

# Point Cloud Deep Learning Solution for Hand Gesture Recognition

César Osimani<sup>1</sup>, Juan Jesus Ojeda-Castelo<sup>2</sup>, Jose A. Piedra-Fernandez<sup>2</sup> \*

<sup>1</sup> Applied Research & Development Center on IT (CIADE-IT) Universidad Blas Pascal, Córdoba (Argentina)

<sup>2</sup> Applied Computing Group (ACG), Department of Informatics University of Almeria, Almeria (Spain)

Received 11 March 2021 | Accepted 11 March 2022 | Published 10 January 2023



## ABSTRACT

In the last couple of years, there has been an increasing need for Human-Computer Interaction (HCI) systems that do not require touching the devices to control them, such as ATMs, self service kiosks in airports, terminals in public offices, among others. The use of hand gestures offers a natural alternative to achieve control without touching the devices. This paper presents a solution that allows the recognition of hand gestures by analyzing three-dimensional landmarks using deep learning. These landmarks are extracted by using a model created with machine learning techniques from a single standard RGB camera in order to define the skeleton of the hand with 21 landmarks distributed as follows: one on the wrist and four on each finger. This study proposes a deep neural network that was trained with 9 gestures receiving as input the 21 points of the hand. One of the main contributions, that considerably improves the performance, is a first layer of normalization and transformation of the landmarks. In our experimental analysis, we reach an accuracy of 99.87% recognizing of 9 hand gestures.

## KEYWORDS

Artificial Neural Network, Computer Vision, Hand Gesture Recognition, Point Cloud.

DOI: 10.9781/ijimai.2023.01.001

## I. INTRODUCTION

**T**HERE is a high interest in using LiDAR scanners (Light Detection and Ranging) which use beams of light to measure distance to objects, allowing to acquire a three-dimensional point cloud of the environment [1]. The information acquired by this type of scanner combined with object color information is interesting for several applications (e.g., construction of three-dimensional models from the scanning of real objects [2], identification of objects within an environment [3], or self-driving cars [4]). Devices that combine color information (standard RGB cameras) and the data obtained by LiDAR scanners are more often called depth cameras or D-RGB (Depth - Red Green Blue) sensors. Among them we can find Kinect for Windows, Leap Motion Controller o Intel RealSense, which can be found in offices and homes as they are affordable. However, they are not consumer devices, as RGB cameras are.

If we get into the topic of Human-Computer Interaction and the constant effort to incorporate increasingly natural interactions, we find the commands by voice or through gestures of the face, body or hands. Let's focus on Computer Vision and the area of study related to hand gestures, particularly to one aimed at controlling Natural User Interfaces (NUI).

The identification of hand gestures can be interesting to create user interfaces with the aim of achieving better experiences, such as in augmented reality applications [5] overlapping virtual contents or digital information in an aligned way with the real image of the hand or applications to control devices. This identification of hand gestures is not trivial considering the hands and their fingers are, generally, occluded from each other, and their contours do not have high contrast.

In this work we propose the identification of 9 hand gestures by interpreting a cloud of 3D reference points obtained through a standard RGB camera. We introduce a neural network architecture which has the follow main advantages: a small number of hidden layers and high prediction hit rate of hand gestures. In this way, we achieve good results in predictions and the possibility of working on CPU not only to make predictions but also to train the network.

The rest of the paper is organized as follows: section II describes the Related Work, section III explains our Proposed Work, section IV explains the results and section V includes the conclusions.

### A. Contributions

A deep learning model has been developed to recognize 9 hand gestures by analyzing a point cloud of sparse 3D landmarks of the hand. The network architecture has at its input a transformation and normalization layer that allows achieving very good classification hit rates, even when using third party datasets containing different user profiles and variable environments.

\* Corresponding author.

E-mail addresses: cosimani@ubp.edu.ar (C. Osimani), juanje.ojeda@ual.es (J. J. Ojeda-Castelo), jpiedra@ual.es (J. A. Piedra-Fernandez).

## II. RELATED WORK

### A. Point Cloud

A point cloud is a fancy name for a group of points in space (here we will refer to three-dimensional space, but the concept is extensible to any dimension). There are different ways to collect point clouds from the objects that exist in an environment, among the most common are LiDAR scanners, depth cameras or some models created with automatic learning to infer reference points for hands [6], faces [7] or skeletons of bodies [8].

Point clouds have been applied mainly in detecting objects as shown in the works described below. In [9] a framework called PointRCNN has been developed to detect 3D objects through point clouds. This framework consists of 2 phases: in the first one 3D bounding boxes are used to generate segmentation masks in a bottom-up architecture. The second phase is essential to improve the efficiency of this approach with the combination of semantic and local spatial features. In [10] VoxelNet is presented, which is a deep network to perform 3D detections, with the particularity of joining the feature extraction processes and the prediction of 3D bounding boxes in one phase, unlike PointRCNN where they were carried out in 2 phases. One of the main advantages of VoxelNet is that it does not perform hand-crafted feature extraction, which can be understood as features extracted from separate images according to a certain manually predefined algorithm based on the knowledge of experts. Features extracted with Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) are commonly known examples of hand-crafted features. Although the previous cases allowed to perform object recognition in a generic way, studies have also been done to focus on the detection of a specific object, that is the case of this work [11] where point cloud data has been applied to perform a vehicle detection in order to integrate it into an autonomous driving system. To achieve this goal, the authors have proposed a 3D convolutional network to improve performance in the point cloud detection task. However, they have also been used in gesture recognition, i.e., in [12] a recognition of hand gestures based on 3D and 2D representations to control a virtual world in 3D was proposed. The 3D features are based on the finger position in the point cloud, while the 2D features come from the outline of the hand drawn from a series of images. This system has the outstanding characteristics that it can recognize both static and dynamic gestures, where the algorithm used to classify static gestures has been Support Vector Machine, while Dynamic Time Warping has been used for dynamic gestures. In addition, in the evaluation process of this work, a 95% success rate was obtained for static gestures and 81.34% for dynamic gestures.

### B. Classification and Segmentation of Point Cloud

Once a point cloud has been collected, it may be necessary to isolate the different objects, that is, to perform a segmentation, or also to classify each of those objects. Deep Learning has a good performance for classification of point clouds and this is demonstrated by the Multilayer Perceptron (MLP) called PointNet [13] that achieves an accuracy of between 80% and 90% for classification of point clouds using the dataset ModelNet40 [14] which contains 40 classes of objects such as chairs, desks, beds, tables and others. The point cloud of a chair is shown in Fig. 1.

PointNet has been applied in many studies, some examples are described then. This work [15] aims to improve PointNet to increase object classification performance which is the main use of this model. To achieve this objective, two actions have been carried out: one is to increase the number of hidden layers of the architecture and the other is to combine the softmax loss function with center loss. In this way, an accuracy of 89.95% has been obtained. In [16] the PointNet



Fig. 1. Point cloud of a chair.

network has been trained in order to verify the performance of this deep network in the human body segmentation task. To perform the segmentation in PointNet, the SMPL model is used, which offers a realistic 3D model of the human body. In this work two types of tasks have been approached: a segmentation and a classification task. In each task a different simplification of the PointNet architecture has been used. In the segmentation task, the points that have been located on the surface of the body are obtained, while in the classification task a binary classification is carried out to identify the body of a man and a woman. On the assumption of gesture recognition, Ge et al [17] propose a PointNet that has the purpose of processing the 3D point clouds to obtain a representation of the pose of the hand in 3D. This system is based on analyzing the 3D point cloud to obtain an estimate of the joints of the hand in 3D and to get better results, the points have been normalized so that it is insensitive to the variations that may arise from the location of each one of them. Furthermore, it has been possible to improve the precision of the position of the fingertips using a PointNet that obtains the neighboring points of the estimation of the location of the fingertips, having as a consequence that the model is more robust.

Among jobs that address the challenge of unsupervised learning with point clouds, we can find FoldingNet [18] and PointCNN [19].

In the same line appears PointNet++ [20] that adds a neighborhood of points to capture features that allow to group close points.

There are also works, such as [21], that improve the segmentation of the different objects in a point cloud by processing point clouds within a temporary space, that is, point clouds obtained from multiple instants of consecutive times.

Regarding the work that adjusts, aligns and superimposes a 3D model of a hand on the hand detected in a single image, we can find [22], which also proposes an approach to the automated collection of data from Youtube to incorporate them into the dataset in order to include data unrelated to the laboratory.

### C. Deep Learning in Gesture Recognition

PointNet is a deep network that has been used in this work to perform gesture recognition, but there are other proposals in Deep Learning that have also been applied to perform this type of recognition. In [23] the aim was to develop a framework to recognize human actions applying the Convolutional Neural Network (CNN). This system consists of two phases. In the first one the activities that involve single-limb are separated from those that are multi-limb to perform the classification of said activities in the next phase. In the classification stage, two CNNs were used to detect the two types of activities that were identified in the previous phase, obtaining a 97.88% hit rate. Khari et al [24] use learning transfer to do static gesture recognition. In this study, the VGG19 model has been trained with RGB and RGB-D images to identify of 24 gestures from the ASL dataset. This proposal has been compared with other models such as VGG16, CaffeNet or Inception V3, being the presented proposal in this work the one with the highest hit rate with 94.8%.

Another type of gesture recognition is based on using devices or sensors, which provide a set of data that are useful for such recognition [25]. deepGesture [26] is a methodology that recognizes gestures with the arm through the data it receives from the gyroscope and accelerometer of an arm band using Convolutional Neural Network and Recurrent Neural Network. In this process, the input data obtained from the arm band are entered in the Convolutional Neural Network to extract the characteristics and then in the layers of the Recurrent Neural Network to improve the performance of gesture recognition, which has improved the precision of each class by 6%.

### III. PROPOSED WORK

The aim of our work is to achieve a hand gesture detection model that allows developing solutions to control devices (such as a graphical user interface, virtual keyboard or mouse) in a natural and intuitive way. In the following we will detail the steps we follow in order to approach the solution. The steps we will detail below are the following: obtaining a point cloud of the hand, choosing the gestures to be used, creating the dataset, normalizing the data, defining the network architecture, training and obtaining the model for the predictions.

#### A. Inference of KeyPoints

This work implements the model *Mediapipe hands* [6] created with automatic learning techniques to infer 21 three-dimensional reference points of a hand from the processing of a single image. These 21 points (from now on KeyPoints) are located: one on the wrist and 4 more points on each finger (as shown in Fig. 2).

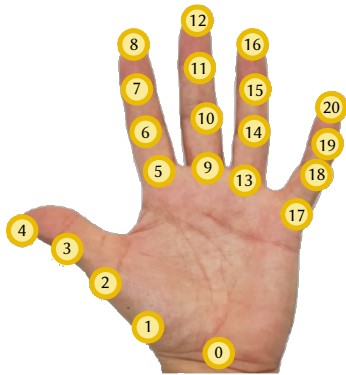


Fig. 2. KeyPoints of a hand.

#### B. Gesture Selection

In order to make a selection of gestures that users can choose from, we have explored gestures from non-verbal communication, sign language and related articles to Human-Computer Interaction. A variety of hand gestures are used in natural user interfaces and it is common to find solutions that use: the tip of the index finger or the open palm of the hand to control the mouse; the closed hand (fist) followed by the open hand to drag & drop; the thumb up to accept or the thumb down to cancel.

We want to obtain a model that recognizes a set of gestures to be able to design solutions in the future where users can select one by one the gestures for different actions (such as clicking the mouse, scrolling, moving the mouse pointer, moving forward or backward in a presentation, accepting or canceling). Just by obtaining an identifier for each gesture, either a letter or a number, we explored different sign languages and selected the following (visualized in the Fig. 3):

- From International Sign language: 1, 4, 5
- From American Sign Language: 9, V, W
- From French Sign Language: A, L, S



Fig. 3. Gesture names: S - 1 - V - W - 4 - 5 - 9 - L - A.

#### C. Dataset

In order to create our dataset, we requested video recordings from 10 volunteers. Each of them recorded a single video of approximately 3 minutes performing the 9 gestures without interrupting the recording. All movements were executed under free style, speed and direction to the personal liking. Subsequently, all the videos were processed in order to extract a sequence of grouped and annotated images for each gesture. A total of 39,150 images were obtained in a balanced way between gestures and volunteers. The *Mediapipe Hands* [6] model was used to extract the 21 keypoints of the hands from the complete set of images. A couple of sample images of this dataset with its detected KeyPoints are shown in Fig. 4.

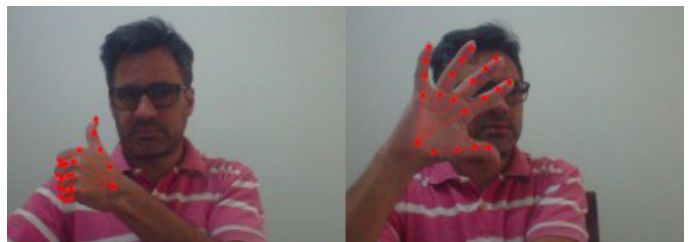


Fig. 4. Example of images to extract KeyPoints.

The dataset consists of a CSV (comma-separated values) file containing 39,150 records (4,350 for each of the 9 gestures) with the information shown in Fig. 5. Each record has 64 columns of information: the name of the gesture plus 21 KeyPoints (x, y, z).

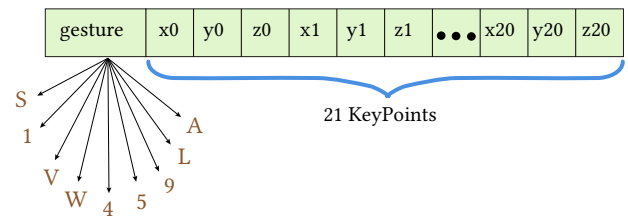


Fig. 5. Stored information.

This dataset is divided into a proportion of 80% for training and validation data (31,320 samples), and 20% for testing (7,830 samples).

In addition, we have downloaded 3 external datasets [27], [28] and [29] to test our model. Since these datasets do not contain our 9 gestures, we have combined them to reach a set of 4,500 samples (500 for each gesture).

#### D. Data Normalization

After generating the dataset of 39,150 images, data normalization is performed, which consists of several transformations (translation, rotation and scaling) so that KeyPoints are located at the origin of three-dimensional space and the middle finger aligned with Y-axis.

Following transformation matrices are used for normalization:

- Matrix (1) to translate to origin.

$$T = \begin{bmatrix} 1 & 0 & 0 & -kp0x \\ 0 & 1 & 0 & -kp0y \\ 0 & 0 & 1 & -kp0z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where KeyPoint 0 is  $(x, y, z) = (kp0x, kp0y, kp0z)$

- Rotation matrix (2) around an arbitrary axis: To align the middle finger with Y-axis.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

where:

$$\begin{aligned} r_{11} &= \cos \theta + u_x^2(1 - \cos \theta) \\ r_{12} &= u_x u_y(1 - \cos \theta) - u_z \sin \theta \\ r_{13} &= u_x u_z(1 - \cos \theta) + u_y \sin \theta \\ r_{21} &= u_y u_z(1 - \cos \theta) + u_x \sin \theta \\ r_{22} &= \cos \theta + u_y^2(1 - \cos \theta) \\ r_{23} &= u_y u_z(1 - \cos \theta) - u_x \sin \theta \\ r_{31} &= u_z u_x(1 - \cos \theta) - u_y \sin \theta \\ r_{32} &= u_z u_y(1 - \cos \theta) + u_x \sin \theta \\ r_{33} &= \cos \theta + u_z^2(1 - \cos \theta) \\ r_{33} &= \cos \theta + u_z^2(1 - \cos \theta) \end{aligned}$$

Here,  $u$  is a unit vector that is perpendicular to the plane formed by KeyPoint 9 vector and Y-axis. Knowing that two vectors are perpendicular (or orthogonal) when their dot product (or scalar product) is equal to zero, then we can calculate the vector  $u$ . Or it is also possible to use the vector product (or cross product) between KeyPoint 9 and Y-axis. To do this, we can use the Rule of Sarrus to calculate the  $3 \times 3$  determinant and thus obtain a vector perpendicular to the plane between KeyPoint 9 and Y-axis. Finally, we divide it by the norm to obtain a unit vector which is the vector  $u$  of the previous expressions.

By Rule of Sarrus we obtain a vector (Eq. 3) that, in general, is not unitary. We consider that the vector on the Y-axis is unitary, that is, it is the vector (0, 1, 0):

$$\begin{aligned} u'_x &= +(kp9y * 0 - 1 * kp9z) = -kp9z \\ u'_y &= -(kp9x * 0 - 0 * kp9z) = 0 \\ u'_z &= +(kp9x * 1 - 0 * kp9y) = +kp9x \end{aligned} \quad (3)$$

When we divide by its norm we get the unit vector  $u$ , as shown in Eq. 4.

$$\begin{aligned} |u'| &= \sqrt{(-kp9z)^2 + 0 + (kp9x)^2} \\ u &= \left( \frac{-kp9z}{|u'|}, 0, \frac{kp9x}{|u'|} \right) \end{aligned} \quad (4)$$

$\theta$  is the angle between the vector formed from the origin to the start of the middle finger (i.e. KeyPoint 9) and the unit vector on the Y-axis. It can be calculated as given in Eq. 5.

$$\theta = \cos^{-1} \left( \frac{kp9y}{\sqrt{(kp9x)^2 + (kp9y)^2 + (kp9z)^2}} \right) \quad (5)$$

- Matrix (6) to rotate palm on the Y-axis so it is aligned with plane  $z = 0$ .

$$R_y = \begin{bmatrix} \cos \beta & 0 & \sin \beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \beta & 0 & \cos \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$\beta$  is the angle on the plane  $y = 0$  of the angle formed between KeyPoint 17 and X-axis. In this way, we align the palm with the plane  $z = 0$  as shown in Eq. 7.

$$\beta = \tan^{-1} \left( \frac{kp17z}{kp17x} \right) \quad (7)$$

- Rotation matrix on Y-axis to place the palm in a frontal way: we use the  $R_y$  matrix to rotate  $180^\circ$  over Y-axis as long as the palm is

in the direction of the negative values of  $z$ . To know if the palm is facing forward or not, a simple calculation is done by detecting where fingertips are facing.

- Mirror with respect to the plane  $x = 0$ : Regardless of whether it is right or left hand, we want to mirror the hand in such a way that the thumb always remains towards positive values of  $x$ . It is easy to detect if the thumb is to the right or to the left by finding out  $x$  values of the KeyPoints belonging to the thumb.
- Scaling: The hand is scaled in order that the magnitude  $|kp0y - kp9y|$  is equal to 100. The matrix (8) is used to solve it.

$$E = \begin{bmatrix} \frac{100}{kp9y} & 0 & 0 & 0 \\ 0 & \frac{100}{kp9y} & 0 & 0 \\ 0 & 0 & \frac{100}{kp9y} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

To obtain normalized values, operations (shown in Eq. 9) are performed with these matrices with each of the 21 KeyPoints of each hand.

$$\begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix} = ER_yRT \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (9)$$

where  $x_n$ ,  $y_n$  and  $z_n$  are the coordinates of the normalized KeyPoints. Bear in mind that, depending on the case, the  $180^\circ$  rotation and/or the mirror with respect to the plane  $x = 0$  is also carried out.

To carry out normalization of the KeyPoints, we developed a tool that allows visualizing the correct normalization of KeyPoints that make up our dataset. In Fig. 6, a hand is shown in its original position and in Fig. 7 it is displayed after normalization. Some KeyPoints were joined with lines for a clear visualization of the hand's skeleton.

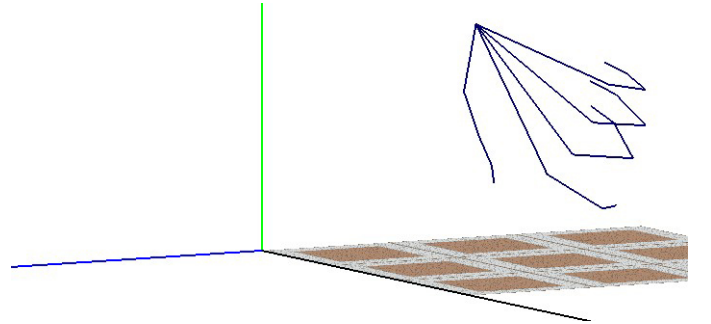


Fig. 6. Skeleton of a hand in its original position.

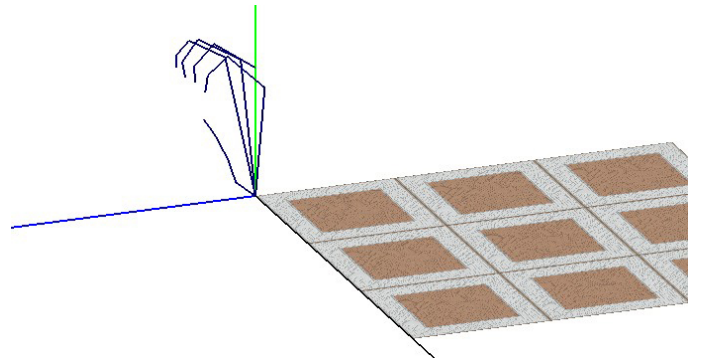


Fig. 7. Skeleton of a hand after normalization.



### E. PointNet Network Architecture

There is a type of neural network called PointNet [30], which receives in its input layer a point cloud for object classification. In general, point clouds are obtained from objects in an environment, and the challenge is to be able to classify and/or segment each one of them from this point cloud, which is just a bunch of isolated points that vaguely describe the structures and surfaces of the objects. The following is a brief discussion of some characteristics when classifying a point cloud:

- Invariance to permutation: a point cloud is a set of raw data, without additional information. It is a collection of  $(x, y, z)$  coordinates without structure. This makes the data invariant to permutations.
- Invariance to transformations: the classification of objects should not change if the point cloud undergoes translation and/or rotation transformations (not so with scaling).
- Importance between neighboring points: each point is not treated independently as the interaction between neighboring points contains useful information.

It is important to note that it is common to consider that a point cloud has a large number of points. The PointNet authors used in their work a cloud of 2048 points for each object, using the ModelNet10 dataset [14], which contains objects belonging to 10 classes.

PointNet network architecture for classification of a point cloud can be visualized and analyzed in [30]. This network takes  $n$  entry points, each one with dimension 3 belonging to  $(x, y, z)$  coordinates. The authors propose 2048 points for each object, so it would have an input with dimension [2048, 3]. It has two groups of layer called T-Net which are also neural networks that perform transformations on the data without modifying its dimension. These T-Net subnets are composed of temporal convolutions (Conv1D) with ReLU activation, batch normalization, 1D Max Pooling and densely connected layers (Fully Connected).

After transformations with T-Net combined with the convolution layers, a Max Pooling (GlobalMaxPooling1D) is performed, taking the global maximum value of the data, decreasing the dimensionality. It is followed by Fully Connected Layers, Dropout layers and a last layer with softmax activation function to obtain the scores for  $k$  output classes. PointNet network uses optimization with Adam stochastic gradient descent method and cross entropy as loss function. We analyze this network architecture and propose some modifications which are discussed below.

### F. Modified Network Architecture

T-Net subnets perform affine transformations in data and we propose to eliminate them, since our dataset already has different transformations that apply a normalization. We also propose to include new convolution layers and Fully Connected Layers, leaving an architecture as shown in Fig. 8, which was one of the best results. Note that the data normalization explained above is carried out beforehand.

### G. Data Increment

While analyzing a graph of 21 KeyPoints of a hand it can be difficult, to the human eye, to identify to which gesture those points correspond. It can be considered that 21 KeyPoints are insufficient to represent the skeleton of a hand, so we can generate extra data by knowing that among certain KeyPoints there is a hand bone (in the palm the metacarpal bones, in the beginning of the fingers the proximal phalanges, followed by the middle phalanges and at the tip of the fingers the distal phalanges).

Layer	Output Shape	Param #
InputLayer	(None, 21, 3)	0
Conv1D	(None, 21, 32)	128
BatchNormalization	(None, 21, 32)	128
Activation	(None, 21, 32)	0
Conv1D	(None, 21, 64)	2112
BatchNormalization	(None, 21, 64)	256
Activation	(None, 21, 64)	0
GlobalMaxPooling1D	(None, 64)	0
Dense	(None, 128)	8320
BatchNormalization	(None, 128)	512
Activation	(None, 128)	0
Dropout	(None, 128)	0
Dense	(None, 7)	903
Total params: 12,359		
Trainable params: 11,911		
Non-trainable params: 448		

Fig. 8. Proposed network architecture.

In this regard, a new parameter is defined which allows to incorporate a certain amount of additional KeyPoints on the bones of the hand. This is achieved by calculating lines that join the KeyPoints that correspond to the ends of the bones mentioned above. In Fig. 9 we can see KeyPoints of a hand with the addition of 10 KeyPoints on each bone.

### H. Training

At this point we have the network architecture defined with the Keras library and the dataset with 31,320 samples for training and validation. We continue with the training in order to obtain a model (in HDF5 format) that allows us to make predictions.



Fig. 9. Hand with 10 extra KeyPoints on each bone.

Keep in mind that we have a total of 39,150 samples in our own dataset plus 4,500 samples obtained from third-party image datasets. From the 39,150 samples, we separated 31,320 for training and validation, and 7,830 for testing. Note that we have two sets of samples for testing, one of which was randomly sampled from our own dataset and the other has been generated from third-party images.

## IV. EXPERIMENTS AND EVALUATION




































### A. Performance of the Proposed Network

Several network trainings were performed modifying parameters such as the number of epochs, learning rate of the Adam optimization method and the number of extra KeyPoints on each bone. In Table I are

TABLE I. 7,830 PREDICTIONS MADE

#	Extra KeyPoints on each bone	Learning rate	Epochs	Total KeyPoints	training loss	training acc	validation loss	validation acc	Correct predictions (ext dataset)	Correct predictions (%) (ext dataset)	Correct predictions (own dataset)	Correct predictions (%) (own dataset)
1	0	0.0005	10	21	0.0068	99.84 %	0.0048	99.95 %	4336 of 4500	96.36 %	7820 of 7830	99.87 %
2	2	0.0005	30	61	0.0085	99.80 %	0.0056	99.89 %	4305 of 4500	95.67 %	7820 of 7830	99.87 %
3	10	0.001	50	221	0.0062	99.85 %	0.0094	99.89 %	4297 of 4500	95.49 %	7820 of 7830	99.87 %
4	0	0.0005	30	21	0.0055	99.88 %	0.0072	99.86 %	4305 of 4500	95.67 %	7819 of 7830	99.86 %
5	5	0.0005	30	121	0.0054	99.90 %	0.0038	99.92 %	4349 of 4500	96.64 %	7819 of 7830	99.86 %
6	0	0.001	30	21	0.0070	99.81 %	0.0027	99.92 %	4315 of 4500	95.89 %	7819 of 7830	99.86 %
7	5	0.001	20	121	0.0084	99.80 %	0.0035	99.92 %	4340 of 4500	96.44 %	7819 of 7830	99.86 %
8	5	0.001	30	121	0.0064	99.86 %	0.0039	99.94 %	4289 of 4500	95.31 %	7819 of 7830	99.86 %
9	10	0.001	30	221	0.0075	99.82 %	0.0073	99.90 %	4296 of 4500	95.47 %	7819 of 7830	99.86 %
10	0	0.0005	50	21	0.0058	99.85 %	0.0050	99.89 %	4320 of 4500	96.00 %	7818 of 7830	99.85 %

TABLE II. NUMBER OF WRONG PREDICTIONS BY MODEL 1

Said...	?	?	?	It was...	Number of wrong predictions
 L				 1	1
 L				 A	1
 5				 9	1
 V				 W	1
 4				 W	2
 1				 V	2
 S				 A	2

shown the results ordered according to successes in predictions made with the test dataset of KeyPoints belonging to 7,830 hand samples and, in addition, 4500 samples of external datasets. The table shows the following: the amount of additional keyPoints added on each bone; the learning rate of Adam optimizer; the amount of epochs for training with batch size of 32 with a division of 80%/20% for training/validation; the total of KeyPoints for each hand; the loss in the training set after all the epochs; the accuracy with the training set; the loss in validation set after all epochs; the accuracy in validation set; the success rate in predictions made with a test set of 4,500 samples of external datasets; and the success rate in predictions made with a test set of 7,830 own samples (independent of the train/val set). The time consumed to perform each prediction is 26 mil-liseconds on average on an Intel® Core™ i7-1165G7 Processor (without GPU).

### B. Metrics

In order to observe the performance of the proposed architecture we will resort to analysis of trained model number 1, 5 and 10 presented in Table I.

- **Model number 1:** In order to have a quick approximation to the performance of this model, let's analyze the confusion matrix in

TABLE III. METRICS FOR TRAINING NUMBER 1

gesture	precision	recall	f1-score	support
s	0.9977	1.0000	0.9989	870
1	0.9977	0.9989	0.9983	870
v	0.9988	0.9977	0.9983	870
w	1.0000	0.9966	0.9983	870
4	0.9977	1.0000	0.9989	870
5	0.9989	1.0000	0.9994	870
9	1.0000	0.9989	0.9994	870
L	0.9977	1.0000	0.9989	870
a	1.0000	0.9966	0.9983	870

*accuracy = 0.9987 for 7830 predictions (870 each class)*

Fig. 10. Each column represents the number of predictions made by this model for each of the 9 gestures, while rows represent the true gesture. For these predictions, the own test set composed of 7,830 samples (870 for each gesture) is used, which is independent of set used for training and validation.

s	870	0	0	0	0	0	0	0	0
1	0	869	0	0	0	0	0	1	0
v	0	2	868	0	0	0	0	0	0
w	0	0	1	867	2	0	0	0	0
4	0	0	0	0	870	0	0	0	0
5	0	0	0	0	0	870	0	0	0
9	0	0	0	0	0	1	869	0	0
L	0	0	0	0	0	0	0	870	0
a	2	0	0	0	0	0	0	1	867
	s	1	v	w	4	5	9	L	a

Fig. 10. Confusion matrix of model number 1.

By breaking down a little the information of the confusion matrix, we can observe the incorrect predictions in Table II where it is shown in the first column the gesture predicted by the model,

in the second column the true gesture and in the third column the number of times that there was confusion. In Table III are presented metrics that mean the following:

- **Precision:** It provides information about false positives, as shown in Eq. 10. It is the ratio between well classified positive cases and the total number of predictions made.

$$\text{precision} = \frac{TP}{TP + FP} \quad (10)$$

where:

TP is the number of true positives

FP is the number of false positives.

- **Recall:** It indicates the ratio of positive classes that the model has been able to predict correctly. To exemplify, if the ratio is too low it means that the model missed too many positives. Being FN the number of false negatives, recall is defined in Eq. 11.

$$\text{recall} = \frac{TP}{TP + FN} \quad (11)$$

- **F1-score:** It combines precision and recall in a single value and allows to compare the performance between several models. F1-score is defined in Eq. 12.

$$F1 - \text{score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (12)$$

- **Support:** Number of predictions made for each class.
- **Accuracy:** It measures the ratio of cases that the model has succeeded, considering all the classes.

From this information we can mention that the model has a precision of 100% for the 'W', '9' and 'A' gestures, which means that in none of the predictions made has resulted in the 'W', '9' or 'A' gesture when they were not. This can be verified in the column 'It was ...' of Table II in which the 'W', '9' and 'A' gestures do not appear.

On the other hand, in the column 'Said ...' of Table II the 'S', '4', '5' and 'L' gestures do not appear, which means that they have a 100% of recall. This means that all predictions made for the 'S', '4', '5' and 'L' gestures have been accurate without having incorrect predictions.

In Fig. 11, the training metrics for each epoch were recorded, including accuracy and loss for training and validation sets. It is observed a correct learning of network parameters with the set of training with the passage of the epochs and with the validation set is observed that after the epoch number 6 does not improve performance significantly. It is worth mentioning that in this model was used a learning rate of 0.0005 for the optimizer Adam and that no extra KeyPoints were added on the hands.

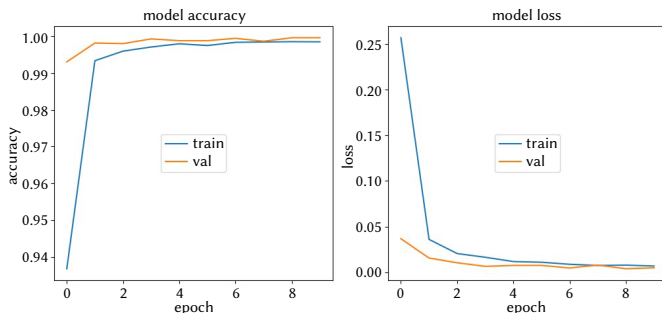


Fig. 11. Model accuracy and model loss for training number 1.

- **Model number 5:** For this model and in a comparative mode we will only analyze the metrics of Table IV and the graphs of Fig. 12. We can observe some minimal differences between precision and recall with respect to model number 1. However, we can use the f1-score metric to make a comparison with which we can indicate that the model number 5 has more erroneous predictions but is still very close to the performance of the previous model. Regarding the metrics during the learning process of the network, a similar behavior to the previous model is observed, where the performance does not improve considerably after the epoch number 10. For this model a learning rate of 0.0005 was used for Adam optimizer and 5 extra KeyPoints were added to each bone, making a total of 121 KeyPoints for each hand.

TABLE IV. METRICS FOR TRAINING NUMBER 5

gesture	precision	recall	f1-score	support
s	0.9977	1.0000	0.9989	870
1	0.9977	1.0000	0.9989	870
v	0.9988	0.9977	0.9983	870
w	1.0000	0.9954	0.9977	870
4	0.9966	0.9989	0.9977	870
5	0.9977	1.0000	0.9989	870
9	1.0000	1.0000	1.0000	870
L	0.9989	0.9989	0.9989	870
a	1.0000	0.9966	0.9983	870

accuracy = 0.9986 for 7830 predictions (870 each class)

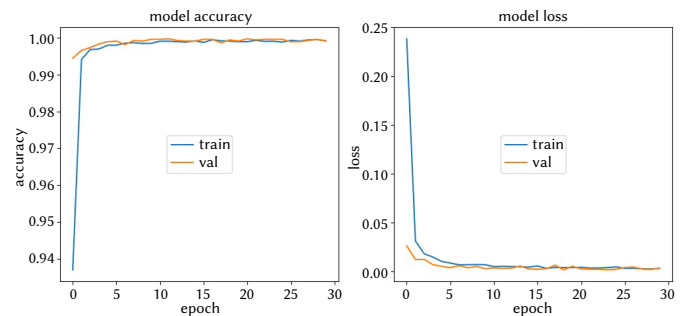


Fig. 12. Model accuracy and model loss for training number 5.

- **Model number 10:** In Table V a similar performance to the previous models is observed. No noticeable differences in the metrics during the learning (Fig. 13).

TABLE V. METRICS FOR TRAINING NUMBER 10

gesture	precision	recall	f1-score	support
s	0.9977	0.9989	0.9983	870
1	0.9977	1.0000	0.9989	870
v	0.9977	0.9977	0.9977	870
w	1.0000	0.9954	0.9977	870
4	0.9966	1.0000	0.9983	870
5	0.9989	0.9989	0.9989	870
9	1.0000	1.0000	1.0000	870
L	0.9989	0.9989	0.9989	870
a	0.9988	0.9966	0.9977	870

accuracy = 0.9985 for 7830 predictions (870 each class)

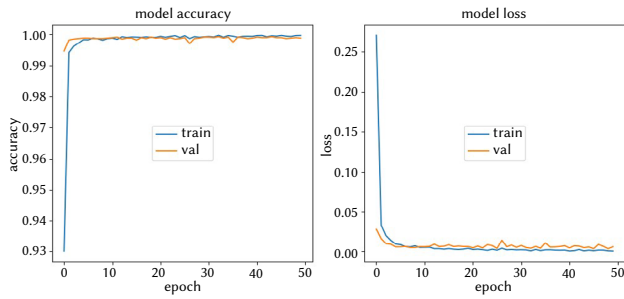


Fig. 13. Model accuracy and model loss for training number 10.

The results in Table I show a high performance of the proposed network. We detect the extra KeyPoints added on each bone would be of little importance, giving an indication that these extra KeyPoints do not provide significant information. We consider it would be important to include other type of information to the input data, such as the flexion angle at each joint and a number that identifies each KeyPoint. That is, if we look at Fig. 2 we can see that each KeyPoint has a number that identifies it and also on some KeyPoints is defined a flexion angle (except in the KeyPoints of the wrist and fingertips that do not have a defined angle). In this way, the input data could be defined as (  $x$ ,  $y$ ,  $z$ , number\_kp, angle\_joint ).

### C. Comparative Results

In order to compare the prediction accuracy of our model against other models, we have chosen our own test set (7,830 samples) and the external test set (4,500 samples). It is worth remembering that the own set is a random extraction of 20% of samples from our complete dataset (39,150 samples) and that the external test set is a collection of samples from third party works [27], [28], [29]. This set of external samples was made in order to obtain heterogeneous data, since they were extracted from images taken in other environments and by other people.

In addition to performing the predictions with our model (model number 1 in Table I) on the two test sets, we also use the PointNet model [13] trained with Adam optimizer with learning rate of 0.001 and 20 epochs, and also with a model created from ours, but without the initial transformation and normalization layer.

One can appreciate in these results the importance of the transformation and normalization layer that is initially applied. It provides a significant increase in accuracy when predictions are made with widely varying samples from different third party sources.

### D. Comparison With ROC and AUC

The Receiver Operating Characteristic (ROC) is a measure of a classifier's predictive quality that compares and visualizes the tradeoff between True Positive Rate ( $TPR = \frac{TP}{TP+FN}$ ) and False Positive Rate ( $FPR = \frac{FP}{FP+TN}$ ). ROC curves are typically used in binary classification, but one of the ways it can be approached is by binarizing the output (per-class). A ROC curve displays the true positive rate on the Y axis and the false positive rate on the X axis. The ideal region is therefore the top-left corner of the plot, where false positives are zero and true positives are one. This leads to Area Under the Curve (AUC), which is a metric that relates false positives and true positives. The higher the AUC, in general, the better the model.

Fig. 14 presents the ROC curve of our model number 1 (from Table I) and shows the high success rate achieved in the predictions. All three models predict considerably well with our dataset, as shown in Table VI, and the ROC curves are very similar to the one presented in Fig. 14. We present the ROC curves of the three models performing the predictions with the external dataset. It can be observed that only our model with the normalization and transformation layer behaves in an acceptable performance. This is shown in Fig. 15, 16 and 17.

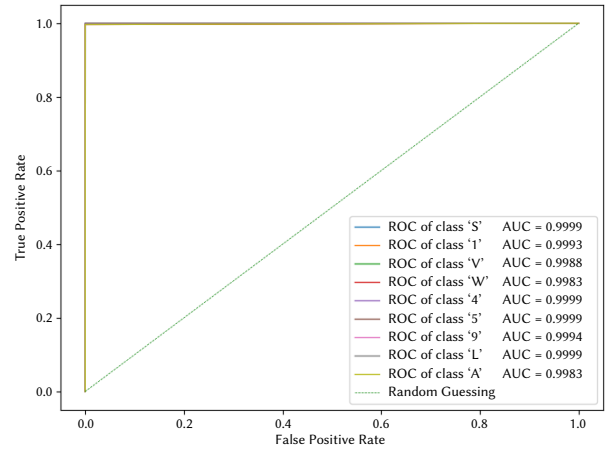


Fig. 14. ROC curves and AUC for our model with own dataset.

TABLE VI. COMPARISON OF MODELS

Model	Accuracy (our dataset)	Accuracy (external dataset)
Our model	7820 of 7830 99.87 %	4336 of 4500 96.36%
PointNet	7660 of 7830 97.82 %	2155 of 4500 47.89%
Our model without normalization	7760 of 7830 99.11 %	1371 of 4500 30.47 %

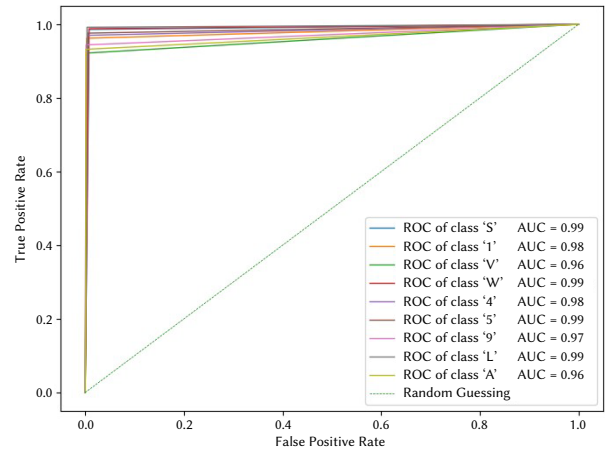


Fig. 15. ROC curves and AUC for our model with external dataset.

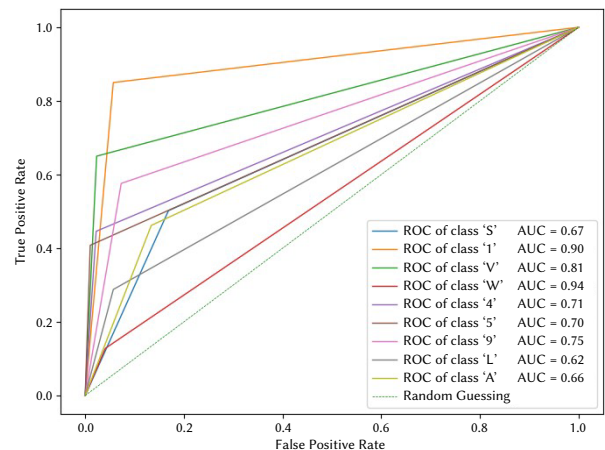


Fig. 16. ROC curves for PointNet model with external dataset.



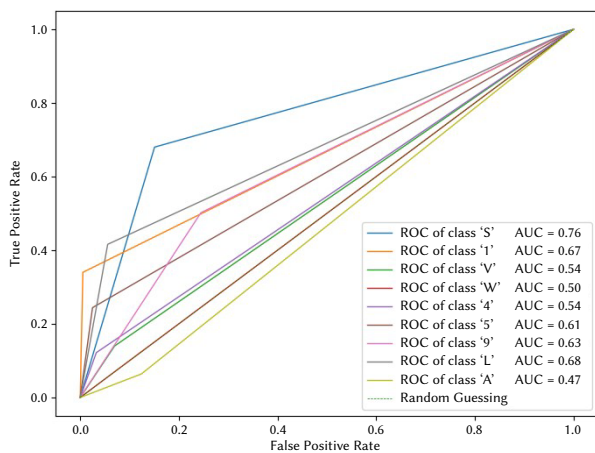


Fig. 17. ROC curves for our model without normalization.

## V. CONCLUSION

In this work, we present a new network architecture for hand gesture recognition using point cloud. The study was focused on the cloud of 3D reference points obtained through a standard RGB camera. The new network (based on PointNet architecture) was trained with hand KeyPoints and thanks to a simple architecture with few hidden layers it is possible to work directly on the CPU.

The results show an accuracy of 99.87% in our hand gesture dataset. It is interesting to extend this study by including new gestures in order to have a wider variety of options for device control, and also to experiment with end users to detect those gestures that are more appropriate to perform certain control commands.

It is important to notice that the transformation and normalization layer allows us to maintain the good prediction performance of our model by using third-party datasets that contain a wide variety of users and physical spaces where samples are taken.

## ACKNOWLEDGMENT

This work was funded by the EU ERDF and the Spanish Ministry of Economy and Competitiveness (MINECO) under AEI Project TIN2017-83964-R. <http://acg.ual.es/projects/cosmart/>

## REFERENCES

- [1] M. Palieri, B. Morrell, A. Thakur, K. Ebadi, J. Nash, A. Chatterjee, C. Kanellakis, L. Carlone, C. Guaragnella, A. a. Aghamohammadi, "Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time," *IEEE Robotics and Automation Letters*, pp. 1–1, 2020.
- [2] W. Zhang, D. Yang, "Lidar-based fast 3d stockpile modeling," in *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, 2019, pp. 703–707.
- [3] S. Muhammad, G. Kim, "Visual object detection based lidar point cloud classification," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2020, pp. 438–440.
- [4] C. P. Hsu, B. Li, B. Solano-Rivas, A. R. Gohil, P. H. Chan, A. D. Moore, V. Donzella, "A review and perspective on optical phased array for automotive lidar," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 27, no. 1, pp. 1–16, 2021.
- [5] E. de Oliveira, E. W. Gonzalez, D. G. Trevisan, L. C. de Castro Salgado, "Investigating users' natural engagement with a 3d design approach in an egocentric vision scenario," in *2020 22nd Symposium on Virtual and Augmented Reality (SVR)*, 2020, pp. 74–82.
- [6] F. Zhang, V. Bazarevsky, A. Vakunov, G. Sung, C.-L. Chang, M. Grundmann, A. Tkachenka, "Mediapipe hands: On-device real-time hand tracking," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, June 2020.
- [7] Y. Kartynnik, A. Ablavatski, I. Grishchenko, M. Grundmann, "Real-time facial surface geometry from monocular video on mobile gpus," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, June 2019.
- [8] V. Bazarevsky, I. Grishchenko, K. Raveendran, M. Grundmann, F. Zhang, T. Zhu, "Blazepose: On-device real-time body pose tracking," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, June 2020.
- [9] S. Shi, X. Wang, H. Li, "Pointcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 770–779.
- [10] Y. Zhou, O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.
- [11] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1513–1518, IEEE.
- [12] S. K. Arachchi, N. L. Hakim, H.-H. Hsu, S. V. Klimenko, T. K. Shih, "Real-time static and dynamic gesture recognition using mixed space features for 3d virtual world's interactions," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2018, pp. 627–632, IEEE.
- [13] H. Seo, S. Joo, "Influence of preprocessing and augmentation on 3d point cloud classification based on a deep neural network: Pointnet," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 895–899.
- [14] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.
- [15] Z. Li, W. Li, H. Liu, Y. Wang, G. Gui, "Optimized pointnet for 3d object classification," in *International Conference on Advanced Hybrid Information Processing*, 2019, pp. 271–278, Springer.
- [16] A. Jerjec, D. Bojanić, K. Bartol, T. Pribanić, T. Petković, S. Petrak, "On using pointnet architecture for human body segmentation," in *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, 2019, pp. 253–257, IEEE.
- [17] L. Ge, Y. Cai, J. Weng, J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8417–8426.
- [18] Y. Yang, C. Feng, Y. Shen, D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215.
- [19] Y. Yu, F. Li, Y. Zheng, M. Han, X. Le, "Clustering-enhanced pointcnn for point cloud classification learning," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–6.
- [20] C. R. Qi, L. Yi, H. Su, L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5099–5108, Curran Associates, Inc.
- [21] Y. Momma, W. Wang, E. Simo-Serra, S. Iizuka, R. Nakamura, H. Ishikawa, "P2net: A post-processing network for refining semantic segmentation of lidar point cloud based on consistency of consecutive frames," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 4110–4115.
- [22] D. Kulon, R. A. Guler, I. Kokkinos, M. M. Bronstein, S. Zafeiriou, "Weakly-supervised mesh-convolutional hand reconstruction in the wild," in *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, June 2020.
- [23] K. K. Verma, B. M. Singh, H. Mandoria, P. Chauhan, "Two-stage human activity recognition using 2d-convnet," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 6, no. 2, 2020.
- [24] M. Khari, A. K. Garg, R. G. Crespo, E. Verdú, "Gesture recognition of rgb and rgb-d static images using convolutional neural networks," *International Journal of Interactive Multimedia & Artificial Intelligence*, vol. 5, no. 7, 2019.
- [25] M. Kim, J. Cho, S. Lee, Y. Jung, "Imu sensor-based hand gesture recognition

for human-machine interfaces,” *Sensors*, vol. 19, no. 18, p. 3827, 2019.

- [26] J.-H. Kim, G.-S. Hong, B.-G. Kim, D. P. Dogra, “deepgesture: Deep learning-based gesture recognition scheme using motion sensors,” *Displays*, vol. 55, pp. 38–45, 2018.
- [27] A. Thakur, “American Sign Language Dataset for Image Classification.” <https://www.kaggle.com/ayuraj/asl-dataset>, 2019. [Online; accessed 2-July-2021].
- [28] A. Memo, L. Minto, P. Zanuttigh, “Exploiting Silhouette Descriptors and Synthetic Data for Hand Gesture Recognition.” <https://lstm.dei.unipd.it/downloads/gesture/>, 2015. [Online; accessed 2-July-2021].
- [29] P. Bao, A. I. Maqueda, C. R. del Blanco, N. García, “Image database for tiny hand gesture recognition.” <https://sites.google.com/view/handgesturedb/home>, 2017. [Online; accessed 2-July-2021].
- [30] C. R. Qi, H. Su, K. Mo, L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.



César Osimani

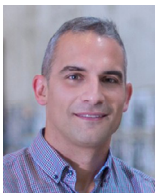
César Osimani is an Associate Professor in the Applied Research & Development Center on IT at Universidad Blas Pascal, Argentina. He received the degree in Telecommunications Engineering and he is currently a PhD student at Universidad Nacional de Córdoba. He is engaged in the research on computer vision, pattern recognition, augmented reality and Human-Computer Interaction.



Juan Jesus Ojeda-Castelo

Computer Science Engineering at University of Almeria. Juan Jesus got a Master degree in Systems and Languages Programming about Computer Science by UNED and he is currently a Ph.D. student at University of Almeria. He is a collaborator of the Project titled Investigar en el uso didactico de la Kinect en el CEEE Princesa Sofia Almeria which is supported by Junta de Andalucía. He is very

interested in Human-Computer Interaction particularly Natural interaction, Computer Vision and Artificial Intelligence especially Deep Learning. The devices that he usually attempts to include in his personal projects are: Microsoft Kinect, Leap Motion, Intel RealSense, so far.



Jose Antonio Piedra-Fernandez

He received his PhD in Computer Science from the University of Almeria in 2005. Currently, he is an Assistant Professor at the Department of Informatics, the same University. He is member of the Research Applied Computing Group (TIC-211), Coordinator of the Master in Computer Engineering since 2014 and Director of the Quality Secretariat since 2017. He works closely with the

James Wang’s research group at The Pennsylvania State University. José Luis Labrandero Prize by the Spanish Association of Remote Sensing in 2007. He got a Patent in the field of recognition of cancer cells using a fuzzy robot vision system. Participates in various national, international and regional projects. He is author and co-author in several scientific publications, among journal articles, national and international book chapters, and publications in national and international congress proceedings. He is reviewer of scientific articles in several international JCR impact journals, such as IEEE Transactions on Geoscience and Remote Sensing or International Journal of Remote Sensing. His research area focuses on computer vision, artificial intelligence, natural interaction systems and serious games applied to the field of education and health.