([9]-[10]) used simple methods to detect hands, assuming that the hand is inside a specific range of depth. Through LMC, we skip this step, because the Leap Motion already handles the detection by itself.

Unlike the above mentioned methods, which rely on dense data such as 3D point clouds, our proposed approach uses a little set of 3D points. These data are gone through the LMC's API [13]. In spite of the lack of a rich data set, our method performs high quality static gesture recognition.

IV. PROPOSED APPROACH

A. Working environment

The overall workflow of our proposed system is composed by 5 steps as illustrated by figure 2. The first step consists of setting up the programming environment, including Unity[1], SDK Leap[2], LMC and C#. Step 2 consists of collecting data to create a library of gestures. The third step concern the features extraction, these features are the introduced as a vector to a classifier, which generates a model for each gesture. Finally, the last step uses the trained classifier to recognize users' gesture.

B. Data collection

At this level, data is extracted from the LMC using the software development kit (SDK) associated with Unity. The LMC returns data representing the geometry of detected hand around its vicinity. The data contains information describing the overall motion of the hand. For our approach, we use the LMC to collect hand and finger data while the sign is performed in the LMC field of view. Among the features of the leap motion controller, we mention the tracking of the hand gestures by giving the 3d coordinate "x, y, z" of the palm and tip of each finger as shown by figure 3.

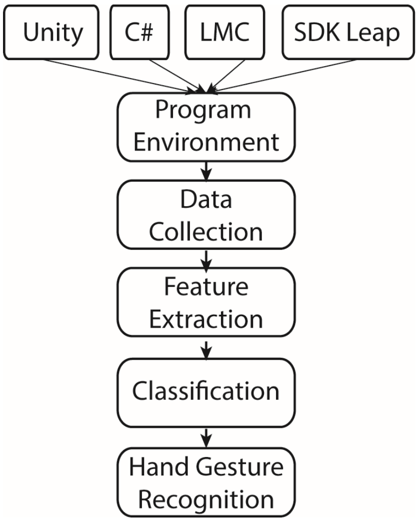


Fig. 2 Overall workflow

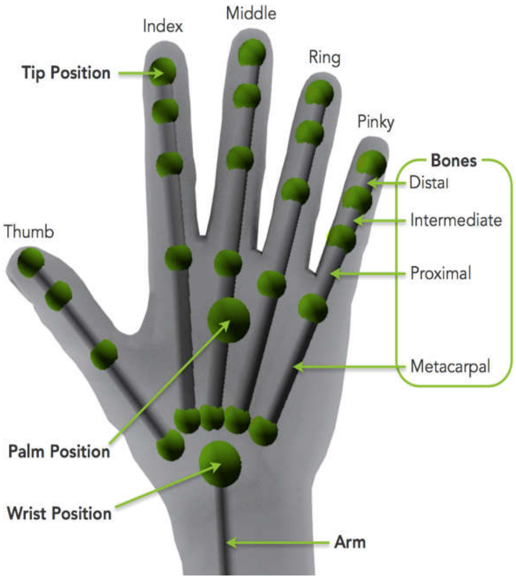


Fig. 3 Palm and tip positions given by the LMC [3]

C. Feature Extraction

The LMC returns frames of 3D data which contains basic tracking information. The data obtained from LMC are analyzed to extract robust features that can be used to identify the signs. LMC provides six 3D points: the center of the hand and the location of each visible fingertip. Having only six points at most, data is extremely sparse. Moreover, we have only a normal of the palm (located at its center) and the radius of the sphere created by the hand's curvature. Thus, for each gesture in a frame, we can read from the device up to eight three-dimensional points.

[1] https://unity3d.com/

[2] https://developer.leapmotion.com/

The absolute position of the hand and fingers are not relevant as gesture features, but they can be used to obtain other meaningful features. We calculate a feature vector based on the angle between two fingers located at p_i and p_j, the angle between a finger and the hand's normal and the distance from the center of the hand to each fingertip. These 16 features are more robust for gesture recognition than using only positional information, since they do not vary with the hand's shift in global position, but with the hand's shift in shape.

Let $i \in$ {thumb, index, middle, ring, little}, $c \in \mathbb{R}^3$ and $n \in \mathbb{R}^3$, be the index of a finger, with its fingertip located at p_i, the center of the hand and its normal, respectively. The vector of finger i is computed by:

$$\vec{v}_i = p_i - c \tag{1}$$

The feature vector F is created by concatenating 4 vectors. The first is built by calculating the norm of the finger's vector:

$$f_1 = \left[\left\| \vec{v}_{thumb} \right\|, \left\| \vec{v}_{index} \right\|, \ldots, \left\| \vec{v}_{little} \right\| \right] \tag{2}$$

The second vector contains the angles between the vectors of adjacent fingers:

$$f_2 = \begin{bmatrix} \left\langle \hat{v}_{thumb}, \hat{v}_{index} \right\rangle \\ \left\langle \hat{v}_{index}, \hat{v}_{middle} \right\rangle \\ \left\langle \hat{v}_{middle}, \hat{v}_{ring} \right\rangle \\ \left\langle \hat{v}_{ring}, \hat{v}_{little} \right\rangle \end{bmatrix} \tag{3}$$

The third vector has the angles between the finger vector and the hand's normal.

$$f_3 = \begin{bmatrix} \left\langle \hat{v}_{thumb}, \hat{v}_{normal} \right\rangle \\ . \\ . \\ \left\langle \hat{v}_{little}, \hat{v}_{normal} \right\rangle \end{bmatrix} \tag{4}$$

Where ||.|| gives the norm of a vector and <.>is the dot product (all vectors are normalized to compute the dot product)

In addition, we include a fourth vector, which is composed by two measures provided by the LMC that we considered relevant: the radius of the sphere created by the hand's curvature and the number of fingers detected:

$$f_4 = \left[radius_{sphere}, number\_fingers \right] \tag{5}$$

The final feature vector is denoted F and given by:

$$F = \left[f_1, f_2, f_3, f_4 \right] \tag{6}$$

D. Classification

For classification, we test Support Vector Machine (SVM) classifier. The SVM is one of the most common machine learning classifiers in use today. The method is chosen because it takes into consideration the state-of-the art method for many different applications. It was derived from learning theory and was widely used in object detection and recognition, text recognition, biometrics, and speech recognition. The original SVM is a binary learning technique with some highly elegant properties [14].

Given a training sample, the SVM constructs a hyper plane as the decision surface in such a way that the margin of separation between positive and negative examples is maximized. A notion central to the development of SVM algorithm is the inner-product kernel between a "Support Vector" x_i and a sample x drawn from the input data space. SVM has made its most significant impact in solving difficult pattern classification problems [15].

Let us consider the training sample:

$$\left\{ \left(x_i, d_i \right) \right\}_{i=1}^{N} \tag{7}$$

Where x_i is the input class for the i^{th} example and d_i is the corresponding desired response. Assuming the class represented by the subset $d_i = +1$ and the class represented by the subset $d_i = -1$ are "linearly separable", the equation of a decision surface in the form of a hyper plane that does the separation is given by:

$$w^T x + b = 0 \tag{8}$$

Where x is an input vector, w is an adjustable weight vector, and b is a bias. Thus we may have:

$$\begin{aligned} w^T x + b \geq 0 \, for \; d_i = +1 \\ w^T x + b < 0 \, for \; d_i = -1 \end{aligned} \tag{9}$$

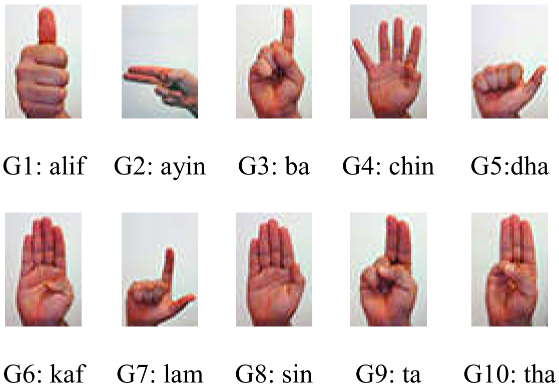
We assume linearly separable class to explain the basic concept of SVM. For a given vector w and bias b, the separation between the hyper plane defined previously and the closest data point is called the margin of separation denoted byρ. The goal of SVM is to find the particular hyper plane for which the margin of separation ρ is maximized [15]. For the case of non-separable classes, a new set of non-negative scalar variables, denoted by$\{\varepsilon_i\}_{i=1}^{N}$, are introduced into the definition of the separation hyper plane. The parameters ε_i are called slack variables, which serve to measure the deviation of a data point from the ideal condition of class separability. The goal here is to find a separation hyper plane for which the misclassification error is minimized.

$$d_i \left(w^T x_i + b \right) \geq 1 - \varepsilon_i, i = 1, 2, \ldots, N \tag{10}$$

V. Experimental results

In order to evaluate the performance of the proposed scheme, we acquired a dataset of gestures using the setup of figure 4. The performed gestures have been acquired with a LMC device.

The database contains 10 different gestures presented in figure 5 and realized by 10 different people. Each gesture is repeated 10 times for a total of 1000 different data samples. Every sign represents a descriptor of a frame captured by the device.


G1: alif G2: ayin G3: ba G4: chin G5:dha

G6: kaf G7: lam G8: sin G9: ta G10: tha

Fig. 4 Gestures from the ArSL contained in the database that has been acquired for experimental results

Our evaluation was performed in the created dataset, table 1 shows the confusion matrix for SVM. The diagonal of the matrix shows the correctly classified examples. The darker a cell is, the more examples it correctly classified for that class. Observing the confusion matrix, one can see that some gestures are more challenging than others. The accuracy is over 91,3% for all the gestures but alphabets "kaf" and "tha" are often confused with others. This is due to a limitation of the sensor, which detects them as a single pointing, moreover, there are not enough characteristics to differentiate between them. From this table, we can notice that alphabets "alif", "ayin" and "baa" that were critical for the LMC, reveal a very high accuracy when recognized from the device.

TABLE I CONFUSION MATRIX FOR PERFORMANCE EVALUATION

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	0,99	0,01								
G2		0,96	0,02	0,02						
G3		0,02	0,96	0,02						
G4			0,04	0,91	0,05					
G5				0,06	0,94					
G6				0,08	0,08	0,78			0,06	
G7							0,90	0,02		0,08
G8							0,09	0,86		0,05
G9					0,03				0,97	
G10							0,08	0,04		0,86

VI. CONCLUSIONS AND PERSPECTIVES

Hand gesture recognition for real-life applications is very challenging because of its requirements on the robustness, accuracy and efficiency. In this paper, we presented a robust part-based hand gesture recognition system using the LMC to obtain the number of fingers, hand sphere radius, fingertips, hand position and normal. These gestures were then used for sign language recognition. All of these features are used as a discriminative feature vector for a hand gesture. By using SVM classifier algorithm, this descriptor is evaluated and the class with highest confidence is assigned as the hand gesture. Our proposed recognition approach is able to recognize 28 static hand gestures of ArSL for letter "alif"-"yah" and digits 0-9 successfully with a correct classification rate of 91%. Furthermore, the high precision obtained for a number of different gesture and users indicates that our approach is reliable for Arabic Sign Language recognition.

However, our schema could be optimal if it complies with certain conditions. Firstly, only one hand at a time. Secondly, the hand needs to be within a specific depth range (~20cm). Thirdly, there should be no obstacle between the LMC and the hand. Finally, the palm should be close to perpendicular and pointing upward.

Given the restrictions imposed, there are numerous potential improvements to the approach as future work. The system can be developed to be able to utilize two hands, to recognize dynamic hand gestures and to recognize hands from different orientations and rotations. In addition, it can minimize real-time fluctuation in capturing image and outputting result and able to deploy the application for practical use.

REFERENCES

[1] M. V.Lamar, M. S.Bhuiyan, and A. Iwata,*"Hand alphabet recognition using morphological PCA and neural networks."* Neural Networks, 1999. IJCNN'99. International Joint Conference on. Vol. 4. IEEE, 1999.

[2] H. Du and T. To, *"Hand Gesture Recognition Using Kinect,"* Boston University, 2011.

[3] Leap Motion, *https://www.leapmotion.com/* (last: access: 2015-12-26).

[4] M. Spiegelmock. *"Leap Motion Development Essentials"*, Packt Publishing (2013).

[5] M. Betke, J. Gips, and P. Fleming, *"The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities,"* IEEE Transactions On Neural Systems and Rehabilitation Engineering, 2002.

[6] X. Chai, G. Li, Y. Lin, Z. Xu, Y. Tang, X. Chen, and M. Zhou, *"Sign Language Recognition and Translation with Kinect,"* IEEE Intl. Conf. on Automatic Face and Gesture Recognition, 2013.

[7] B. N. Estrela, G. Camara-Chavez, M. F. Campos, W. R. Schwartz, and E. R. Nascimento, *"Sign Language Recognition using Partial Least Squares and RGB-D Information,"*Workshop de Visão Computacional (WVC), 2013.

[8] A. W. Vieira, E. R. Nascimento, G. L. Oliveira, Z. Liu, and M. F. Campos, *"On the improvement of human action recognition from depth map sequences using Space–Time Occupancy Patterns,"* Pattern Recognition Letters, vol. 36, pp. 221–227, 2014.

[9] P. Breuer, C. Eckes, and S. Muuller, *"Hand gesture recognition with a novel ir time-of-flight range camera – a pilot study,"* Proceedings of the 3rd Intl. Conf. on Computer vision/computer graphics collaboration techniques, 2007.

[10] X. Liu and K. Fujimura, *"Hand gesture recognition using depth data,"* IEEE Intl. Conf. on Automatic Face and Gesture Recognition, 2004.

[11] J. Molina, M. Escudero-Vinolo, and A. Signoriello, *"Real-time user independent hand gesture recognition from time-of-flight camera video using static and dynamic models,"* Machine Vision and Applications, 2011.

[12] S. Oprisescu and E. Barth, *"3D Hand Gesture Recognition using the Hough Transform,"* Advances in Electrical and Computer Engineering, 2013.

[13] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, *"Analysis of the Accuracy and Robustness of the Leap Motion Controller,"* Sensors, 2013.

[14] M. H. Rahman and J. Afrin, *"Hand gesture recognition using multiclass support vector machine."* International Journal of Computer Applications 74.1 (2013).

[15] S. Haykin, *"Neural Network and Learning Machine"*, 3rd Edition, Prentice Hall; 2008.