Noname manuscript No.
(will be inserted by the editor)

Hand Gesture Recognition with Jointly Calibrated Leap Motion and Depth Sensor

Giulio Marin · Fabio Dominio · Pietro Zanuttigh

Received: date / Accepted: date

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 types 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 two different classifiers, one based on multi-class SVMs and one exploiting Random Forests. Different feature selection algorithms have also been tested in order to reduce the complexity of the approach. 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.

Keywords Depth · Gesture Recognition · Calibration · Kinect · Leap Motion · SVM

1 Introduction

Automatic hand gesture recognition is a very intriguing problem that, if efficiently solved, could open the way to many applications in several different

All authors
Dept. of Information Engineering
Tel.: +39-049-8277774
Fax: +39-049-8277699
E-mail: maringiu,dominiof,zanuttigh@dei.unipd.it

fields, e.g. human-computer interaction, computer gaming, robotics and automatic sign-language interpretation. The problem can be solved both by using wearable devices and with vision-based approach. Vision-based hand gesture recognition [27] 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 [12]. These representations contain a 2D description of the three-dimensional hand pose, which is often difficult to properly understand, and consequently the performed gesture. This is mainly 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 [6] 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^{TM} 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. An overview of the various available approaches will be presented in Section 2.

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 designed for 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 are more accurate (according to a recent study [29] 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 provided 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. 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 is also considered. This is made possible by an ad-hoc approach for the joint calibration of the two devices. Finally, two reliable classification schemes based on Support Vector Machines (SVM) and Random Forests (RF) are proposed. The performances of each of the two devices alone and of their joint exploitation are evaluated. Finally feature selection schemes are considered in order to reduce the dimensionality of the feature vectors. 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 2 presents a brief overview of the related works, then Section 3 presents the general architecture of the proposed gesture recognition system. The two following sections present the feature descriptors extracted from the Leap Motion data (Section 4) and from depth data (Section 5). A method for the joint calibration of the 3D measures from the two devices is presented in Section 6. Then the classification stage is described in Section 7. Experimental results are presented in Section 8 and finally Section 9 draws the conclusions.

2 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 [14], 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., [25] and [28] 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 [23] and [22]. Another approach based on an SVM classifier is [8], 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 [24]. Multi-view setups have also been used for this task [2], 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 [21]. The device has been used for Arabic sign language recognition in [18]: in this work the data extracted from the sensor is fed directly to two different machine learning classification algorithms, one based on a Naive Bayes Classifier and one exploiting Multilayer Perceptron Neural Networks. Another recent work [26] analyzes the trajectory of a finger returned by the Leap Motion in order to recognize handwriting. The approach exploits Dynamic Time Warping and a nearest neighbor search. The sensor has also been used for signature recognition using features based on the optical flow and on the trajectories in a recent work [19]. A gesture interface based on the Leap Motion has been presented in [9], where the authors use the device to control a robot arm.

Finally the work that is more related to this one is [16]. In this work we combined the depth-based descriptors of [8] with some ad-hoc descriptors for the Leap Motion data and fed the two sets to an SVM classifier. However, this approach handles the two sensors separately, with two independent processing pipelines and without jointly calibrating them. The approach proposed in this paper starts from [16] but presents a method for the joint calibration of the two devices and exploits the calibration to extract the features with the combined use of the two sensors. Furthermore the employed feature set has also been updated with a larger feature set for the Leap Motion and new feature descriptors for the depth data.

3 Problem Formulation

The general architecture of the approach presented in this paper is shown in Fig. 1: there are two different feature extraction pipelines, one for the Leap Motion data and one for depth data and finally a classification stage that takes in input all the features and recognizes the performed gesture.

The Leap Motion feature extraction pipeline, described in Section 4 exploits only the data from this sensor and extracts 4 different types of features, i.e., fingertip distances from the centroid of the hand, fingertip elevations from the palm plane, the angles between the vectors connecting the fingertips with the palm center and the 3D positions of the fingertips in the hand reference system.

The second feature extraction pipeline, described in Section 5, is instead mainly based on the information extracted from the depth sensor, even if it exploits also some information from the Leap Motion. It extracts four different sets of features based on the distances of the finger samples from the hand center, on the local curvature of the hand contour, on the similarity between distance feature histograms and on the connected components in the convex hull of the hand shape. Note how the combined use of the data from the two sensors requires the joint calibration of the two devices. An ad-hoc approach

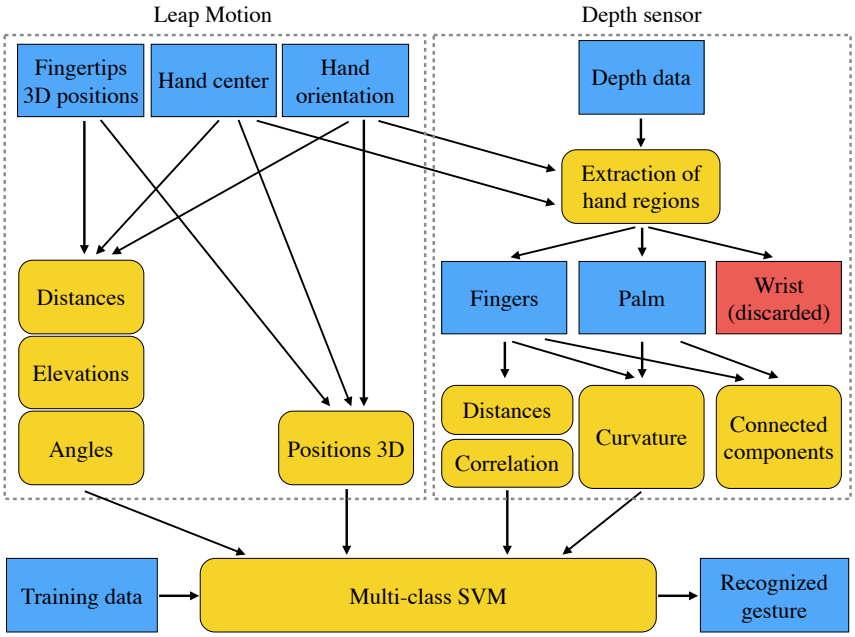


Fig. 1 Pipeline of the proposed approach.

for this critical step based on the fingertips positions in the two reference systems is presented in Section 6.

Finally, the features are processed with the classification method based on Support Vector Machines (SVM) presented in Section 7.

4 Feature extraction from the Leap Motion data

As already stated, the Leap Motion device provides only a limited set of relevant points and not a complete description of the hand shape. The amount of information is more limited if compared to the one from depth sensors like the Kinect, but on the other side the device provides 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 (Fig. 2):

- **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 C** that represents the 3D location roughly corresponding to the center of the palm region in the 3D space.

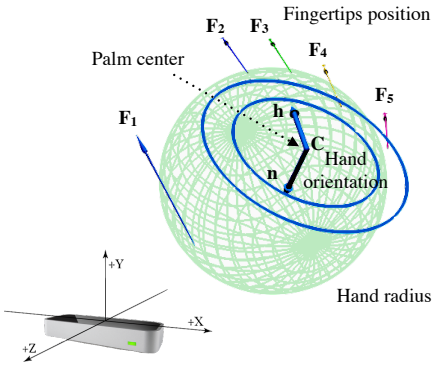


Fig. 2 Data acquired by the Leap Motion device.

- **Hand orientation** consists in two unit vectors representing the hand orientation computed in the palm center **C**. The first vector, denoted with **h**, points from the palm center to the direction of the fingers, while the second, denoted with **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 [29] 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 recognition 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, 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 sleeve 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, the 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 angles respect to the hand direction **h**. To this purpose, we consider the projection of the hand region into the palm plane described by **n** and passing through **C**, as depicted in Fig. 4. The plane is then divided into five angular regions S_i, $i = 1, ..., 5$ as

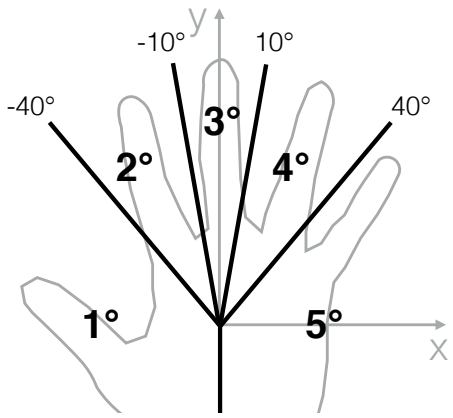


Fig. 3 Angular regions in the palm plane.

in Fig. 3, 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 **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 and these will be described in the rest of this section:

- **Fingertip angles**: angles corresponding to the orientation of each fingertip projected on the palm plane with respect to the hand orientation **h**.
- **Fingertip distances**: 3D distances of the fingertips from the hand center.
- **Fingertip elevations**: distances of the fingertips from the palm region plane.
- **Fingertip positions**: x, y and z coordinates of the fingertips in the 3D space.

All the feature 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. Fig. 4 depicts a sample gesture acquisition and the related feature set.

4.1 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

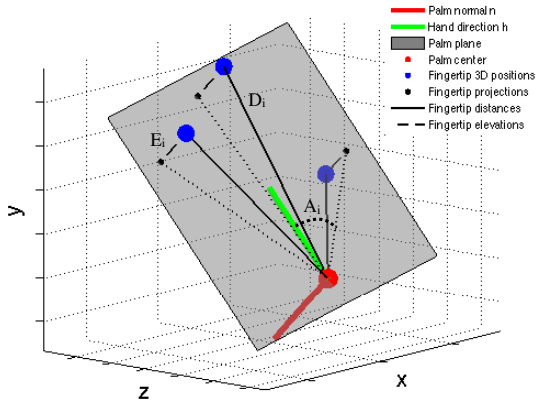


Fig. 4 Considered Leap Motion features on a gesture example (sample of gesture G8 from our dataset).

is defined as:

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

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 has been set to $[0.5, 1]$ to better discriminate, in the classification step, 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 as described before. Fingertip angles features are then collected into vector $\mathbf{F}^a$.

4.2 Fingertip distances

This feature represents the distance of each fingertip from the palm center. Distances are defined as:

$$D_i = ||\mathbf{F}_i - \mathbf{C}||/S, i = 1, ..., N \tag{2}$$

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

4.3 Fingertip elevations

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

$$E_i = \text{sgn}((\mathbf{F}_i - \mathbf{F}_i^{\pi}) \cdot \mathbf{n})||\mathbf{F}_i - \mathbf{F}_i^{\pi}||/S, i = 1, ..., N \tag{3}$$

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 have been set to 0. Note that as for the fingertip angles, the values range has been scaled to the interval $[0.5, 1]$ and then collected into vector $\mathbf{F}^e$.

4.4 Fingertip 3D positions

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:

$$\begin{aligned} P_i^x &= (\mathbf{F}_i - \mathbf{C}) \cdot (\mathbf{n} \times \mathbf{h}) \\ P_i^y &= (\mathbf{F}_i - \mathbf{C}) \cdot \mathbf{h} \\ P_i^z &= (\mathbf{F}_i - \mathbf{C}) \cdot \mathbf{n} \end{aligned} \tag{4}$$

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

5 Feature 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 [7, 8]. 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 safely 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 description.

- **Distance features**: consider the distance of each point of the hand contour from the palm center to describe the hand shape.
- **Correlation features**: these are a measure of similarity between distance features.
- **Connected components features**: exploiting the convex hull, compute the size and the number of connected components in the performed gesture.

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

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

In our previous approach [8] 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 [8], however it is clear that it is a quite complex operation as most of the computation time of the entire pipeline of [8] was spent on this step. Moreover, there is 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 scenarios.

Since in the proposed approach the Leap Motion data are also available, this information can be exploited to make the identification of the hand position and of its orientation faster and more reliable. Firstly the hand centroid computed by the Leap Motion $\mathbf{C}$ can be expressed according to the depth camera coordinate system using the calibration information. In this way, if the Leap Motion correctly recognizes the hand, we can ensure that the hand is properly identified even if there are objects of similar shape and color in the depth sensor acquisition. Moreover, we can also avoid the use of color information thus making the approach faster and allowing the use of depth sensors that do not have an associated color camera (e.g., industrial matricial ToF sensors like MESA or PMD devices). In this section we will assume that the two devices have been jointly calibrated obtaining a rotation matrix R and a translation vector $\mathbf{t}$ between the two reference systems. How to perform

the calibration will be the subject of Section 6. The location of the Leap Motion hand centroid in the depth camera reference system will be denoted with $\mathbf{C}_D = R\mathbf{C} + \mathbf{t}$ 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:

$$\mathcal{H} = \{X : \|\mathbf{X} - \mathbf{C}_D\|^2 \leq r_h\} \tag{5}$$

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 [8]. A more refined scheme employing an ellipse in place of the circle can also be used [17]. Let us denote with $\mathbf{C}_{palm}$ the final circle and with r its radius computed by the algorithm.

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 4), therefore also the computation of the PCA can be avoided. Another critical aspect in the approach of [8] 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 on 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 the circle of radius r centered on $\mathbf{C}_{palm}$; the finger samples set $\mathcal{F}$ contains the samples 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 ($\mathcal{W}$).

5.2 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 X in the hand contour, extracted from the hand mask in the depth image, the distance $d(\mathbf{X})$ with respect to the hand center $\mathbf{C}_{palm}$:

$$d(\mathbf{X}) = ||\mathbf{X} - \mathbf{C}_{palm}|| \tag{6}$$

Given the hand orientation, then, we are able to provide a coherent function $d(\mathbf{X})$ among different gestures and repetitions. For example 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, as it depends on the actual distance of the hand from the camera. Therefore, in order to make the descriptor independent from the hand to camera distance, 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. 7a.

The distance function $d(\mathbf{X})$ 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 [8]: the approach proposed in this work turned out to be simpler, faster and more accurate.

5.3 Correlation features

This feature set is based on the similarity between distance functions of subsection 5.2. For each considered gesture, a reference acquisition is selected and the corresponding distance function is computed with the approach of Eq. 6, thus obtaining a set of reference functions $d^r_g(\mathbf{X})$, 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 $d^r_g(\mathbf{X})$ is selected:

$$R_g = \max_{\Delta} \left[\rho \left(d(\mathbf{X}), d^r_g(\mathbf{X} + \Delta) \right), \rho \left(d(-\mathbf{X}), d^r_g(\mathbf{X} + \Delta) \right) \right] \tag{7}$$

where $g = 1, \ldots, G$ and $d(-\mathbf{X})$ is the flipped version of the distance function to account for the possibility for the hand to 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^ρ_g for each of them. Note how, ideally, the correlation with the correct gesture should have a larger value than the others.

5.4 Curvature features

This feature set describes the curvature of the hand edges on the depth map. A scheme based on on integral invariants [15,13] has been used. The approach for the computation of this feature is basically the same of [8]. 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 $f^c_{b,s}$ 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.

5.5 Connected components features

Another useful clue used for gesture recognition schemes [20] is the convex hull of the hand shape in the depth map. The idea is to look for regions within the convex hull of the hand shape but not belonging to the hand. These typically correspond to the empty regions between the fingers and those are a good clue to recognize the fingers arrangement. Let $\mathcal{S} = C_{hull}(\mathcal{B}) \setminus \mathcal{B}$ be the difference between the convex hull and the hand shape (see Fig. 5 a and b). Region $\mathcal{S}$ 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 in the processing small components due to noise, as the one shown on the right of the hand in Fig. 5 c). The output of this procedure is the set $\mathcal{S}_{cc} = \{S_i : S_i > T_{cc}\}$ (Fig. 5 c and d).

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

$$f^{cc}_i = \frac{area(S_i | S_i \in \mathcal{S}_{cc})}{area(C_{hull}(\mathcal{B})))} \tag{8}$$

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 pinky). These numbers are then collected into vector $\mathbf{F}^{cc}$.

6 Calibration

Since the employed acquisition setup jointly exploits the 3D measures from two different sensors, i.e, the Leap Motion device and the depth sensor (with optionally a color camera rigidly attached to the depth one), it is necessary to jointly calibrate the two devices in order to bring the measures of one sensor in the reference system of the other. The proposed approach is independent from the relative position of the two sensors, however notice that a set of practical limitations of the sensors limits the choices in the setup construction:

a) b) c) d)

Fig. 5 Areas of the connected components: a) and b): difference between the convex hull and the hand shape; b) connected components in set $\mathcal{S}_{cc}$ highlighted in green.

- The Leap Motion must be placed under the hand, typically on the desk looking up. Furthermore its operating range is limited.
- The depth sensor has typically a minimum distance that it can acquire. This distance depends on the employed sensor, e.g., the Kinect we used for the results section has the limitation that it cannot acquire objects closer than $50[cm]$ to the sensor. The maximum distance is instead typically bigger than the Leap Motion one.
- If the palm plane is roughly perpendicular to the optical axis of the depth camera more depth samples are acquired for the hand leading to better performances
- Inside the working range, also having the sensor closer to the hand leads to more accurate data

Considering all the previous observations, we found that the setup that allows to obtain the best performance is the one shown in Fig.6. As it is possible to see from the figure, in the proposed setup 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 most gesture acquisition systems using this sensor.

The aim of the calibration procedure is to estimate the extrinsic parameters of the two devices, i.e., the coordinate system transformation between the reference systems of the two devices, or equivalently the position of one sensor with respect to the other one. Notice that our implementation for testing the algorithm uses the Kinect sensor but the proposed calibration algorithm remains valid also for other depth cameras. In particular, our approach does not require an additional color stream. Furthermore, 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 [10], 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 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

Fig. 6 Acquisition setup.

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, including 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 is not used in the proposed approach).

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 4), 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. 10). 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 pre-processing they can be ordered and then associated to the correct fingers. The same points then need to be detected also from the depth camera. The two sets of points will then be used inside the calibration algorithm. 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. Moreover, 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, the system re-calibration only requires the acquisition of a couple of frames of the user's open hand. Notice that a new calibration is mandatory only if the devices are moved, and is optional when a new user starts to interact with the system.

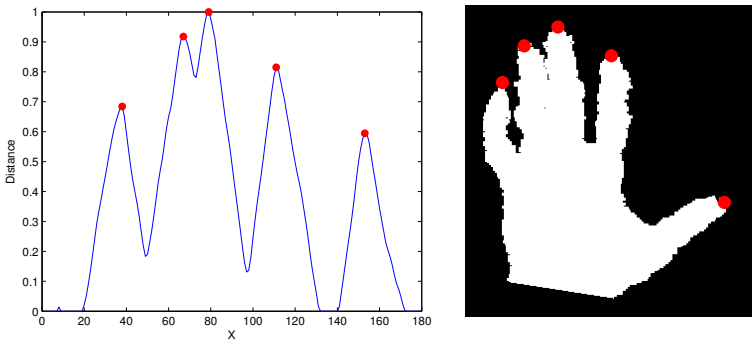
6.1 Extraction of fingertips position from Leap Motion data

Starting from the hand orientation and the palm center estimated from the Leap Motion, the palm plane can be extracted and the fingertips projected on it. 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 remaining fingertips by increasing angular values, up to the fingertip with the greatest angular value associated to the pinky. Section 4 presents a description of the data acquired from the sensor and in particular provides more details on the angle computation. After this operation we obtain a set of 5 points $\mathcal{X}_L = \mathbf{X}_L^1, ..., \mathbf{X}_L^5$ describing the fingertips in the Leap Motion coordinate system.

6.2 Extraction of fingertip positions from depth data

For the depth sensor, instead, a more complex approach is required to extract fingertip positions from the acquired depth image. In order for the calibration process to be completely automatic, we decided to avoid the need to manually selecting points, relying instead on 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. Notice that the hand extraction scheme of Section 5.1 exploits also the Leap Motion data so it can not be directly applied in this case. The extraction of the hand has instead been performed using the approach of [8] where the hand center is initially estimated by using a Gaussian filter on the samples density and then refined by fitting a circle on the palm. Finally PCA is used for the computation of the hand orientation.

Then the hand contour is analyzed using the same approach used for the distance features in Section 5. The distance d of each point X of the hand contour from the palm center is computed, thus obtaining the 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 needs 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 have been detected, the correspondent values in the depth image are selected and through the projection matrix of the depth camera they are back-projected in the 3D space obtaining the 3D coordinates of the fingertips in the depth camera coordinate system $\mathcal{X}_D = \{\mathbf{X}_D^1, ..., \mathbf{X}_D^5\}$. Fig. 7 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 noticing 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. 7 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 on the hand mask of the depth image.

6.3 Roto-translation estimation

The final step is the computation of the roto-translation that links the two reference systems. In order to be more robust against noise, even if a single frame is theoretically sufficient we acquire several frames. Let us denote with $\mathcal{X}_{L,f}$ and $\mathcal{X}_{D,f}$ the sets of points acquired by the Leap Motion and the depth camera respectively, each relative to each frame $f = 1, ..., F$. With the acquired fingertip 3D positions, the goal is to find the roto-translation parameters R and $\mathbf{t}$ that minimize the average reprojection error of all the considered fingertip points in all the acquired frames:

$$(R, \mathbf{t}) = \arg\min_{R,\mathbf{t}} \sum_{f=1}^{F} \sum_{i=1}^{5} ||R\mathbf{X}^i_{L,f} + \mathbf{t} - \mathbf{X}^i_{D,f}||^2 \tag{9}$$

i.e., to find the best roto-translation that brings the point cloud $\mathcal{X}_L$ to the point cloud $\mathcal{X}_D$ (the point clouds $\mathcal{X}_L$ and $\mathcal{X}_D$ are the union of all the points clouds of the considered frames). Since the corresponding set of equations corresponds to an over-determined system and the measures are affected by noise, we used a RANSAC robust estimation approach to solve it. 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 (9) for all the tested people is about 9 $[mm]$.

7 Gesture classification

The approaches of Sections 4 and 5 produce eight different feature vectors, four for the Leap Motion data and four for the depth data. Each vector describes some relevant clues regarding the performed gesture and in order to perform the recognition, two different classification schemes have been used,

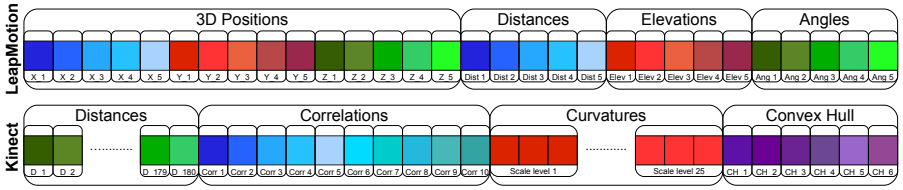


Fig. 8 Feature vectors extracted from the two devices.

one based on a multi-class Support Vector Machine classifier and one based on Random Forests. 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 $\mathbf{V}_{depth} = [\mathbf{F}^l, \mathbf{F}^\rho, \mathbf{F}^c, \mathbf{F}^{cc}]$ that collects the features computed from depth information. Feature vectors extracted from the two devices are visually summarized in Fig. 8. Each vector can be used alone or together with any of the other descriptors. The combination of multiple feature descriptors can be obtained by simply concatenating the vectors corresponding to the selected features. The target of the approach is to classify the performed gestures into G classes, one for each gesture in the considered database.

The first classification scheme exploits a multi-class SVM classifier [4] based on the *one-against-one* approach. 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 been 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 by 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 in the validation set. The couple of parameters that gives the best accuracy on average is 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.

Alternatively we also tested a second classification scheme exploiting Random Forests (RF) [3]. Each tree has been trained on a random sampling of the training set leaving out one third of the sampled vectors for the estimation of the out-of-bag error. The only model parameter to optimize, differently from the pair for the RBF kernel of SVM, is the size m of the feature subset in each node. The parameter controls a trade-off between the tree correlation and the predictive "strength" of each tree, and may be easily found by analyzing the out-of-bag error. The size of the forest, is not a critical parameter since the classification error remains relatively stable if a sufficient number of trees is used. In our case we trained a Random Forest of 100 decision trees

with a default value of $m = \sqrt{|\mathbf{F}|}$ with $|\mathbf{F}|$ the length of the feature vectors in the dataset ($|\mathbf{F}| = 435$ when all the considered features are used). The implementation of the Random Forest classifier provided by Matlab has been used.

Finally, since the considered vectors contain a large number of elements we also considered the use of feature selection schemes in order to reduce the number of features and avoid the usage of useless or redundant descriptors. Three different feature selection schemes have been tested. The first uses the *F-score* approach [5], i.e., the F-score is computed for each feature and the most discriminative features according to this measure are selected (i.e., the features with an F-score bigger than a pre-defined threshold). Two different thresholds have been used in order to produce two subsets with 16 and 128 features.

The second scheme is based on the Forward Sequential Selection (FSS) algorithm [1]. In this case, starting from the empty set, at each step a new feature is added to the selected ones by choosing the one that allows to obtain the larger improvement in the classification accuracy with respect to the previous step (the SVM classifier previously described has been used to evaluate the classification accuracy).

Finally a third feature selection scheme exploiting Random Forests has been tested. In this case a classification is performed with the approach of [3] and the out-of-bag error is estimated. Then, in order to measure the importance of the various features, the values of one of the features are permuted and the out-of-bag error is estimated again. The procedure is repeated for each feature and the importance of each feature is given by the normalized average increase of the out-of-bag error after the permutation. This approach is detailed in [5]. The number of selected features is the same of the previous cases in order to allow a fair comparison.

8 Experimental results

The results have been obtained using the setup depicted in Fig. 6. A Leap Motion device and a Kinect have been used to jointly acquire the data relative to the performed gestures. The first generation 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 6 and synchronized in time. A software synchronization scheme has been used: its precision is sufficient for the recognition of gestures based on static poses like the ones considered in this paper. The considered dataset of gestures contains the 10 different gestures shown in Fig. 10 executed by 14 different people. 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 is available on our website at the url `http://lttm.dei.unipd.it/downloads/gesture`. In order

to compute the results we split the dataset in a train and a test set by using the *leave-one-person-out* approach of Section 7, i.e., we placed in the training set the data from all the users except one and in the test set the data from the remaining user. Since the amount of data associated to a single user (100 samples) is not sufficient for a reliable assessment of the performances we executed 14 completely independent tests changing each time the person in the test set, i.e., as shown in Fig. 9, in each test we used a train set with 13 people and a test set with a single person that is the remaining one. The results of the 14 tests have been averaged to obtain the final accuracy. Note that this is a more challenging test than the standard *leave-one-out* approach, since not only it guarantees that the data in the train set is different from the ones in the test set as in the standard case, but also that the train set does not contain any sample from the user in the test set. This means that the system should be able to classify the data from the user in the test set from what it has *learned* from users different from the one that is using it, a typical situation in real setups. This approach has been used to train both classifiers, i.e., the Support Vector Machines (SVM) one and the one exploiting Random Forests (RF) as explained in Section 7. In this section we will firstly report the performance that can be obtained by using the SVM classifier (that is the better performing one) with the various feature types of each of the two sensors alone. Then the results that can be obtained by combining the two sensors will be presented. Finally we will show the accuracy that can be obtained with various combinations of classifiers (SVM or RF) and of feature selection strategies.

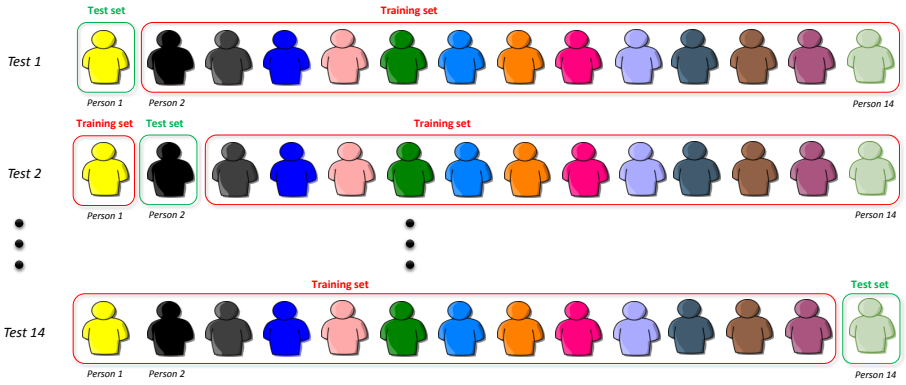


Fig. 9 The results are the average of 14 independent tests each one performed by placing a person in the test set and the remaining 13 in the train set.

Let us start from the Leap Motion device. Table 1 shows the accuracy obtained using the classification algorithm of Section 7 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

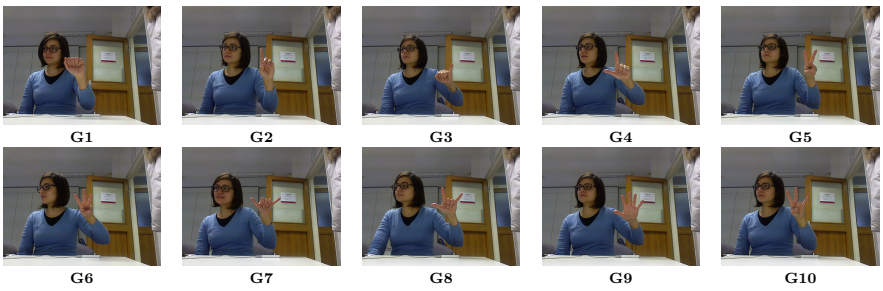


Fig. 10 Gestures from the American Sign Language (ASL) contained in the database that has been acquired for experimental results.

some gestures is not always optimal, as it is possible to see from the confusion matrix in Table 2. For example, gestures G2 and G3 are sometimes confused with gesture G1. This is due mostly to the false positives returned by the Leap Motion sensor that sometimes detects a raised finger in 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%
$\mathbf{F}^d + \mathbf{F}^a + \mathbf{F}^e$	80.9%

Table 1 Performance with the Leap Motion data.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	**0.893**		0.021	0.064	0.021					
G2	0.300	**0.564**	0.136							
G3	0.143	0.093	**0.700**	0.043			0.021			
G4	0.029			**0.900**			0.050		0.007	0.014
G5	0.050	0.050	0.029	0.021	**0.757**	0.014	0.021	0.021		0.036
G6	0.007		0.029		0.029	**0.836**	0.014	0.014		0.071
G7	0.014		0.036	0.079			**0.814**	0.029	0.007	0.021
G8				0.036	0.029		0.029	**0.829**		0.079
G9		0.007		0.007				0.014	**0.971**	
G10			0.014		0.036	0.007	0.050	0.007		**0.886**

Table 2 Confusion matrix for the 3D positions from the Leap Motion data. *Yellow* cells represent true positive, while *gray* cells show false positive with failure rate greater than 5%.

Fingertip distance features allow to obtain an accuracy of about 76%: they are 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 three 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 three 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 a 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 ($\mathbf{F}^l$)	94.4%
Correlations features ($\mathbf{F}^\rho$)	68.7%
Curvature features ($\mathbf{F}^c$)	86.2%
Convex Hull features($\mathbf{F}^{cc}$)	70.5%
$\mathbf{F}^l + \mathbf{F}^c$	96.35%

Table 3 Performance with the depth data.

Table 3 shows the results obtained from the depth information 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. This descriptor alone allows 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 time they still can be considered for applications where the running time and the memory footprint of the descriptors 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 does 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 sim-

ple 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 (Table 4). 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	**0.971**	0.021	0.007							
G2	0.007	**0.971**	0.021							
G3	0.007		**0.986**		0.007					
G4		0.036		**0.964**						
G5		0.007			**0.986**	0.007				
G6		0.036			0.036	**0.893**		0.014		0.021
G7			0.014				**0.986**			
G8				0.007		0.014		**0.964**	0.007	0.007
G9				0.007		0.007			**0.986**	
G10					0.007	0.043		0.021		**0.929**

Table 4 Confusion matrix for the combined use of distance and curvature descriptors from depth data. *Yellow* cells represent true positive.

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

Table 5 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 5. The corresponding confusion matrix (Table 7) 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 consider that Leap Motion data are used also for the computation of the depth-based features (i.e., for the initial centroid and hand orientation) and allow to reduce the computational time as it will be shown at the end of this section. Furthermore Leap Motion data allow a more reliable extraction of the hand in some complex settings, a feature that is not possible to appreciate on the employed dataset. Finally the Leap Motion provides a few but very relevant features and allows to obtain a good accuracy with a smaller number of features with respect to the depth-based approach.

A comparison with [16], that presents an earlier version of this approach, shows how the proposed algorithm clearly outperform the previous method (see Table 6). By exploiting both sensors, the accuracy is 96.5% against 91.3% of the previous scheme, a quite relevant improvement. This result is mostly due to the improvement in the feature extraction scheme from depth data, that has an accuracy of 96.3% instead of 89.7% of the previous scheme. This proves the reliability of the new depth features extraction algorithm exploiting the Leap Motion data and a more refined distance features extraction scheme.

Feature set	Accuracy	
	Marin et Al. [16]	Proposed method
Leap Motion features	80.9%	**81.5**%
Kinect features	89.7%	**96.3**%
Leap Motion + Kinect features	91.3%	**96.5**%

Table 6 Comparison between the performances of the proposed approach and of [16].

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1	**0.979**	0.021								
G2	0.014	**0.964**	0.021							
G3	0.007	0.007	**0.986**							
G4		0.029		**0.971**						
G5	0.007				**0.986**	0.007				
G6		0.029			0.043	**0.886**		0.007		0.036
G7			0.014				**0.986**			
G8				0.014		0.014		**0.957**	0.007	0.007
G9				0.007		0.007			**0.986**	
G10					0.007	0.029		0.014		**0.950**

Table 7 Confusion matrix for the combined use of Leap Motion and depth data. *Yellow* cells represent true positive.

In Section 7 a second classification scheme based on Random Forests has been presented. This approach is simple and fast and does not require the complex grid search procedure for the optimization of the parameters. On the other side this classifier has slightly lower performances than the SVM approach and with the complete feature set is able to achieve an accuracy of 94.7%, a very good result but about 2% lower than the one of the SVM classifier.

The proposed approach makes use of a large number of features, with the complete feature set each vector has 435 elements. Furthermore there is also a much larger number of feature values extracted from the Kinect data with respect to the ones from the Leap Motion. For these reasons it is reasonable to employ a feature selection scheme to reduce the number of features and to better balance the information coming from the two sensors. As already described three different feature selection strategies have been tested, i.e., F-Score, Sequential Feature Selection and Random Forests. All the three

methods have been tested both with the SVM and the RF classifier. For each combination of feature selection strategy and classifier we selected the 16 and 128 best features. The results are presented in Table 8. The table shows how by properly selecting the best features it is possible to greatly reduce the number of employed features with only a limited impact on the performances.

The F-Score feature selection method is the simplest and fastest but also the one leading to the worst results. In particular if the number of features is reduced to 128 (about one third of the original number of features), this approach is still able to achieve acceptable performances with a loss of about 2% on the accuracy of the SVM classifier. If the number of features is further reduced to 16 this approach is instead not able to properly select a good combination of features, mostly due to the fact that it does not properly account for the correlation between the different features. In this case there is a huge performance drop with an accuracy of 60%, more than 36% less than the one obtained with all the features. If the F-Score approach is used together with the Random Forests classifier the results are very similar with losses on the accuracy of 2.1% (128 features) and of 37.2% (16 features).

The sequential feature selection algorithm is instead the best performing one when the SVM classifier is used. The accuracy is very close to the original value with both 128 and 16 features. Even by using only 16 features the accuracy is only 0.7% less than the optimal value obtained by using all the features. This is a quite impressive result and opens the way to several optimization and simplification strategies for the proposed approach. Results are very good also for the Random Forest classifier, the loss in this case is 0.6% with 128 features and 4% with 16 features. Notice how in this case the reduction to 16 features has a more noticeable impact.

Finally Random Forests can be used also for the feature selection. If they are used together with the SVM classifier the performances are very good but slightly worse than the ones of the sequential feature selection scheme, specially if 16 features are used. In this case there is a loss of about 3%, much better than the F-score but not so good as the sequential feature selection result. When, instead, Random Forests are used for both the feature selection and the classification, results are very similar to the sequential feature selection strategy (in fact even better although with a very small difference), according to the idea that having the same approach used for both steps also allows to simplify and speed-up the training procedure.

Concluding, the best solution for optimal performances is to use the Sequential Feature Selection scheme together with the SVM classifier. The Random Forests for both training and classification can be used when a simpler and faster training phase is needed.

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 performances have been obtained. The initial hand detection phase, that took $46ms$ in the implementation of the approach of [8] and that we used to start the de-

	SVM			RF		
Feature selection strategy	435	128	16	435	128	16
F-Score	96.5	94.5%	60.1%	94.7	92.6%	57.5%
Sequential		95.9%	95.8%		94.1%	90.7%
Random Forests		95.8%	93.7%		94.2%	90.8%

Table 8 Performances with different combinations of classification algorithms and feature selection strategies.

velopment of this work can now be completed in a few milliseconds thanks to the exploitation of the Leap Motion centroid. Notice also that the processing of color data for the check on skin color compatibility has also been removed in this work since it was used only in the initial phase. 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 are 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.

9 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 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 algorithms. 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. Performances remain very good even when the classification algorithm is changed or feature selection approaches are used to reduce the dimensionality of the feature vectors.

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. D.W. Aha and R.L. Bankert. A comparative evaluation of sequential feature selection algorithms. In Doug Fisher and Hans-J. Lenz, editors, *Learning from Data*, volume 112 of *Lecture Notes in Statistics*, pages 199–206. Springer New York, 1996.
2. L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *Proc. of ECCV*, October 2012.
3. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
4. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.
5. Yi-Wei Chen and Chih-Jen Lin. Combining svms with various feature selection strategies. In *Feature extraction*, pages 315–324. Springer, 2006.
6. C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo. *Time-of-Flight Cameras and Microsoft Kinect*. SpringerBriefs in Electrical and Computer Engineering. Springer, 2012.
7. 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*, pages 9–16. ACM, 2013.
8. F. Dominio, M. Donadeo, and P. Zanuttigh. Combining multiple depth-based descriptors for hand gesture recognition. *Pattern Recognition Letters*, 50:101—111, December 2014.
9. Cesar Guerrero-Rincon, Alvaro Uribe-Quevedo, Hernando Leon-Rodriguez, and Jong-Oh Park. Hand-based tracking animatronics interaction. In *Robotics (ISR), 2013 44th International Symposium on*, pages 1–3, 2013.
10. D. Herrera, J. Kannala, and J. Heikkilä. Joint depth and color camera calibration with distortion correction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(10):2058–2064, 2012.
11. C. Keskin, F. Kirac, Y.E. Kara, and L. Akarun. Real time hand pose estimation using depth sensors. In *ICCV Workshops*, pages 1228 –1234, nov. 2011.
12. D.I. Kosmopoulos, A. Doulamis, and N. Doulamis. Gesture-based video summarization. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–1220–3, 2005.
13. 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.
14. 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.
15. S. Manay, D. Cremers, Byung-Woo Hong, A.J. Yezzi, and S. Soatto. Integral invariants for shape matching. *IEEE Trans. on PAMI*, 28(10):1602 –1618, 2006.
16. G. Marin, F. Dominio, and P. Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Paris, France, 2014.
17. 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.
18. M. Mohandes, S. Aliyu, and M. Deriche. Arabic sign language recognition using the leap motion controller. In *Industrial Electronics (ISIE), IEEE 23rd International Symposium on*, pages 960–965, June 2014.
19. I. Nigam, M. Vatsa, and R. Singh. Leap signature recognition using hoof and hot features. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, Paris, France, 2014.

20. 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.
21. 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*, OzCHI '13, pages 175–178, New York, NY, USA, 2013. ACM.
22. Z. Ren, J. Meng, and J. Yuan. Depth camera based hand gesture recognition and its applications in human-computer-interaction. In *Proc. of ICICS*, pages 1 –5, 2011.
23. 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*, pages 1093–1096. ACM, 2011.
24. 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*, pages 1297–1304. IEEE, 2011.
25. P. Suryanarayan, A. Subramanian, and D. Mandalapu. Dynamic hand pose recognition using depth data. In *Proc. of ICPR*, pages 3105 –3108, aug. 2010.
26. S. Vikram, L. Li, and S. Russell. Handwriting and gestures in the air, recognizing on the fly. In *ACM Conference on Human Factors in Computing Systems (CHI) Extended Abstracts*, 2013.
27. J.P. Wachs, M. Kölsch, H. Stern, and Y. Edan. Vision-based hand-gesture applications. *Commun. ACM*, 54(2):60–71, February 2011.
28. J Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu. Robust 3d action recognition with random occupancy patterns. In *Proc. of ECCV*, 2012.
29. F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.