Hand gesture recognition with jointly calibrated Leap Motion and depth sensor

Giulio Marin, Fabio Dominio and Pietro Zanuttigh
Department of Information Engineering
University of Padova, Italy

***Abstract*—Novel 3D acquisition devices like depth cameras and the Leap Motion have recently reached the market. Depth cameras allow to obtain a complete 3D description of the framed scene while the Leap Motion sensor is a device explicitly targeted for hand gesture recognition and provides only a limited set of relevant points. This paper shows how to jointly exploit the two type of sensors for accurate gesture recognition. An ad-hoc solution for the joint calibration of the two devices is firstly presented. Then a set of novel feature descriptors is introduced both for the Leap Motion and for depth data. Various schemes based on the distances of the hand samples from the centroid, on the curvature of the hand contour and on the convex hull of the hand shape are employed and the use of Leap Motion data to aid feature extraction is also considered. The proposed feature sets are fed to a multi-class SVM classifier and experimental results show that a very high accuracy can be obtained from the proposed method. The current implementation is also able to run in real-time.**

***Index Terms*—Depth, Gesture Recognition, Calibration, Kinect , Leap Motion, SVM.**

I. Introduction

Automatic hand gesture recognition is a very intriguing problem that, if efficiently solved, could open the way to many applications in several different fields, e.g. human-computer interaction, computer gaming, robotics, automatic sign-language interpretation. The problem can be solved both by using wearable devices and with vision-based approach. Vision-based hand gesture recognition [1] is less invasive and paves the way for a more natural interaction, however it is also a very challenging problem.

Until a few years ago all the available approaches were based on the extraction of the information from images and videos [2]. These representations contain a 2D description of the three-dimensional hand pose, making it often difficult to properly understand the hand pose and consequently the performed gesture due to the complex 3D movements that the hand and fingers can do and to the presence of many inter-occlusions between the various hand parts.

The introduction of Time-Of-Flight cameras and of low cost consumer depth cameras based on structured light [3] has made 3D data acquisition available to the mass market, thus opening the way to a new family of computer vision methods that exploit 3D information to recognize the performed gestures. In particular the success of Microsoft's Kinect™ has shown how natural interfaces based on the acquisition of 3D data can be efficiently employed in commercial applications. However, notice how the standard usage of this device allows to recognize the whole body gestures but not the small details associated to the pose of the fingers. In order to exploit the data of the Kinect and of similar devices for hand gesture recognition, several methods have been proposed. The basic idea behind most of them is to extract relevant features from the depth data and then applying machine-learning techniques to the extracted features, in Section II an overview of the various available approaches will be presented.

The Leap Motion device is another recently introduced sensor based on vision techniques targeted to the extraction of 3D data, but differently from the Kinect that provides a 3D description of the framed scene, this device is explicitly targeted to hand gesture recognition and directly computes the position of the fingertips and the hand orientation. Compared with depth cameras like the Kinect, it produces a far more limited amount of information (only a few keypoints instead of the complete depth description) and works on a smaller 3D region. On the other side the extracted data is more accurate (according to a recent study [4] its accuracy is of about $200\mu m$) and it is not necessary to use computer vision algorithms to extract the relevant points since they are directly provided by the device. The software provided with the Leap Motion recognizes a few movement patterns only, e.g., swipe or tap, and the exploitation of Leap Motion data for more complex gesture recognition systems is still an almost unexplored field.

Since the Kinect and the Leap Motion have quite complementary characteristics (e.g., a few accurate and relevant keypoints against a large number of less accurate 3D points), it seems reasonable to exploit them together for gesture recognition purposes. If the information carried out by the two devices has to be jointly considered, a calibration of the whole system is needed. This paper, following this rationale, presents a novel approach for the combined use of the two devices for hand gesture recognition (Fig. 1 shows a general overview). Firstly ad-hoc approach for the joint calibration of the two devices is presented. Then reliable feature extraction schemes from both the Kinect and the Leap Motion data are introduced. The use of joint information from the two devices for more reliable and faster feature extraction will also be considered. Finally, a reliable classification scheme based on Support Vector Machines (SVM), suitable both for each of the two devices

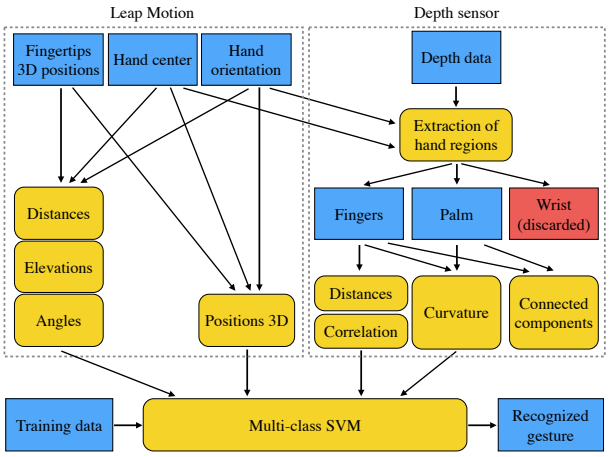


Fig. 1: Pipeline of the proposed approach.

alone and for the joint exploitation of the two sensors, is proposed. This work has several novel contributions: it presents the first attempt to detect gestures from the data acquired by the Leap Motion proposing reliable approaches for the feature extraction and for the gesture classification based on these features; it shows how to jointly calibrate the Leap Motion with depth cameras like the Kinect, a quite challenging task due to the limited amount of data provided by the Leap Motion; finally it shows how to jointly exploit the two devices for gesture recognition.

The paper is organized in the following way: Section II presents a brief overview of the related works, then Section III proposes a method for the joint calibration of the 3D measures from the two devices. Section IV present a novel set of feature descriptors that can be extracted from the Leap Motion data. Section V presents the feature extraction scheme from depth data and shows how to combine depth and Leap Motion data in order to reduce the computation time and improve the accuracy of the extracted features. Then the classifying algorithm is described in Section VI. Experimental results are presented in Section VII and finally Section VIII draws the conclusions.

II. Related Works

Hand gesture recognition from data acquired by the Kinect or other consumer depth cameras is a novel but very attractive research field. Many approaches have been presented, mostly based on the standard scheme of extracting relevant features from the depth data and then applying machine-learning techniques to the extracted features. In the approach of [5], silhouette and cell occupancy features are extracted from the depth data and used to build a shape descriptor. The descriptor is then used inside a classifier based on action graphs. Other approaches, e.g., [6] and [7] are based on volumetric shape descriptors. The two approaches both exploit a classifier based on Support Vector Machines (SVM). The histograms of the distance of hand edge points from the hand center are instead used in the approaches of [8] and [9]. Another approach based on an SVM classifier is [10], that employs 4 different types of features extracted from the depth data.

Other approaches instead estimate the complete 3D hand pose from depth data. Keskin et Al. [11] try to estimate the pose by segmenting the hand depth map into its different parts, with a variation of the machine learning approach used for full body tracking in [12]. Multi-view setups have also been used for this task [13], since approaches based on a single camera are affected by the large amount of occluded parts, making the pose estimation rather challenging.

Differently from the Kinect, the exploitation of Leap Motion data for gesture recognition systems is still an almost unexplored field. A preliminary study on the usage of this device for sign language recognition has been presented in [14]. Another gesture interface based on the Leap Motion has been presented in [15], where the authors use the device to control a robot arm.

III. Calibration

The employed acquisition setup, shown in Fig.2, consists of a depth sensor with optionally a color camera rigidly attached to the depth one (e.g., Time-Of-Flight cameras, Microsoft Kinect, Creative Senz3D, Asus Xtion PRO and other similar devices), and a Leap Motion device. Our implementation for testing the algorithm uses the Kinect sensor but the general pipeline remains valid also for the other devices. In particular, our approach does not require an additional color stream.

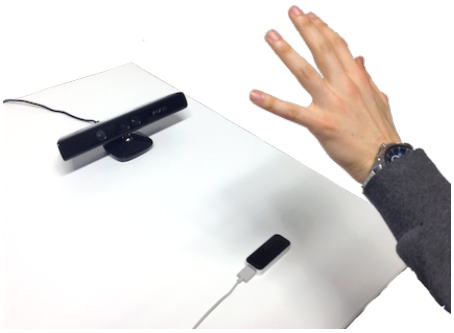


Fig. 2: Acquisition setup.

The aim of the calibration procedure is to estimate the extrinsic parameters of the two devices, i.e., the coordinate system transformation between the two devices, or equivalently the position of one sensor with respect to the other one. In addition, the two devices need also to be independently calibrated in order to correctly locate points in the 3D space. The Leap Motion software already provides a calibration tool, while the Kinect requires an external calibration, e.g., it is possible to use the approach of [16], in which both the color and the depth map from the sensor are used to extract intrinsic and extrinsic parameters. Our gesture recognition scheme requires only to associate to each point in the scene a depth value, therefore only

the projection matrix of the depth camera will be used. Given the two sensors independently calibrated, for every acquisition we get two sets of data describing the scene. The Leap Motion provides a point cloud with up to 6 points, one for the palm center and up to 5 for the fingertips. Data retrieved from the Kinect consist instead in a full frame depth map with an associated color image (the latter will not be used in the gesture recognition pipeline).

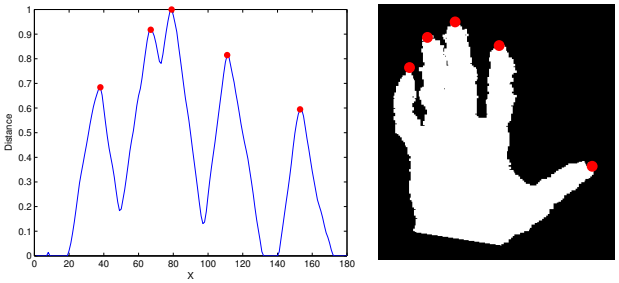
In order to find the roto-translation between the two sensors, the standard procedure requires to have the 3D coordinates of a set of points in the two coordinate systems. From the description of Leap Motion data (Section IV), it naturally follows that the only calibration clue that can be used is the hand itself. We decided to use the open hand gesture as the calibration tool (i.e., gesture G9 of the results database, see Fig. 8), this is because the Leap Motion software is not able to provide a one-to-one map between fingertips and real fingers, it just gives the positions in a random fashion; when 5 fingers are detected though we are quite sure that all the fingertips have been detected and with a few preprocessing they can be ordered and then associated to known fingers. The same points then need to be detected also from the depth camera, therefore a procedure for extracting the fingertips 3D coordinates will be presented.

As it is possible to see from Fig. 2, in order to be able to retrieve useful information from both the sensors, the Leap Motion has to be put under the performed gesture, while the depth sensor has been placed a little more forward, facing the user, as in a regular gesture recognition setup. The proposed calibration of a Leap Motion and a depth sensor, allows to easily make the two devices working together, without the need of external tools like checkerboards or other classic calibration devices. This is a key requirement for a human-computer interaction system, indeed, the proposed approach allows to easily set up a gesture recognition system exploiting the two devices, without the need of having them rigidly attached to a fixed structure. Whenever one of the two devices is moved, to calibrate the system it is sufficient to acquire a couple of frames of the user's open hand.

A. *Extraction of fingertips position*

Starting from the hand orientation and the palm center estimated from the Leap Motion, the palm plane can be extracted and the fingertips projected onto this plane. We decided to use the hand direction as a reference and then to associate to the thumb the fingertip with the most negative angle between the principal axis and the projected fingertip and to the other fingers the other fingertips by increasing angular values, until the fingertip with the greatest angular value associated to the pinkie. Section IV presents a description of the data acquired from the sensor and in particular provides more details on the angles computation. After this operation we obtain a set of 5 points $\mathbf{X}_L = X_L^1, ..., X_L^5$ describing the fingertips in the Leap Motion coordinate system.

For the depth sensor, instead, a more complex approach is required to extract fingertips positions from the acquired depth image. In order for the calibration process to be completely automatic, we decided to avoid the need to manually select points, relying instead in an automatic fingertips extraction algorithm. The idea is to extract the hand region from the acquired depth stream and then to process the hand contour to detect fingertips. The extraction of the hand has been performed using the approach of [10], while for the hand contour we exploit the distance feature as explained in Section V. The distance d of each point $\mathbf{X}$ of the hand contour from the palm center is computed, thus obtaining function $d(\mathbf{X})$. The fingertips are then assumed to be the points of the fingers at the maximum distance from the center. Given the function $d(\mathbf{X})$, its local maxima are the points $\bar{\mathbf{X}}$ where $f'(\bar{\mathbf{X}}) = 0$ and $f''(\bar{\mathbf{X}}) < 0$. Due to the inaccuracy in the depth image, the hand contour is usually irregular and need to be smoothed before searching for the local maxima. In addition, only the 5 highest maxima are used and a minimum distance between two candidates is guaranteed in order to avoid multiple detections on the same finger. Once these points are detected, the correspondent values in the depth image are selected and through the projection matrix of the depth camera they are back-projected in 3D space obtaining the 3D coordinates of the fingertips in the depth camera coordinate system $\mathbf{X}_D = X_D^1, ..., X_D^5$. Figure 3 shows an example of function $d(\mathbf{X})$, of the detected local maxima and of the relative fingertips in the depth image. It is worth notice that the Leap Motion API does not specify which actual point of the finger shape is returned as the fingertip, therefore we decided to consider as fingertip the farthest point of the finger.



a) b)

Fig. 3: Hand contour and detected fingertips: a) distance of each point of the hand contour, the red circles are the detected local maxima; b) projected local maxima onto the hand mask of the depth image.

B. *Roto-translation estimation*

With the two sets of fingertip 3D positions $\mathbf{X}_L$ from the Leap Motion and $\mathbf{X}_D$ from the depth camera, the goal is

to find the best roto-translation $\mathbf{R}$ and $\mathbf{t}$ that solves the following registration problem

$$\min ||\mathbf{R}\mathbf{X}_L + \mathbf{t} - \mathbf{X}_D||_F^2 \quad (1)$$

i.e., to find the best roto-translation that brings the point cloud $\mathbf{X}_L$ to the point cloud $\mathbf{X}_D$. In order to be more robust against noise, we acquire a couple of frames, extract the fingers and than compute the best parameters using a the RANSAC robust estimation approach. From our tests we found out that the assumption of considering as fingertip the extreme point of the finger is quite a valid assumption and that the mean error obtained from the square root of (1) for all the tested people is about 9 mm.

IV. Features extraction from the Leap Motion data

As already reported the Leap Motion device provides only a limited set of relevant points and not a complete description of the hand shape. From one side the amount of information is more limited if compared with depth sensors like the Kinect, but from the other side the device provide directly some of the most relevant points for gesture recognition and allows to avoid complex computations needed for their extraction from depth and color data. The Leap Motion sensor mainly provides the following data:

- **Number of detected fingers** $N \in [0, 5]$ that the device is currently seeing.
- **Position of the fingertips** $\mathbf{F}_i$, $i = 1, ..., N$. Vectors $\mathbf{F}_i$ containing the 3D positions of each of the detected fingertips. The sensor however does not provide a mapping between the vectors $\mathbf{F}_i$ and the fingers.
- **Palm center** $\mathbf{C}$ that represent the 3D location roughly corresponding to the center of the palm region in the 3D space.
- **Hand orientation** consists on two unit vectors representing the hand orientation computed in the palm center $\mathbf{C}$. The first vector, denoted with $\mathbf{h}$, points from the palm center to the direction of the fingers, while the second, denoted with $\mathbf{n}$, is the normal to the plane that corresponds to the palm region pointing downward from the palm center.
- **Hand radius** r is a scalar value corresponding to the radius of a sphere that roughly fits the curvature of the hand (it is not too reliable and it is not used in the proposed approach).

Note that the accuracy is not the same for all the reported data vectors. The 3D positions of the fingertips are quite accurate: according to a recent research [4] the error is about 200 μm. This is a very good accuracy, specially if compared to the one of depth data acquired by the Kinect and from other similar devices. While the localization of the detected fingers is accurate, their detection is not too reliable. There are some situations in which the sensor is not able to recognize all the fingers. Fingers folded over the hand or hidden from the sensor viewpoint are not captured,

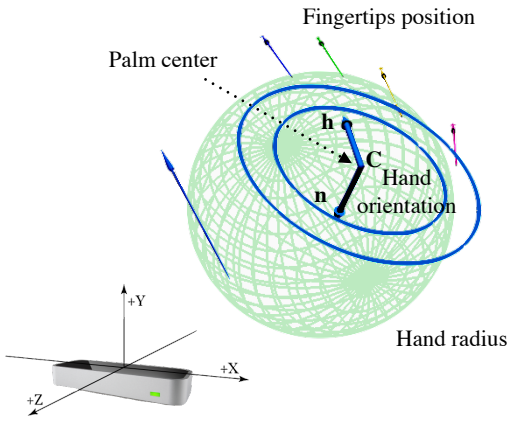


Fig. 4: Data acquired by the Leap Motion device.

furthermore fingers touching each other are sometimes detected as a single finger. Even in situations where the fingers are visible and separated from the hand and the other fingers it can happen that some fingers are lost, specially if the hand is not perpendicular to the camera. Another typical issue of this sensor is that protruding objects near the hand, like bracelets or sleeves edges, can be confused with fingers. These issues are quite critical and must be taken into account in developing a reliable gesture recognition approach since in different executions of the same gesture the number of captured fingers could vary. For this reason simple schemes based on the number of detected fingers have poor performance.

As previously stated then, Leap Motion does not provide a one-to-one map between fingers and fingertips detected, in the proposed approach we deal with this issue by sorting the features on the basis of the fingertip angle with respect to the hand direction $\mathbf{h}$. In order to account for the fingers misalignment, we consider the projection of the hand region into the palm plane described by $\mathbf{n}$ and passing through $\mathbf{C}$, as depicted in Fig. 5. The plane is then divided into five angular regions S_i, $i = 1, ..., 5$ and each captured finger is assigned to a specific region according to the angle between the projection of the finger in the plane and the hand direction $\mathbf{h}$. Note that a unique matching between the sectors and the fingers is not guaranteed, i.e., some of the sectors S_i could be associated to more than one finger and other sectors could be empty. When two fingers lie in the same angular region, one of the two is assigned to the nearest adjacent sector if not already occupied, otherwise the maximum between the two feature values is selected.

In this work we analyze 4 different types of features computed from the Leap Motion data that will be described in the rest of this section:

- **Fingertip angles**: angles corresponding to the orientation of each fingertip projected onto the palm plane with respect to the hand orientation $\mathbf{h}$.
- **Fingertip distances**: 3D distance of the fingertips from the hand center.

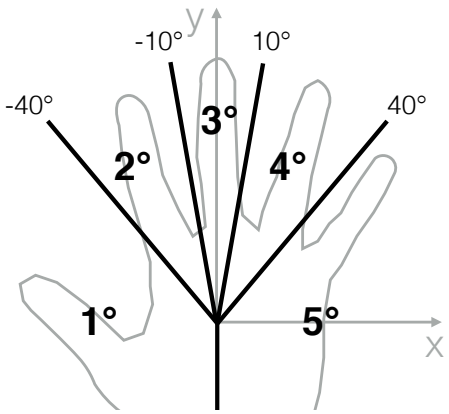


Fig. 5: Angular regions in the palm plane.

- **Fingertip elevations**: distances of the fingertip from the palm region plane.
- **Fingertip 3D positions**: x, y and z coordinates of the fingertips.

All the features values (except for the angles) are normalized in the interval $[0, 1]$ by dividing the values for the distance between the hand center and the middle fingertip length $S = ||\mathbf{F_{middle}} - \mathbf{C}||$ in order to make the approach robust to people with hands of different size. The scale factor S can be computed during the calibration of the system.

A. Fingertip angles

The computation of this feature plays a key role also for the other features since the angle is used as a metric to order the fingertips. The fingertip angle is defined as:

$$A_i = \angle(\mathbf{F}_i^{\pi} - \mathbf{C}, \mathbf{h}), i = 1, ..., N \quad (2)$$

where $\mathbf{F}_i^{\pi}$ is the projection of $\mathbf{F}_i$ on the plane identified by $\mathbf{n}$, and corresponds to the orientation of the projected fingertip with respect to the hand orientation. The estimated hand orientation $\mathbf{h}$ and consequently the fingertips angles are strongly affected by the number of detected fingers. In order to be scale independent, the obtained values A_i have been scaled and the interval have been set to $[0.5, 1]$, to better discriminate the valid values from the missing ones, that have been set to 0. These values have also been used to assign each finger to the corresponding sector. Fingertip angles features are then collected into vector $\mathbf{F}^a$.

B. Fingertip distances

This feature represent the distance of each fingertip from the palm center. Distance is defined as:

$$D_i = ||\mathbf{F}_i - \mathbf{C}||/S, i = 1, ..., N \quad (3)$$

and they are ordered according to increasing angles. At most one feature value is associated to each sector and the missing values has been set to 0. Fingertip distances are collected into vector $\mathbf{F}^d$.

C. Fingertip elevations

Another descriptor for a fingertip is its the elevation from the palm plane. Elevation is defined as:

$$E_i = \text{sgn}((\mathbf{F}_i - \mathbf{F}_i^{\pi}) \cdot \mathbf{n})||\mathbf{F}_i - \mathbf{F}_i^{\pi}||/S, i = 1, ..., N \quad (4)$$

and thanks to the sign operator it describes also to which of the two semi-spaces, defined by the palm plane, the fingertip belongs. As for the previous features, there is at most one feature value for each sector and the missing values has been set to 0. Note that as for the fingertips angles, the values range has been scaled to the interval $[0.5, 1]$ and then collected into vector $\mathbf{F}^e$.

D. Fingertip 3D position

This feature set represents the positions of the fingertips in the 3D space. As for the previous features, firstly the fingertips have been ordered according to increasing angles, then, since a reliable hand gesture recognition system must be independent from the hand position and orientation inside the frame, it is necessary to normalize the coordinates with respect to the hand position and orientation.

$$\begin{aligned} P_i^x &= (\mathbf{F}_i - \mathbf{C}) \cdot (\mathbf{n} \times \mathbf{h}) \\ P_i^y &= (\mathbf{F}_i - \mathbf{C}) \cdot \mathbf{h} \\ P_i^z &= (\mathbf{F}_i - \mathbf{C}) \cdot \mathbf{n} \end{aligned} \quad (5)$$

It is worth noticing that the fingertip 3D positions can be seen as the compact representation of the combination of angles, distances and elevation, i.e., of the first three features. Fingertip 3D positions have been collected into vector $\mathbf{F}^p$.

V. Features extraction from depth camera data

In the proposed approach, gestures are acquired with both a Leap Motion and a depth camera, we used a Kinect for testing the algorithm but any other depth camera can be used for this purpose. Feature extraction from depth data requires two main steps: firstly the hand is extracted from the rest of the scene using the acquired depth information, then, a set of features is computed from the segmented region.

The first step is quite time-consuming if solved by using only the depth and color data as we did in our previous works [10], [17]. In addition, most of the works available in the literature, dealing with hand extraction, assume that the hand is the closest object to the camera, an assumption that is often violated in a typical human-computer interaction domain, where there can be other objects in the scene closer to the camera. In the proposed approach the Leap Motion information is exploited in this first step both to improve the accuracy and to reduce the computation time. Using this information, the assumption that the hand is the closest object can be removed.

In the second step four different kinds of features are computed from the depth data:

- **Curvature features**: analyze the hand contour shape to extract the particular shape.
- **Distance features**: consider the distance of each point of the hand contour to describe the hand shape.
- **Correlation features**: this is a measure of similarity between distance features.
- **Connected components features**: exploiting the convex hull compute size and number of connected components in the performed gesture.

In the remaining section, firstly we will present our approach to segment out the hand using Leap Motion information, then the 4 different features are described.

A. Extraction of the hand from the combined use of depth and Leap Motion data

In our previous approach [10] the extraction of the hand from color and depth data was performed with a time-consuming procedure based on several steps. Firstly the closest point was localized on the depth data. Then a multiple thresholding on the depth values, on the distance from the closest point and on the color values with respect to the skin color was used to obtain a first estimate of the hand samples. The hand centroid was estimated in the subsequent step by finding the maximum of the output of a Gaussian filter with a large standard deviation applied to the estimated hand mask (this corresponds to assume that the densest region belongs to the hand palm). A circle is then fitted on the hand palm to precisely locate its center and to divide the hand into palm, wrist and fingers regions. Finally PCA is exploited to compute the hand orientation. The details of this approach can be found in [10], however it is clear that it is a quite complex operation, indeed, most of the computation time of the entire pipeline of [10] was spent on this step. Furthermore there are a couple of critical assumptions, i.e., that the closest point matching the skin color correspond to the hand and that the palm is the densest region, that can lead to wrong detections in particular situations. This typically does not happen in simple settings with a user is in front of the computer, but limits the applicability of the approach in more complex settings scenarios.

Since in the proposed approach the Leap Motion data is also available and the two devices have been jointly calibrated using the approach of Section III, it seems reasonable to exploit the data from this sensor in the detection step. The hand centroid computed by the Leap Motion $\mathbf{C}$ can be projected to the depth camera coordinate system obtaining the point $\mathbf{C}_D = \mathbf{R}\mathbf{C} + \mathbf{t}$ and used as a starting point for the hand detection. A sphere of radius r_h is then centered on $\mathbf{C}_D$ and the samples inside the sphere are selected, i.e:

$$\mathcal{H} = \{X : \|X - C_D\|^2 \leq r_h\} \tag{6}$$

where X is a generic 3D point acquired by the depth camera and r_h is set on the basis of the physical hand size (in the tests, $r_h = 10[cm]$ has been used). The points in the set $\mathcal{H}$ inside the sphere represent the initial hand estimate. This allows to remove the assumption that the hand is the closest point to the sensor. Furthermore, the thresholding in the color space can be avoided, as well as the acquisition and processing of color data, making this step faster and simpler. The centroid located by the Leap Motion is very reliably located in the hand region but its localization is not too accurate, due to the uncertainty in the position estimated from the Leap Motion. For this reason, its position is optimized with the circle fitting scheme of [10]. Let us denote with C_{palm} the final circle and with R its radius. A more refined scheme employing an ellipse in place of the circle can also be used [18].

The hand orientation can also be extracted from the Leap Motion data (it is given by the vectors $\mathbf{h}$ and $\mathbf{n}$ as discussed in Section IV), therefore the computation of the PCA can also be avoided. Another critical aspect in the approach of [10] is that with PCA the orientation was quite well estimated, but the direction was supposed always pointing upward, with the proposed approach instead this assumption can be removed, relying in the direction estimated by the Leap Motion.

Finally the hand samples are subdivided into fingers, palm and wrist regions. Palm samples ($\mathcal{P}$) are the ones inside $\mathbf{C}_{palm}$; the fingers samples set $\mathcal{F}$ contains the samples $\mathbf{X}$ outside $\mathbf{C}_{palm}$ that satisfy $(\mathbf{X} - \mathbf{C}_D) \cdot \mathbf{h} > R$, i.e., the ones outside the circle in the direction of $\mathbf{h}$; the remaining samples are associated to the wrist region ($\mathcal{W}$).

B. Distance features

This feature set aims at capturing the profile of the hand contour in order to extract informative description of the performed gesture. We start by considering each point $\mathbf{X}$ in the hand contour, extracted from the hand mask in the depth image, at each point, the distance $d(\mathbf{X})$ with respect to the hand center $\mathbf{C}_{palm}$ is computed

$$d(\mathbf{X}) = ||\mathbf{X} - \mathbf{C}_{palm}|| \tag{7}$$

Given the hand orientation then, we are able to provide a coherent function $d(\mathbf{X})$ among different gestures and repetitions, i.e., we can set as starting point $\mathbf{X_1}$ the intersection between the hand contour and the hand direction $\mathbf{h}$, and then proceed clockwise with the other points until the last one $\mathbf{X_n}$. For each acquisition though, the number of points in the hand contour n is not fixed, therefore in order to be consistent, the function $d(\mathbf{X})$ is sampled to get 180 values (this value can be chosen even smaller without excessively impacting the overall accuracy, but reducing the computation time). An example of this function is shown in Fig. 3 a).

The distance function d is then normalized by the length L_{max} of the middle finger in order to scale the values within the range $[0, 1]$ and to account for different hand sizes among people. The distance samples are collected into feature vector $\mathbf{F}^l$. Notice that this descriptor is different

from the distance descriptors used in [10]: the approach proposed in this work turned out to be simpler, faster and more accurate.

C. Correlation features

This feature set is based on the similarity between distance functions of subsection V-B. For each considered gesture, a reference acquisition is selected and the corresponding distance function is computed with the approach of Eq. 7, thus obtaining a set of reference function $d_g^r(\mathbf{X})$, where g is the considered gesture. The distance function of the acquired gesture $d(\mathbf{X})$ is also computed and the maximum of the correlation between the current histogram $d(\mathbf{X})$ and a shifted version of the reference histogram $d_g^r(\mathbf{X})$ is selected:

$$R_g = \max_{\Delta} \left[\rho\left(d(\mathbf{X}), d_g^r(\mathbf{X}+\Delta)\right), \rho\left(d(-\mathbf{X}), d_g^r(\mathbf{X}+\Delta)\right) \right] \tag{8}$$

where $g = 1, ..., G$ and $d(-\mathbf{X})$ is the flipped version of the distance function to account for the fact that the hand could have either the palm or the dorsum facing the camera. The computation is performed for each of the candidate gesture, thus obtaining a set $\mathbf{F}^{\rho}$ containing a different feature value f_g^{ρ} for each of them. As expected, ideally the correlation with the correct gesture should have a larger value than the other features.

D. Curvature features

This feature set describes the curvature of the hand edges on the depth map. A scheme based on on integral invariants [19], [20] has been used. The approach for the computation of this feature is basically the same of [10]. The main steps of the approach are here briefly recalled. The curvature feature extractor algorithm takes as input the edge points of the palm and fingers regions and the binary mask B_{hand} corresponding to the hand samples on the depth map. A set of circular masks with increasing radius is then built on each edge sample (for the results $S = 25$ masks with radius varying from $0.5cm$ to $5cm$ have been used, the radius correspond to the scale level at which the computation is performed).

The ratio between the number of samples falling in B_{hand} for each circular mask and the size of the mask is computed. The values of the ratios (i.e., curvatures) range from 0 (extremely convex shape) to 1 (extremely concave shape), with 0.5 corresponding to a straight edge. The $[0, 1]$ interval is quantized into N bins. Feature values $f_{b,s}^c$ collects how many edge samples have a curvature of a value inside bin b at scale level s. The values are finally normalized by the number of edge samples and the feature vector $\mathbf{F}^c$ with $B \times S$ entries is built. For faster processing, the circular masks can be replaced with simpler square masks and then integral images can be used for the computation. This approximation, even if not perfectly rotation invariant, is significantly faster and the performance loss is very small.

E. Connected components features

Another useful clue used for gesture recognition schemes [21] is the convex hull of the hand shape in the depth map. The idea is to compute the convex hull of the hand shape in the depth map and to look for regions within the convex hull region but not belonging to the hand. These typically correspond to the empty regions between the fingers and are a good clue to recognize the fingers arrangement. Let $S = C_{hull}(\mathcal{B}) - \mathcal{B}$ be the difference between the convex hull and the hand shape (see Fig. 6 a and b). Region S is made of a few connected components S_i. The size of each region S_i is compared with a threshold T_{cc} and the ones that are smaller than the threshold are discarded (this allows to avoid considering small components due to noise as it is possible on the right of the hand in Fig. 6 c). The output of this procedure is the set $\mathcal{S} = \{S_i : S_i > T_{cc}\}$ (Fig. 6 c and d).

The feature set is given by the ratios between the area of the each connected components and the convex hull area, i.e.:

$$f_i^{cc} = \frac{area(S_i | S_i \in \mathcal{S})}{area(C_{hull}(\mathcal{B})))} \tag{9}$$

where the areas have been sorted according to the angle of their centroid with respect to the hand direction (i.e., from the thumb to the little). These numbers are then collected into vector $\mathbf{F}^{cc}$.

Fig. 6: Areas of the connected components: a) and b): difference between the convex hull and the hand shape; c) and d) connected components in set $\mathcal{S}$ highlighted in green.

VI. Gesture classification

The approaches of Sections IV and V produce eight different feature vectors, four for the Leap Motion data and four for the depth data. Each vector describe some relevant clues regarding the performed gesture and in order to recognize it, a multi-class Support Vector Machine classifier is used. There are 8 feature vectors grouped into the two sets $\mathbf{V}_{leap} = [\mathbf{F^a}, \mathbf{F^d}, \mathbf{F^e}, \mathbf{F^p}]$ that contains all the features extracted from Leap Motion data and $\mathbf{V}_{kin} = [\mathbf{F^l}, \mathbf{F}^{\rho}, \mathbf{F^c}, \mathbf{F^{cc}}]$ that collect the features computed from depth information. Each vector can be used alone or together with any of the other descriptors. The combination of multiple features descriptors can be obtained by simply concatenating the features vectors corresponding to the selected features.

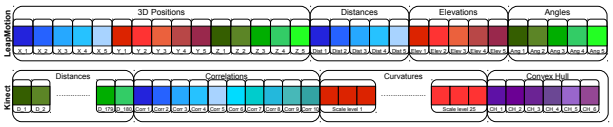


Fig. 7: Feature vectors extracted from the two devices.

The target of the approach is to classify the performed gestures into G classes, one for each gesture in the considered database. A multi-class SVM classifier [22] based on the *one-against-one* approach has been used. In the employed scheme a set of $G(G-1)/2$ binary SVM classifiers are used to test each class against each other. The output of each of them is chosen as a *vote* for a certain gesture. For each sample in the test set, the gesture with the maximum number of votes is selected as the output of the classification.

In particular a non-linear Gaussian Radial Basis Function (RBF) kernel has been selected and the classifier parameters have tuned exploiting grid search and cross-validation on the training set. Let us consider a training set containing data from M users. The space of parameters (C, γ) of the RBF kernel is divided with a regular grid. For each couple of parameters the training set is divided into two parts, one containing $M-1$ users for training and the other with the remaining user for validation and performance evaluation. The procedure is repeated M times changing the user used for the validation. The couple of parameters that give the best accuracy on average are selected as the output of the grid search. Finally the SVM has been trained on all the M users of the training set with the optimal parameters.

VII. Experimental results

The results have been obtained using the setup depicted in Fig. 2. A Leap Motion device and a Kinect have been used to jointly acquire the data relative to the performed gestures. The Kinect depth camera has been selected due to its large diffusion, however any other depth camera, e.g. Creative's Senz3D or the second generation Kinect can be used in the proposed approach. The two devices have been jointly calibrated using the approach of Section III and synchronized in time. A software synchronization has been used: its precision is sufficient for the recognition of gestures based on static poses like the ones considered in this paper, for gesture based on fast movements probably more accurate synchronization approaches would be needed. The considered dataset of gestures contains the 10 different gestures shown in Fig. 8. 14 different people have been used and each user has repeated each gesture 10 times for a total of 1400 different data samples. Up to our knowledge this is the first database containing both depth data and Leap Motion data and it will be made publicly available on the web in order to allow future comparisons with other approaches.

Let us start from the Leap Motion device. Table I shows the accuracy obtained using the classification algorithm of Section VI on the data from this sensor. The 3D positions of the fingertips give a very good representation of the arrangement of the fingers and allow to obtain an accuracy of 81.5%. They allow to recognize the majority of the gestures even if the recognition of some gestures is not always optimal, as it is possible to see from the confusion matrix in Fig. 9. For example, gestures G2 and G3 are sometimes confused with gesture G1.

Feature set	Accuracy
Fingertips 3D positions ($\mathbf{F^p}$)	81.5%
Fingertips distances ($\mathbf{F^d}$)	76.1%
Fingertips angles ($\mathbf{F^a}$)	74.2%
Fingertips elevations ($\mathbf{F^e}$)	73.1%
$\mathbf{F^d + F^a + F^e}$	80.9%

TABLE I: Performance with the Leap Motion data.

Fingertip distance features allow to obtain an accuracy of about 76%, they are also able to recognize most gestures but there are some critical issues, e.g. G2 and G3 are easily confused. A relevant issue for this descriptor is the limited accuracy of the hand direction estimation from the Leap Motion that does not allow a precise match between the fingertips and the corresponding angular regions (i.e., it is not easy to recognize which finger has been raised if a single finger is detected). The other two features have slightly lower performance. The angles allow to obtain an accuracy of 74.2% and a similar result (73%) can be obtained from the elevations alone. The last 3 features can be combined together since they capture different properties of the fingers arrangement. Their combination leads to an accuracy of almost 81%, better than any of the 3 features alone. This result is quite similar to the performance of the 3D positions, consistently with the fact that the two distances from the center and the plane, together with the angle can be viewed as a different representation of the position of the point in 3D space.

Results from the Leap Motion data are good but not completely satisfactory, better results can be obtained from the depth data, that offers a more informative description of the arrangement of the hand in 3D space. Notice that depth contains the complete 3D structure of the hand but it is also a lower-level scene description and a larger amount of processing is needed in order to extract the features from it.

Feature set	Accuracy
Distance features ($\mathbf{F^d}$)	94.4%
Correlations features ($\mathbf{F}^{\rho}$)	68.7%
Curvature features ($\mathbf{F^c}$)	86.2%
Convex Hull features($\mathbf{F^{cc}}$)	70.5%
$\mathbf{F^d + F^c}$	96.35%

TABLE II: Performance with the depth data.

Table II shows the results obtained from the depth in-

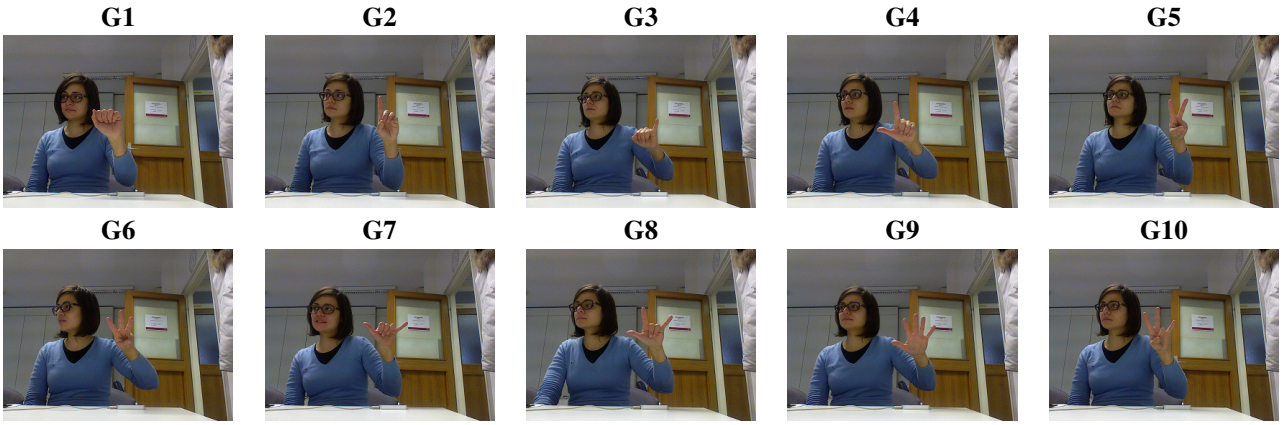


Fig. 8: Gestures from the American Sign Language (ASL) contained in the database that has been acquired for experimental results.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	0,89	0,02	0,06	0,02	0,00	0,00	0,00	0,00	0,00	0,00
G2	0,30	0,56	0,14	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G3	0,14	0,09	0,70	0,04	0,00	0,00	0,02	0,00	0,00	0,00
G4	0,03	0,00	0,00	0,90	0,00	0,00	0,05	0,00	0,01	0,01
G5	0,05	0,05	0,03	0,02	0,76	0,01	0,02	0,02	0,00	0,04
G6	0,01	0,00	0,03	0,00	0,03	0,84	0,01	0,01	0,00	0,07
G7	0,01	0,00	0,04	0,08	0,00	0,00	0,81	0,03	0,01	0,02
G8	0,00	0,00	0,00	0,04	0,03	0,00	0,03	0,83	0,00	0,08
G9	0,00	0,01	0,00	0,01	0,00	0,00	0,00	0,01	0,97	0,00
G10	0,00	0,00	0,01	0,00	0,04	0,01	0,05	0,01	0,00	0,89

Fig. 9: Confusion matrix for the 3D positions from the Leap Motion data. The larger errors have been highlighted.

formation acquired with a Kinect. Distance features are the best performing descriptor and allow to obtain an accuracy of 94.4%, much higher than the one that can be obtained from the Leap Motion sensor. These descriptor alone allow to recognize all the gestures with an high accuracy.

Correlation features have lower performance (68.7%). This descriptor is also based on the distances of the hand samples from the hand centroid, but compared to the distances they contain a less informative description (the feature vector size is also much smaller) that is not sufficient for an accurate recognition. However thanks to the small descriptor size and very fast computation of the descriptor they still can be considered for applications where the running time and the memory footprint of the descriptor are critical.

Another very good descriptor is the curvature of the hand contour. It allows a correct recognition of 86.2% of the considered gestures. Only distance features outperforms this descriptor. It has also the advantage that it do not rely on the computation of the hand center and orientation, making it very useful in situations where an estimation of these parameters is difficult. Finally, the convex hull features have an accuracy of 70.5%, slightly better than the correlations even if not too impressive. Again its small size and simple computation makes this descriptor interesting when a trade-off between performance and accuracy is needed.

The combination of multiple descriptors allows to improve the performance, e.g., by combining the two best performing descriptors, distances and curvatures a quite impressive accuracy of 96.35% can be obtained as it is possible to see also from the corresponding confusion matrix (Fig. 10). This is an indication that the different descriptors capture different properties of the hand arrangement and contain complementary information.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	0,97	0,02	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G2	0,01	0,97	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G3	0,01	0,00	0,99	0,00	0,01	0,00	0,00	0,00	0,00	0,00
G4	0,00	0,04	0,00	0,96	0,00	0,00	0,00	0,00	0,00	0,00
G5	0,00	0,01	0,00	0,00	0,99	0,01	0,00	0,00	0,00	0,00
G6	0,00	0,04	0,00	0,00	0,04	0,89	0,00	0,01	0,00	0,02
G7	0,00	0,00	0,01	0,00	0,00	0,00	0,99	0,00	0,00	0,00
G8	0,00	0,00	0,00	0,01	0,00	0,01	0,00	0,96	0,01	0,01
G9	0,00	0,00	0,00	0,01	0,00	0,01	0,00	0,00	0,99	0,00
G10	0,00	0,00	0,00	0,00	0,01	0,04	0,00	0,02	0,00	0,93

Fig. 10: Confusion matrix for the combined use of distance and curvature descriptors from depth data. The larger errors have been highlighted.

Feature set	Accuracy
$\mathbf{F^d + F^c + F^p}$	$\mathbf{96.5\%}$

TABLE III: Performance from the combined use of the two sensors.

Descriptors based on the Leap Motion data and on the depth data can also be combined together. In the last test we combined the 3D positions from the Leap Motion with the two best descriptors from depth data, i.e., the distances and the curvatures. The obtained accuracy is 96.5% as shown in Table III. The corresponding confusion matrix (Fig. 11) shows also how the recognition rate is very high for all the considered gestures. The improvement with respect to depth data alone is limited, as expected since the accuracy from the 3D positions of the Leap Motion is much lower, however a small improvement is present, showing that

some additional information content is present in the Leap Motion data.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	0,98	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G2	0,01	0,96	0,02	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G3	0,01	0,01	0,99	0,00	0,00	0,00	0,00	0,00	0,00	0,00
G4	0,00	0,03	0,00	0,97	0,00	0,00	0,00	0,00	0,00	0,00
G5	0,01	0,00	0,00	0,00	0,99	0,01	0,00	0,00	0,00	0,00
G6	0,00	0,03	0,00	0,00	0,04	0,89	0,00	0,01	0,00	0,04
G7	0,00	0,00	0,01	0,00	0,00	0,00	0,99	0,00	0,00	0,00
G8	0,00	0,00	0,00	0,01	0,00	0,01	0,00	0,96	0,01	0,01
G9	0,00	0,00	0,00	0,01	0,00	0,01	0,00	0,00	0,99	0,00
G10	0,00	0,00	0,00	0,00	0,01	0,03	0,00	0,01	0,00	0,95

Fig. 11: Confusion matrix for the combined use of Leap Motion and depth data. The larger errors have been highlighted.

Finally, notice how the proposed approach is particularly suitable for real time gesture recognition schemes. The current implementation in C++ (that has not been fully optimized) has been tested on a not too performing desktop PC with an Intel Q6600 processor and 4Gb of RAM and real-time performance have been obtained The initial hand detection phase, that took $46ms$ in the implementation of the approach of [10] and that we used to start the development of this work can now be completed in a few milliseconds thanks to the exploitation of the Leap Motion centroid. The extraction of palm and fingers regions with the circle fitting requires about $25ms$. The orientation of the hand is also directly computed from the Leap Motion data (this step took about $4ms$ in the old approach). Feature extraction is quite fast, the most demanding ones are curvature descriptors that take about 28 ms to be computed while the other features are way faster to be computed. Finally SVM classification is performed in just $1ms$. This allows to obtain a frame rate of about $15fps$ if depth data is used with respect to the $10fps$ achieved by the previous approach on the same computer. Gesture recognition with the Leap Motion data alone is very fast (just a few milliseconds) but performances are also lower.

VIII. Conclusions

In this paper an effective gesture recognition pipeline for the Leap Motion, for depth sensors and for their combined usage has been proposed. The different nature of data provided by the Leap Motion (i.e., a higher level but more limited data description) with respect to the depth cameras, poses challenging issues for which effective solutions have been presented. An ad-hoc calibration scheme has allowed to jointly calibrate the Leap Motion with depth sensors. The limited number of points computed by the first device makes this task quite challenging but the proposed scheme allows to obtain a good accuracy sufficient for the joint exploitation of the data from the two devices. Several different feature sets have been presented for both sensors. Four different types of features have been extracted from the Leap Motion while different types of descriptors have been computed from the depth data based on different clues likes the distances from the hand centroid, the curvature of the hand contour and the convex hull of the hand shape. It has also been shown how to exploit Leap Motion data to improve the computation time and accuracy of the depth features.

Experimental results have shown how the data provided by Leap Motion, even if not completely reliable, allows to obtain a reasonable overall accuracy with the proposed set of features and classification algorithm. A very good accuracy can be obtained from depth data that is a more complete description of the hand shape, in particular distance and curvature descriptors allow to obtain almost optimal performances.

Future work will address the the recognition of dynamic gestures with the proposed setup and improved schemes for the detection and localization of the fingertips jointly exploiting the data from the two sensors.

References

[1] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," *Commun. ACM*, vol. 54, no. 2, pp. 60–71, Feb. 2011.

[2] D. Kosmopoulos, A. Doulamis, and N. Doulamis, "Gesture-based video summarization," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. 3, 2005, pp. III–1220–3.

[3] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo, *Time-of-Flight Cameras and Microsoft Kinect*, ser. SpringerBriefs in Electrical and Computer Engineering. Springer, 2012.

[4] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[5] A. Kurakin, Z. Zhang, and Z. Liu, "A real-time system for dynamic hand gesture recognition with a depth sensor," in *Proc. of EUSIPCO*, 2012.

[6] P. Suryanarayan, A. Subramanian, and D. Mandalapu, "Dynamic hand pose recognition using depth data," in *Proc. of ICPR*, aug. 2010, pp. 3105 –3108.

[7] J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu, "Robust 3d action recognition with random occupancy patterns," in *Proc. of ECCV*, 2012.

[8] Z. Ren, J. Yuan, and Z. Zhang, "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *Proc. of ACM Conference on Multimedia*. ACM, 2011, pp. 1093–1096.

[9] Z. Ren, J. Meng, and J. Yuan, "Depth camera based hand gesture recognition and its applications in human-computer-interaction," in *Proc. of ICICS*, 2011, pp. 1 –5.

[10] F. Dominio, M. Donadeo, and P. Zanuttigh, "Combining multiple depth-based descriptors for hand gesture recognition," *Pattern Recognition Letters*, 2013.

[11] C. Keskin, F. Kirac, Y. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *ICCV Workshops*, nov. 2011, pp. 1228 –1234.

[12] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1297–1304.

[13] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys, "Motion capture of hands in action using discriminative salient points," in *Proc. of ECCV*, October 2012.

[14] L. E. Potter, J. Araullo, and L. Carter, "The leap motion controller: A view on sign language," in *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, ser. OzCHI '13. New York, NY, USA: ACM, 2013, pp. 175–178.

[15] C. Guerrero-Rincon, A. Uribe-Quevedo, H. Leon-Rodriguez, and J.-O. Park, "Hand-based tracking animatronics interaction," in *Robotics (ISR), 2013 44th International Symposium on*, 2013, pp. 1–3.

[16] D. Herrera, J. Kannala, and J. Heikkilä, "Joint depth and color camera calibration with distortion correction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 10, pp. 2058–2064, 2012.

[17] F. Dominio, M. Donadeo, G. Marin, P. Zanuttigh, and G. M. Cortelazzo, "Hand gesture recognition with depth data," in *Proceedings of the 4th ACM/IEEE international workshop on Analysis and retrieval of tracked events and motion in imagery stream*. ACM, 2013, pp. 9–16.

[18] G. Marin, M. Fraccaro, M. Donadeo, F. Dominio, and P. Zanuttigh, "Palm area detection for reliable hand gesture recognition," in *Proceedings of MMSP 2013*, 2013.

[19] S. Manay, D. Cremers, B.-W. Hong, A. Yezzi, and S. Soatto, "Integral invariants for shape matching," *IEEE Trans. on PAMI*, vol. 28, no. 10, pp. 1602 –1618, 2006.

[20] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. Lopez, and J. V. B. Soares, "Leafsnap: A computer vision system for automatic plant species identification," in *Proc. of ECCV*, October 2012.

[21] F. Pedersoli, N. Adami, S. Benini, and R. Leonardi, "Xkin - extendable hand pose and gesture recognition library for kinect," in *In: Proceedings of ACM Conference on Multimedia 2012 - Open Source Competition*, Nara, Japan, Oct. 2012.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.