Combining multiple depth-based descriptors for hand gesture recognition

Fabio Dominio, Mauro Donadeo, Pietro Zanuttigh*

Department of Information Engineering, University of Padova, Italy

Abstract

Depth data acquired by current low-cost real-time depth cameras provide a more informative description of the hand pose that can be exploited for gesture recognition purposes. Following this rationale, this paper introduces a novel hand gesture recognition scheme based on depth information. The hand is firstly extracted from the acquired data and divided into palm and finger regions. Then four different sets of feature descriptors are extracted, accounting for different clues like the distances of the fingertips from the hand center and from the palm plane, the curvature of the hand contour and the geometry of the palm region. Finally a multi-class SVM classifier is employed to recognize the performed gestures. Experimental results demonstrate the ability of the proposed scheme to achieve a very high accuracy on both standard datasets and on more complex ones acquired for experimental evaluation. The current implementation is also able to run in real-time.

Keywords: Gesture recognition, Support Vector Machines, Depth, Kinect

*Corresponding author: Pietro Zanuttigh. email: zanuttigh@dei.unipd.it, phone: +39 049 827 7782, fax: +39 049 827 7699, Address: Dept. of Information Engineering, Via Gradenigo 6/B, 35131 Padova, Italy

Email address: dominiof,donadeom,zanuttigh@dei.unipd.it (Fabio Dominio, Mauro Donadeo, Pietro Zanuttigh)

1. Introduction

Hand gesture recognition is an intriguing problem for which many different approaches exist. Even if gloves and various wearable devices have been used in the past, vision-based approaches able to capture the hand gestures without requiring any physical device to be worn allow a more natural interaction with computers and many other devices. This problem is currently raising a high interest due to the rapid growth of application fields where it can be efficiently applied, as reported in recent surveys, e.g. (Wachs et al., 2011; Garg et al., 2009). These include human-computer interaction, where gestures can be used to replace the mouse in computer interfaces and also to allow a more natural interaction with mobile and wearable devices like smartphones, tablets or newer devices like the Google glasses. Also the navigation of 3D virtual environments is more natural if controlled by gestures performed in the 3D space. In robotics gestures can be used to control and interact with the robots in a more natural way. Another key field is computer gaming, where devices like Microsoft's Kinect have already brought gesture interfaces to the mass market. Automatic sign-language interpretation will also allow to help hearing and speech impaired people to interact with the computer. Hand gesture recognition can be applied in the healthcare field to allow a more natural control of diagnostic data and surgical devices. Gesture recognition is also being considered for vehicle interfaces.

Several hand gesture recognition approaches, based on the analysis of images and videos, can be found in literature (Wachs et al., 2011; Zabulis et al., 2009). Images and videos provide a bidimensional representation of the hand

pose, which is not always sufficient to capture the complex movements and inter-occlusions characterizing hand gestures. Three dimensional representations offer a more accurate description of the hand pose, but are more difficult to be acquired. The recent introduction of low-cost consumer depth cameras, such as Time-Of-Flight cameras and Microsoft's Kinect^{TM}, has made depth acquisition available to the mass market, thus widely increasing the interest in gesture recognition approaches taking advantage from three-dimensional information.

In order to recognize the gestures from depth data the most common approach is to extract a set of relevant features from the depth maps and then exploit machine learning techniques to the extracted features. Kurakin et al. (2012) uses a single depth map and extract silhouette and cell occupancy features for building a shape descriptor that is then fed into a classifier based on action graphs. Suryanarayan et al. (2010) extract 3D volumetric shape descriptors from the hand depth to be classified with a Support Vector Machine. Volumetric features and an SVM classifier are also used by Wang et al. (2012). In Keskin et al. (2012) the classification is instead performed using Randomized Decision Forests (RDFs). RDFs are also used by Pugeault and Bowden (2011) that also combines together color and depth information to improve the accuracy of the classification. Another approach consists in analysing the segmented hand shape and extract features based on the convex hull and on the fingertips positions as in Wen et al. (2012) and Li (2012). A similar approach is used also by the Open-source library *XKin* (Pedersoli et al., 2012). Finally, Ren et al. (2011b) and Ren et al. (2011a) compare the histograms of the distance of hand edge points from the hand center.

If the target is the recognition of dynamic gestures, motion information and in particular the trajectory of the hand's centroid in the 3D space can be exploited (Biswas and Basu, 2011). In Doliotis et al. (2011) a joint depth and color hand detector is used to extract the trajectory that is then fed to a Dynamic Time Warping (DTW) algorithm. Finally, Wan et al. (2012) exploits both the convex hull on a single frame and the trajectory of the gesture. A related harder problem is the estimation of the hand pose from the depth data (Oikonomidis et al., 2011),(Ballan et al., 2012),(Keskin et al., 2011).

In most of the previously cited works depth data is mainly used to reliably extract the hand silhouette in order to exploit approaches derived from hand gesture recognition schemes based on color data. This paper instead uses a set of three-dimensional features to properly recognize complex gestures by exploiting the 3D information on the hand shape and finger posture contained in depth data. Furthermore instead of relying on a single descriptor extraction scheme, different types of features capturing different clues are combined together to improve the recognition accuracy. In particular the proposed hand gesture recognition scheme exploits four types of features: the first two sets are based on the distance from the palm center and the elevation of the fingertips, the third contains curvature features computed on the hand contour and the last set of features is based on the geometry of the palm region accounting also for fingers folded over the palm. The constructed feature vectors are then combined together and fed into an SVM classifier in order to recognize the performed gestures. The proposed approach introduces several novel elements: it jointly exploits color and depth

data to reliably extract the hand region and is able to extract wrist, palm and finger regions; it fully exploits three-dimensional data for the feature extraction, and finally it combines features based on completely different clues to improve the recognition rate.

The paper is articulated as follows: Section 2 introduces the general architecture of the proposed gesture recognition system, Section 3 explains how the hand region is extracted from the acquired depth data and segmented into arm, palm and fingers regions. Section 4 describes the computation of the proposed feature descriptors, and Section 5 presents the classification algorithm. Section 6 reports the experimental results and finally Section 7 draws the conclusions.

2. Proposed gesture recognition system

The proposed gesture recognition system (Fig. 1) encompasses three main steps. In the first step the hand samples are segmented from the background exploiting both depth and color information. The previous segmentation is then refined by further subdividing the hand samples into three non overlapping regions, collecting palm, fingers and wrist/arm samples respectively. The last region is discarded, since it does not contain information useful for gesture recognition. The second step consists in extracting the four feature sets that will be used in order to recognize the performed gestures, i.e.:

- **Distance features:** this set describes the Euclidean 3D distances of the fingertips from the estimated palm center.
- **Elevation features:** this set accounts for the Euclidean distances of the fingertips from a plane fitted on the palm samples. Such distances

may also be considered as the *elevations* of the fingers with respect to the palm.

- **Curvature features:** this set describes the curvature of the contour of the palm and fingers regions.

- **Palm area features:** this set describes the shape of the palm region and helps to state whether each finger is raised or bent on the palm.

Finally, during the last step, all the features are collected into a *feature vector* to be fed into a multi-class Support Vector Machine classifier in order to recognize the performed gesture.

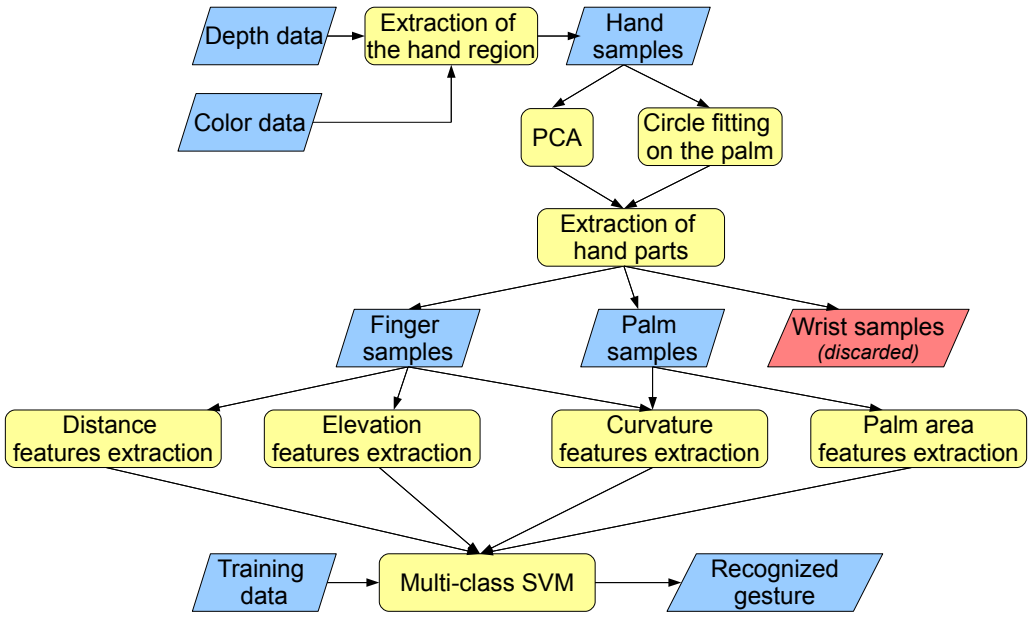


Figure 1: Architecture of the proposed gesture recognition system.

3. Hand segmentation

The first step in the proposed method is the segmentation of the hand. Although depth information alone may be enough for this purpose, we exploit both depth and color information in order to recognize the hand more robustly. The data acquired by the Kinect™ color camera is first projected on the depth map and both a color and a depth value are associated to each sample. Note that the Kinect™ depth and color cameras have been previously jointly calibrated by the method proposed in (Herrera et al., 2012). After projection, the acquired depth map $D(u, v)$ is thresholded on the basis of color information. More specifically, the colors associated to the samples are converted into the CIELAB color space and compared with a reference skin color that has been previously acquired[1]. The difference between each sample color and the reference skin color is evaluated and the samples whose color difference is below a pre-defined threshold are discarded. This first thresholding will only retain depth samples associated with colors compatible with the user's skin color that are very likely to belong to the hand, the face or other uncovered body parts. After the skin color thresholding the hand region has a higher chance to be the object nearest to the Kinect™. Note that this is the only step of the algorithm where color data is used. In applications where the hand is proven to be always the closest object to the sensor, the usage of color information may be skipped in order to simplify the acquisition of the data and to improve computation performances.

[1] A reference hand or alternatively a standard face detector (Viola and Jones, 2001) can be used to extract a sample skin region.

Let us denote with $\mathbf{X}_{u,v}$ a generic 3D point acquired by the depth camera, i.e., the back-projection of the depth sample in position (u, v). A search for the sample with the minimum depth value D_{min} on the thresholded depth map is performed. The corresponding point $\mathbf{X}_{min}$ is chosen as the starting point for the hand detection procedure. In order to avoid to select as $\mathbf{X}_{min}$ an isolated artifact due to measurement noise, our method verifies the presence of an adequate number of samples with a similar depth value in a 5×5 region around $\mathbf{X}_{min}$. If the cardinality threshold is not satisfied we select the next closest point and repeat the check.

Let us now denote by $\mathcal{H}$ the hand samples set. Points belonging to $\mathcal{H}$ cannot have a depth that differs from $\mathbf{X}_{min}$ of more than a value T_{depth} that depends on the hand size. $\mathcal{H}$ may be then expressed as:

$$\mathcal{H} = \{\mathbf{X}_{u,v} | D(u, v) < D_{min} + T_{depth}\} \tag{1}$$

T_{depth} can be measured from a reference user's hand, but we experimentally noted that an empirical threshold of $T_{depth} = 10cm$ is acceptable in most cases (we used this value for the experimental results). In order to remove also most of the retained arm samples, we perform a further check on $\mathcal{H}$, namely we remove each $\mathbf{X}_{u,v} \in \mathcal{H}$ that has a distance in the 3D space from $\mathbf{X}_{min}$ larger than a threshold T_{size} that also depends on the hand size (for the experiments we set $T_{size} = 20cm$). Note how T_{depth} and T_{size} only depend on the physical hand size and not on the hand position or the sensor resolution.

The proposed algorithm allows to reliably segment the hand samples from the scene objects and from the other body parts. An example of a thresholded depth map obtained with our approach is shown in Fig. 2c. Now, in order to extract the feature sets described in Section 2 it is necessary to detect

the palm region. A 2D binary mask $B(u, v)$ is built on the lattice (u, v) associated to the acquired depth map in the following way:

$$B(u,v) = \begin{cases} 1 & \text{if } \mathbf{X}_{u,v} \in \mathcal{H} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

i.e., the entries of $B(u, v)$ are non-zero for the indexes corresponding to the samples in $\mathcal{H}$.

Our palm detection approach consists in estimating the largest circle that can be fitted on the palm region in $B(u, v)$. For this purpose, it is first necessary to find a good starting point $\mathbf{C}$ for the circle fitting algorithm. In order to select point $\mathbf{C}$ we exploit the fact that the palm region in B has the highest point density, since usually the palm area is larger than the fingers and the wrist. We filter $B(u, v)$ with a 2D Gaussian kernel with a very large standard deviation. We used $\sigma = 150 \cdot \frac{1[m]}{D_{min}}$. Note that the value of σ is scaled according to the minimum distance in order to make the window size in metric units invariant to the hand distance from the Kinect$^{\text{TM}}$and ensure that the support of the filter is always large enough to capture the thickness of the hand or arm regions. The Gaussian filter output consists in a blurred grayscale image $B^f(u, v)$ with values proportional to points density (see Fig. 2d). We set $\mathbf{C} = \mathbf{C_g}$, where $\mathbf{C_g}$ is the point of $B^f(u, v)$ that has the maximum gray level value (i.e., density). In some unlucky cases $\mathbf{C_g}$ may not lie near the palm center, but somewhere in the arm region if the arm points density is higher than the hand ones. Note also that there may also be multiple points with the maximum density. In order to avoid these situations and deliver a suitable position for $\mathbf{C_g}$ we perform a further thresholding on $B^f(u, v)$. Let us denote with $b_{max} = \max_{u,v}(B^f(u, v))$ the maximum computed density and

with $T_d \in [0,1]$ a threshold value (in our experiments we set $T_d = 0.9$, i.e., $T_d \cdot b_{max}$ correspond to 90% of the maximum density). A new 2D binary mask $B^T(u,v)$ is computed:

$$B^T(u,v) = \begin{cases} 1 & \text{if } B^f(u,v) \geq T_d \cdot b_{max} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$B^T(u,v)$ contains one or more blobs representing possible candidates to contain $\mathbf{C_g}$. We compute each blob centroid and we eventually choose as $\mathbf{C_g}$ the centroid of the *nearest* blob to X_{min} defined above.

The circle fitting procedure is the following: a circle with initial center position $\mathbf{C} = \mathbf{C_g}$ and radius $r = 1[pxl]$ is first expanded in $B(u,v)$ by increasing r until the 95% of the points inside it belong to $\mathcal{H}$ (we left a tolerance of 5% to account for errors due to noise or artefacts of the depth sensor). After the maximum radius value satisfying the threshold is found, $\mathbf{C}$ is shifted towards the direction that maximizes the density of the samples inside $\mathcal{H}$ contained in the circle. The radius r is then increased again, and we continue to iterate the two phases until the largest possible circle has been fitted on the palm area (Fig. 2e). The final position of $\mathbf{C}$, denoted by $\mathbf{C_f}$ corresponds to the center of the palm. The corresponding 3D point $\mathbf{C_f}$, that from now on we will call the *centroid* of the hand, will play an important role in the proposed algorithm together with the final radius value r_f. Furthermore the position of the centroid is also useful in order to reconstruct the trajectory followed by the hand in dynamic gestures, that is very useful in many applications (e.g., for the control of virtual mouses or of browsing of 3D scenes) and is one of the key points for the recognition of dynamic gestures.

Sometimes the circle does not accurately correspond to the palm area, mostly because the shape of the palm can be narrow and long and because in many acquired gestures the hand is not parallel to the imaging plane and the circular shape gets distorted by the perspective projection. In order to deal with these issues we also introduced a more accurate model where an ellipse is fit to the palm region. We start from $\mathbf{C_f}$ and build 12 regions corresponding to different partially superimposed angular directions (we used an overlap of 50% between each sector and the next one as shown in Fig. 2g) and for each region we select the point of the hand contour inside the region that is closest to the center. In this way we get a polygon contained inside the hand contour that approximates the hand palm. The choice of using partially superimposed sectors and to take the minimum distance inside each sector ensures that the polygon corners are chosen at the basis of the fingers and the finger samples are not included in the polygon. Finally the ellipse that better approximates the polygon in the least-square sense is computed using the method from Fitzgibbon and Fisher (1995) (Fig. 2h).

Once all the possible palm samples have been detected, we fit a 3D plane π on them by using SVD and RANSAC. Then Principal Component Analysis (PCA) is applied to the 3D points in $\mathcal{H}$ in order to extract the main axis that roughly corresponds to the direction $\mathbf{i_x}$ of the vector going from the wrist to the fingertips. Note that the direction computed in this way is not very precise and depends on the position of the fingers in the performed gesture. It gives, however, a general indication of the hand orientation. In order to build a 3D coordinate system centred on the point $\mathbf{C_f}$ previously defined, the axis $\mathbf{i_x}$ is then projected on plane π. Let us denote by $\mathbf{i_x^\pi}$ this projection, and by $\mathbf{i_z^\pi}$

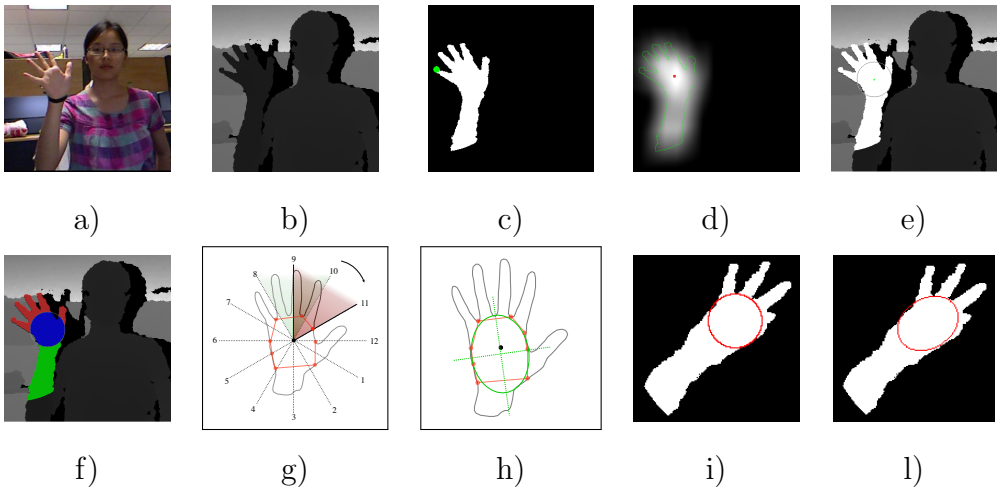


Figure 2: Extraction of the hand and palm samples: a) Acquired color image; b) Acquired depth map; c) Extracted hand samples (the closest sample is depicted in green); d) Output of the Gaussian filter applied on the mask corresponding to $\mathcal{H}$ with the maximum (i.e., $\mathbf{C_g}$) in red; e) Circle fitted on the hand with the point $\mathbf{C_p}$ in green; f) Palm (blue), finger (red) and wrist (green) regions subdivision; g) Angular sectors used for the computation of the ellipse; h) Fitting of the ellipse over the palm; i,l) Comparison of the circle and ellipse fitting on the same sample gesture. *(Best viewed in colors)*

the normal to plane π; note that $\mathbf{i}_\mathbf{x}^\pi$ and $\mathbf{i}_\mathbf{z}^\pi$ are orthogonal by definition. The missing axis $\mathbf{i}_\mathbf{y}^\pi$ is obtained by the cross-product of $\mathbf{i}_\mathbf{z}^\pi$ and $\mathbf{i}_\mathbf{x}^\pi$ thus forming a right-handed reference system $(\mathbf{i}_\mathbf{x}^\pi, \mathbf{i}_\mathbf{y}^\pi, \mathbf{i}_\mathbf{z}^\pi)$. The points coordinates in this reference system will be denoted with (x_{2D}, y_{2D}, z_{2D}).

Note also that $\mathbf{C_f}$ does not necessary lie on π (e.g. it could lie on a finger folded over the palm). In order to place $\mathbf{C_f}$ closer to the real hand center, we project it on π. Let us denote the corrected centroid by $\mathbf{C_p}$. The proposed coordinate system is depicted in Fig. 3.

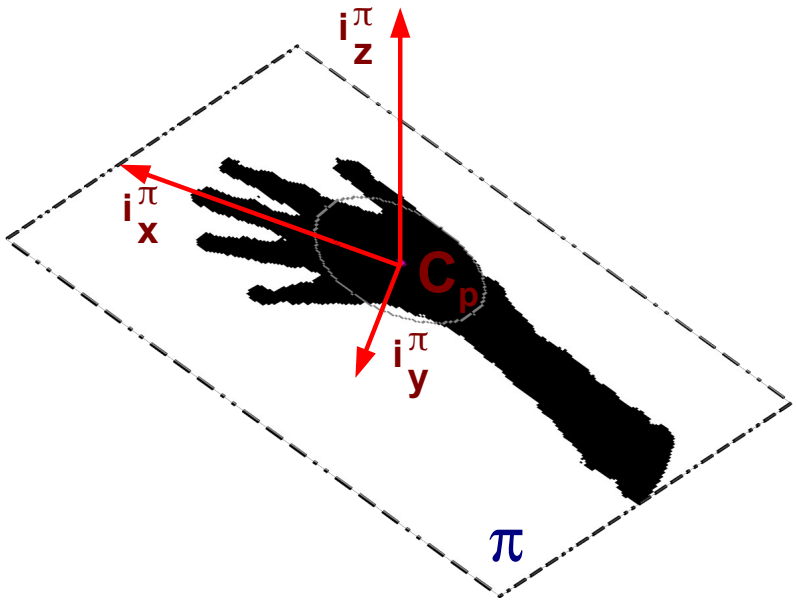


Figure 3: Reference system $(\mathbf{i}_\mathbf{x}^\pi, \mathbf{i}_\mathbf{y}^\pi, \mathbf{i}_\mathbf{z}^\pi)$ computed on the basis of the estimated plane and of the PCA output, used for the features extraction.

At this point, we have all the information required to segment $\mathcal{H}$ into three regions:

- $\mathcal{P}$ containing points corresponding to the hand palm (the samples inside

the circle or ellipse).

- $\mathcal{W}$ containing the points of $\mathcal{H}$ lying on the sub-space $x_{2D} \leq -r_f$. Such samples belong to the wrist and forearm, and will be discarded next.

- $\mathcal{F}$ containing the points of $\mathcal{H} - \mathcal{P} - \mathcal{W}$, which correspond to the fingers region.

Finally, the set $\mathcal{H}_e = (\mathcal{H} - \mathcal{W}) = (\mathcal{P} + \mathcal{F})$ containing the hand palm and fingers points is also computed. At this point all the information needed by the proposed feature extraction scheme is available.

4. Extraction of the relevant features

4.1. Distance features

The computation of this feature set starts from the construction of a histogram representing the distance of the edge samples in $\mathcal{F}$ from the hand centroid $\mathbf{C}_p$ (note that the proposed scheme considers only finger edges, differently from other schemes like Ren et al. (2011b)).

Let R_f be the 3D radius r_f back-projected to the plane π. Note that if the more accurate fitting model with the ellipse is employed R_f represents the distance from C_f to the edge of the ellipse and is not a constant value. For each 3D point $\mathbf{X_i} = \mathbf{X}_{u,v} \in \mathcal{F}$ in the fingers set, we compute its normalized distance from the centroid $d_{\mathbf{X_i}} = \|\mathbf{X_i} - \mathbf{C_p}\| - R_f, \mathbf{X_i} \in \mathcal{F}$ and the angle θ_{X_i} between vector $\mathbf{X}_\mathbf{i}^\pi - \mathbf{C_p}$ and axis $\mathbf{i}_\mathbf{x}^\pi$ on the palm plane π, where $\mathbf{X}_\mathbf{i}^\pi$ is the projection of $\mathbf{X_i}$ on π. We then quantize θ with a uniform quantization step Δ (in the current implementation we used $\Delta = 2°$) into a discrete set of values θ_q. Each θ_q value thus corresponds to an angular sector $\mathcal{I}(\theta_q) =$

$\theta_q - \frac{\Delta}{2} < \theta \leq \theta_q + \frac{\Delta}{2}$. We then select the farthest point inside each sector $\mathcal{I}(\theta_q)$, thus producing a histogram $L(\theta)$:

$$L(\theta_q) = \max_{\mathcal{I}(\theta_q)} d_{\mathbf{X_i}} \tag{4}$$

For each gesture in the database we build a reference histogram $L_g^r(\theta)$ of the type shown in Fig. 4. We also define a set of angular regions corresponding to the raised fingers intervals in each gesture (shown in Fig. 4) that will be used for computing the features.

As pointed out in Section 2, the direction of the PCA main axes is not very precise and furthermore is affected by several issues, e.g., the number of raised fingers in the performed gesture and the size of the retained wrist region after hand detection. The generated distance histogram may, then, not be precisely aligned with the gesture templates, and a direct comparison of the histograms in this case is not possible.

For this reason, in order to compare the performed gesture histogram with each gesture template we first align them by looking for the argument maximizing the cross-correlation between the acquired histogram and the translated version of the reference histogram of each gesture[2]. We also consider the possibility of flipping the histogram to account for the fact that the hand could have either the palm or the dorsum facing the camera, evaluating:

$$\begin{aligned} \Delta_g &= \operatorname*{argmax}_{\Delta} \left(\rho\left(L(\theta), L_g^r(\theta + \Delta)\right)\right) \\ \Delta_g^{rev} &= \operatorname*{argmax}_{\Delta} \left(\rho\left(L(-\theta), L_g^r(\theta + \Delta)\right)\right) \end{aligned} \tag{5}$$

where symbol $\rho(a(\cdot), b(\cdot))$ denotes the value of the cross correlation between

[2]In Equations (5) and (6) L is considered as a periodic function with period 2π.

$a(\cdot)$ and $b(\cdot)$. This gives us the translational shift Δ that aligns the acquired histogram with the reference histograms of each gesture. Let us denote by $L_g(\theta)$ the histogram aligned to the gesture reference histogram $L_g^r(\theta)$. The translational shift to be applied to $L(\theta)$ will be either Δ_g and Δ_g^{rev} depending on the one maximizing the correlation, i.e. we define $L_g(\theta)$ as:

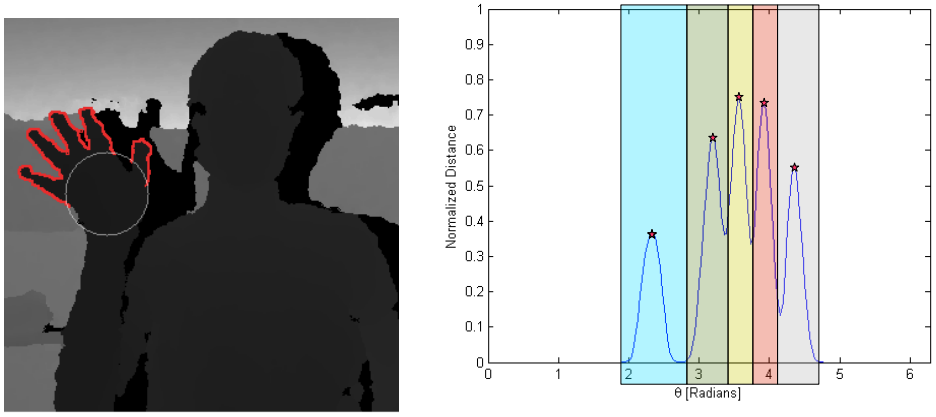
$$L_g(\theta) = \begin{cases} L(\theta - \Delta_g) & \text{if } \max_{\Delta} \rho\left(L(\theta), L_g^r(\theta + \Delta)\right) \geq \max_{\Delta} \rho\left(L(-\theta), L_g^r(\theta + \Delta)\right) \\ L(-\theta - \Delta_g^{rev}) & \text{otherwise} \end{cases} \tag{6}$$

Note that there can be a different alignment Δ_g for each gesture, and that we can define different regions in each gesture reference histogram corresponding to the various features of interest. This approach basically compensates for the limited accuracy of the direction computed by the PCA in Section 2.

The alignment procedure solves one of the main issues related to the direct application of the approach of Ren et al. (2011b). Fig. 5 shows some examples of the computed histograms for three different gestures. Note that the fingers raised in the various gestures are clearly visible from the plots.

If the database has G different gestures to be recognized, the feature set $\mathcal{F}^l$ contains a value for each finger $j \in \{1, .., 5\}$ in each gesture $g \in \{1, .., G\}$. The feature value $f_{g,j}^l$ associated to finger j in gesture g corresponds to the maximum of the aligned histogram in the angular region $\mathcal{I}(\theta_{g,j}) = \theta_{g,j}^{min} < \theta < \theta_{g,j}^{max}$ associated to finger j in gesture g (see Fig. 4), i.e. :

$$f_{g,j}^l = \frac{\max_{\mathcal{I}(\theta_{g,j})} L_g(\theta)}{L_{max}} \tag{7}$$



a) b)

Figure 4: Histogram of the edge distances with the corresponding feature regions: a) finger edges $\mathcal{F}$; b) associated histogram $L(\theta)$ with the regions corresponding to the different features $f^l_{g,j}$ (feature points highlighted with red stars).

All the features are normalized by the length L_{max} of the middle finger in order to scale them within range $[0, 1]$ and account for the fact that the hands of different people have different size. Note that there can be up to $G \times 5$ features, though their actual number is smaller since not all the fingers are raised in each gesture (e.g., in the experimental results dataset there are 10 different gestures and we used 24 features). The distance features are collected into feature vector $\mathbf{F^l}$.

4.2. Elevation features

The construction of the elevation features is analogous to the one employed for the distance features of Section 4.1.

We start by building an histogram representing the distance of each sample in $\mathcal{F}$ from the palm plane π, namely, for each sample $\mathbf{X_j}$ in $\mathcal{F}$ we compute

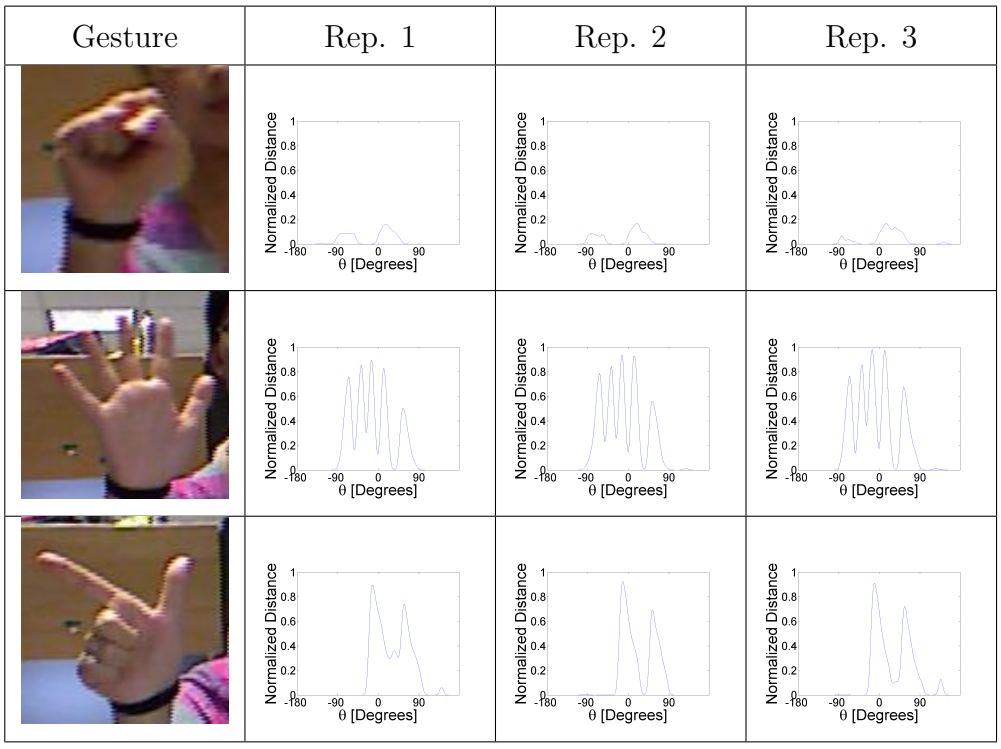


Figure 5: Examples of aligned distance histogram $L_g(\theta)$ for 3 sample frames corresponding to different gestures.

its distance from plane π:

$$e_{\mathbf{X_j}} = sgn\left((\mathbf{X_j} - \mathbf{X_j^\pi}) \cdot \mathbf{i_y^\pi}\right) |\mathbf{X_j} - \mathbf{X_j^\pi}|, \quad \mathbf{X_j} \in \mathcal{F} \tag{8}$$

where $\mathbf{X_j^\pi}$ is the projection of $\mathbf{X_j}$ on π. The sign of $e_{\mathbf{X_j}}$ accounts for the fact that $\mathbf{X_j}$ can belong to any of the two hemi-spaces defined by π, i.e., $\mathbf{X_j}$ can either be on the front or behind π.

Now, as we did for the distance features, for each angular sector corresponding to a θ_q value we select the point with greatest absolute distance from the plane, thus producing an histogram $E(\theta)$:

$$E(\theta_q) = \begin{cases} \max\limits_{\mathcal{I}(\theta_q)} e_{\mathbf{X_j}} & \text{if } \left|\max\limits_{\mathcal{I}(\theta_q)} e_{\mathbf{X_j}}\right| > \left|\min\limits_{\mathcal{I}(\theta_q)} e_{\mathbf{X_j}}\right| \\ \min\limits_{\mathcal{I}(\theta_q)} e_{\mathbf{X_j}} & \text{otherwise} \end{cases} \tag{9}$$

Histogram $E(\theta_q)$ uses the same regions computed in Section 4.1. The histogram $E(\theta)$ corresponding to the performed gesture is then aligned to the various reference gestures in G using the alignment information already computed in Section 4.1. Let $E^g(\theta)$ be histogram $E(\theta)$ aligned with the g^{th} gesture template. The elevation features are then computed according to:

$$f^e_{g,j} = \begin{cases} \dfrac{1}{L_{max}} \max\limits_{\mathcal{I}(\theta_{g,j})} E^g(\theta) & \text{if } \left|\max\limits_{\mathcal{I}(\theta_{g,j})} E^g(\theta)\right| > \left|\min\limits_{\mathcal{I}(\theta_{g,j})} E^g(\theta)\right| \\ \dfrac{1}{L_{max}} \min\limits_{\mathcal{I}(\theta_{g,j})} E^g(\theta) & \text{otherwise} \end{cases} \tag{10}$$

Note that in our approach the alignments computed in Section 4.1 are used here both to save computation time and because the correlations from distance data are more reliable than the ones computed on elevation information. Finally note that the vector $\mathbf{F^e}$ of the elevation features has the same structure and number of elements of the vector $\mathbf{F^l}$ of the distance features.

4.3. Curvature features

The third proposed descriptor is based on the curvature of the hand shape edges. Since depth data coming from real-time depth cameras are usually rather noisy we decided to avoid differential operators for curvature description relying, instead, on integral invariants (Manay et al., 2006; Kumar et al., 2012).

Our feature extractor algorithm takes as input the hand edge points $\mathcal{H}_e$ and the binary mask $B(u, v)$. Let us denote by $\mathcal{H}_c = \partial\mathcal{H}_e$ the boundary of $\mathcal{H}_e$, namely the subset of all the points $\mathbf{X_i} \in \mathcal{H}_e$ belonging to the hand contour only. Consider a set of S circular masks $M_s(\mathbf{X_i}),\ s = 1, .., S$ with radius r_s centred on each edge sample $\mathbf{X_i} \in \mathcal{H}_c$. In our experiments we used 25 masks with r_s varying from $0.5cm$ to $5cm$.

Let $V(\mathbf{X_i}, s)$ denote the curvature in $\mathbf{X_i}$, expressed as the ratio of the number of samples of $\mathcal{H}_e$ falling in the mask $M_s(\mathbf{X_i})$ over $M_s(\mathbf{X_i})$ size, namely:

$$V(\mathbf{X_i}, s) = \frac{\sum_{\mathbf{X_j} \in M_s(\mathbf{X_i})} B(\mathbf{X_j})}{|M_s(\mathbf{X_i})|} \tag{11}$$

where $|M_s(\mathbf{X_i})|$ denotes the cardinality of $M_s(\mathbf{X_i})$. $B(\mathbf{X_j}) = B(u_j, v_j)$, where (u_j, v_j) are the 2D coordinates corresponding to $\mathbf{X_j}$. Note that $V(\mathbf{X_i}, s)$ is computed for each sample $\mathbf{X_i} \in \mathcal{H}_c$. The radius r_s value corresponds, instead, to the scale level at which feature extraction is performed. Differently from Kumar et al. (2012) and other approaches, the radius r_s is defined in metrical units and is then converted to the corresponding pixel size on the basis of the distance between the camera and the hand. In this way the descriptor is invariant with respect to the distance between the hand and the camera.

Curvature masks are rotation invariant but for faster processing we also included the option of replacing the circular masks with simpler square masks and then using integral images for fast computation of the samples in the mask. This approach, even if not perfectly rotation invariant, proved to be significantly faster and the performance loss is practically unnoticeable.

The values of $V(\mathbf{X_i}, s)$ range from 0 (extremely convex shape) to 1 (extremely concave shape), with $V(\mathbf{X_i}, s) = 0.5$ corresponding to a straight edge. We quantized the $[0, 1]$ interval into N bins of equal size $b_1, .., b_N$. The set $\mathcal{V}_{b,s}$ of the finger edge points $\mathbf{X_i} \in \mathcal{H}_c$ with the corresponding value of $V(\mathbf{X_i}, s)$ falling to bin b for the mask s is expressed as:

$$\mathcal{V}_{b,s} = \{\mathbf{X_i} | \frac{(b-1)}{B} < V(\mathbf{X_i}, s) \leq \frac{b}{B}\} \tag{12}$$

For each radius value s and for each bin b we choose as curvature feature, denoted by $f^c_{b,s}$, the cardinality of the set $V(\mathbf{X_i}, s)$ normalized by the contour length $|\mathcal{H}_c|$, i.e.:

$$f^c_{b,s} = \frac{|\mathcal{V}_{b,s}|}{|\mathcal{H}_c|} \tag{13}$$

Note that, thanks to the normalization, the curvature feature $f^c_{b,s}$ takes values in $[0, 1]$, that is, the same interval shared by both the distances and elevations feature. Finally, we collect all the curvature features $f^c_{b,s}$ within feature vector $\mathbf{F^c}$ with $B \times S$ entries, ordered by increasing values of indexes $s = 1, 2, ...S$ and $b = 1, 2, ..., N$. By resizing $\mathbf{F^c}$ into a matrix with S rows and N columns, and by considering each $f^c_{b,s}$ as the value of the pixel with coordinates (b, s) in a grayscale image, it is possible to graphically visualize the overall curvature descriptor $\mathbf{F^c}$ as exemplified in Fig. 6.

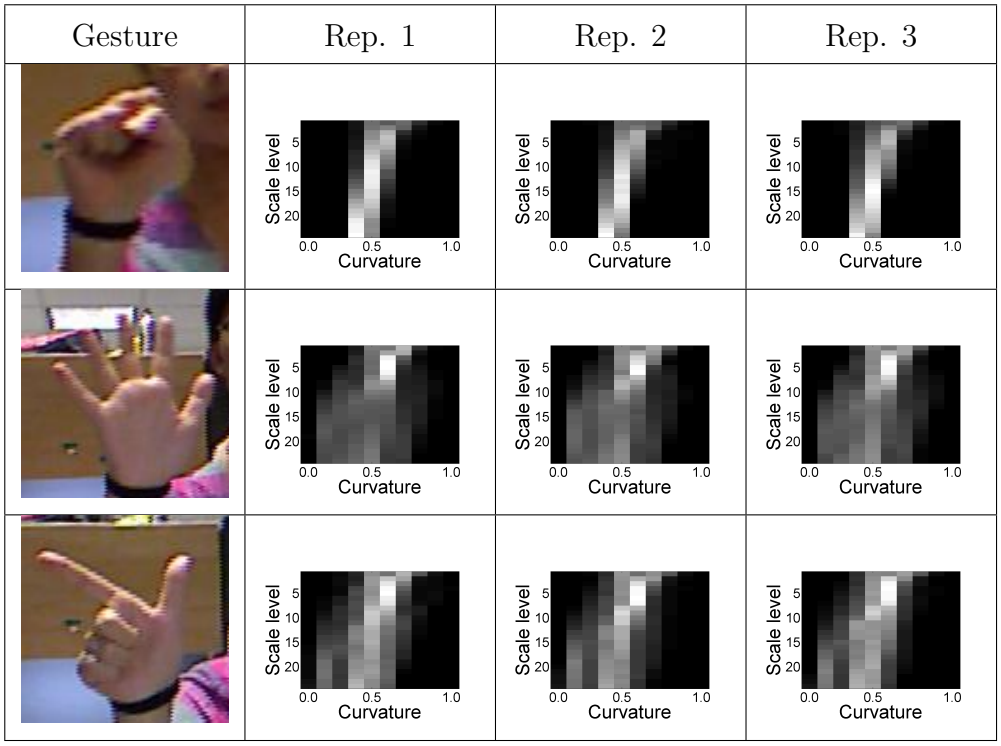


Figure 6: Examples of curvature descriptors for 3 sample frames from different gestures.

4.4. Palm area features

The last set of features describes the displacement of the samples in the palm region $\mathcal{P}$. Note that $\mathcal{P}$ corresponds to the palm area, but it may also include finger samples if some fingers are folded over the palm. The idea is to subdivide the palm region into six different areas, defined over the plane π, as shown in Fig. 7. The circle or ellipse defining the palm area is firstly divided into two parts: the lower half is used as a reference for the palm position, and a 3D plane π_p is firstly fitted to this region. The upper half is divided into 5 regions $\mathcal{A}_j, j = 1, .., 5$ roughly corresponding to the regions close to the different fingers as shown in Fig. 7, i.e., each region corresponds to the area that is affected by the position of a finger. The various area features account for the deformation the palm shape undergoes in the corresponding area when the related finger is folded or is moved. In particular notice how the samples corresponding to the fingers folded over the palm are associated to $\mathcal{P}$ and are not captured by distance or elevation features, but they are used for the computation of palm area features. The areas positions on the plane strictly depend on the parameters defining the palm area (i.e., the center $\mathbf{C}_f$ and the radius r_f of the circle or the two axes of the ellipse), the fingers widths (a standard subdivision of the upper half of the circle has been used but it can also be optimized on the basis of the specific user's hand) and on the direction $\mathbf{i}_\mathbf{x}^\pi$ corresponding to $\theta = 0$. Since the center $\mathbf{C_f}$ and radius r_f or axes have already been computed in Section 3, the only missing element is the alignment of the θ directions. Again, the alignment information computed in Section 4.1 is used to align the regions template (scaled by r_f, or scaled and stretched according to the two axes of the ellipse) with the hand direction $\mathbf{i}_\mathbf{x}^\pi$.

We perform an alignment for each gesture template, with the same approach used for the distance features, in order to extract an area feature set for each alignment. The areas aligned with the template of each gesture will be denoted with $\mathcal{A}_j^g$, where g denotes the corresponding gesture. In this way the set of points $\mathbf{X_i}$ in $\mathcal{P}$ associated to each of the regions $\mathcal{A}_j^g$ is computed. Then, each area $\mathcal{A}_j^g$ is considered and the distance between each sample $\mathbf{X_i}$ in $\mathcal{A}_j^g$ and π_p is computed. The average of the distances of the samples of the area A_j^g :

$$f_{g,j}^a = \frac{\sum_{\mathbf{X_i} \in \mathcal{A}_j^g} \|\mathbf{X_i} - \mathbf{X_i^\pi}\|}{|A_j^g|} \tag{14}$$

is taken as the feature corresponding to the area A_j^g. All the area features are collected within vector $\mathbf{F^a}$, made by $G \times 5$ area features, one for each finger in each possible gesture, following the same rationale of $\mathbf{F^l}$ and $\mathbf{F^e}$. The entries of $\mathbf{F^a}$ are finally scaled in order to assume values within range $[0, 1]$, as the other feature vectors.

Figure 7: Regions corresponding to the various area features shown over a sample gesture.

5. Gesture classification with Support Vector Machines

The feature extraction approach of Section 4 provides four feature vectors describing relevant properties of the hand samples. In order to recognize the gestures from the feature vectors built in Section 4, we employed a multi-class Support Vector Machine classifier. Each acquired gesture is described by a feature vector $\mathbf{F} = [\mathbf{F^l}, \mathbf{F^e}, \mathbf{F^c}, \mathbf{F^a}]$ obtained by concatenating the four different feature vectors $\mathbf{F^l}$, $\mathbf{F^e}$, $\mathbf{F^c}$ and $\mathbf{F^a}$. Note that $\mathbf{F^l}$, $\mathbf{F^e}$ and $\mathbf{F^a}$ represent features corresponding to the various possible hypotheses about the current gesture, while $\mathbf{F^c}$ basically contains the histograms of the curvature distribution for all the scale levels.

The gesture recognition problem consists in classifying the vectors $\mathbf{F}$ into G classes corresponding to the various gestures of the considered database. The employed classification algorithm is based on the *one-against-one* approach, i.e., a set of $G(G-1)/2$ binary SVM classifiers is used to test each class against each other and each output is chosen as a *vote* for a certain gesture. The gesture with the maximum number of votes is the result of the recognition process. In particular we used the SVM implementation in the LIBSVM package (Chang and Lin, 2011). We set a non-linear Gaussian Radial Basis Function (RBF) as the kernel and we tuned the classifier parameters by a grid search approach and cross-validation on the training set. Assume a training set containing data from N users : to perform the grid search we divided the space of parameters (C, γ) of the RBF kernel with a regular grid and for each couple of parameters the training set is divided into two parts, one containing $N-1$ users for training and the other the remaining user for validation and the performances are evaluated. We repeat the

procedure changing each time the user used for the validation and we select the couple of parameters that give the best accuracy on average. Finally we train the SVM on all the N users of the training set with the optimal parameters.

6. Experimental results

The performances of the proposed approach have been evaluated using two different datasets containing data acquired by Microsoft's Kinect (however the approach is independent of the employed depth camera). The first is the database provided by Ren et al. (2011b), containing 10 different gestures performed by 10 different people. Each gesture is repeated 10 times for a total of 1000 different depth maps with related color images. The second dataset is a sub-set of the American Sign Language gestures acquired in our laboratory (shown in Fig. 8 and available on our website). It contains 12 different gestures performed by 14 different people and repeated 10 times.

Since our approach requires a learning stage, we considered two different operational possibilities. In the first simpler approach (it will be denoted as *user training*) we randomly split the database into 2 parts, one is used to train the SVM classifier and the other made by the remaining depth maps was used as test set. More precisely the training set contains 8 randomly selected repetitions of each gesture by each person while the remaining 2 have been put in the test set. For each gesture, one of the repetitions in the training set was used for the computation of the reference histogram of Eq. (5). The complete training set was then used to train the different SVM classifiers. This subdivision of the database corresponds to having gestures from all the

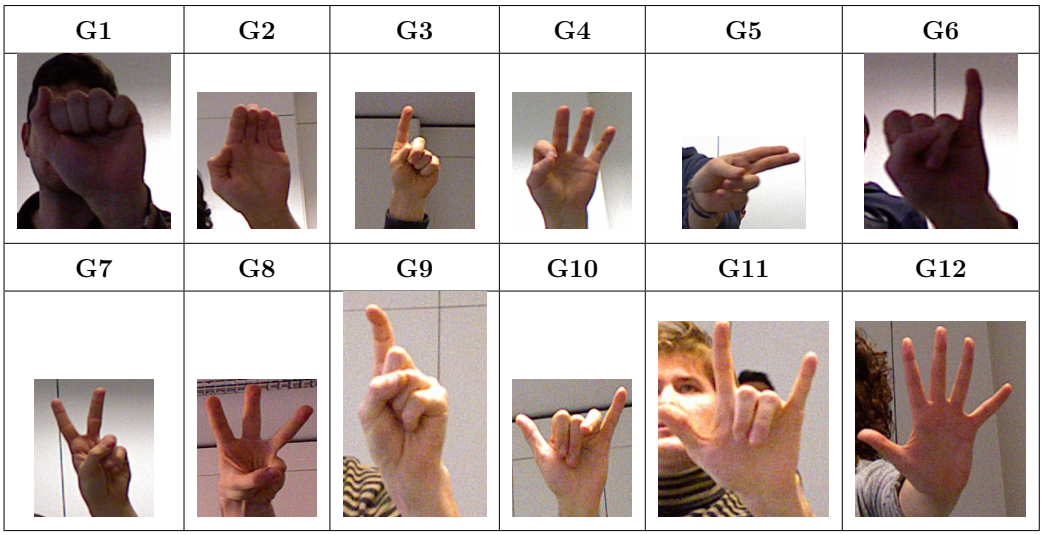


Figure 8: Gestures from the American Sign Language (ASL) contained in the database that has been acquired for the experimental results.

subjects in both the train and test sets, i.e., the people using the system had to "train" it before by performing the different gestures. Since in this approach data samples from the same person are present in both the train and test set it can be viewed as something similar to the concept of *validation* in classification literature. In many practical situations is necessary to have a system that is able to recognize the gestures performed by a new user without re-training the system with this user. Hence the training must be performed on a set of people different from the end users. For this reason, we performed a second more challenging set of tests by splitting the database in a training set made by $N - 2$ people (i.e., 8 people for the first dataset and 12 for the second), and a test set with the remaining two people. The training (it will be called *generic training*) has been hence performed with different people than the ones used for the testing. Since in this approach the test set contains

data from a person that has not trained the system there is less correlation between the test and train sets and the problem is more challenging (if the previous test can be considered as the *validation*, this correspond to the use of a *test set* unrelated to the training one).

The first column of Table 1 shows the results obtained on the first database with the *user training* approach. Distance features $\mathbf{F^l}$ alone provide an accuracy of about 96%. Note that distance descriptors are very good in capturing the fact that the various fingers are folded over the palm or raised, an important element in the recognition of many gestures. The curvature-based classifier allows to obtain even better performances (97.5%) by using the $\mathbf{F^c}$ feature vectors. In particular the distance only classifier is able to recognize some of the gestures that curvature only one can not handle, and vice-versa. Elevation features have lower performances on the first dataset ($85,5\%$). This is due to the fact that in most gestures in the dataset the fingers lay very close to the palm plane. They, however, play an important role in recognizing more complex gestures not included in this dataset, where some fingers point out of the palm plane. Finally, area based features allows to obtain an accuracy of $84,5\%$.

Better performances can be obtained by combining different classifiers together. For example, by combining distance and curvature features it is possible to obtain an almost optimal accuracy of 99.5%. This is because the two classifiers have complementary characteristics, since the two descriptors are based on totally different clues. By further adding the elevation and area features, it is possible to recognize all the performed gestures and obtain a 100% accuracy.

We repeated the same tests for the *generic training* case. The results are shown in the second column of Table 1. Distance based features $\mathbf{F^l}$ alone already allow to obtain very good performances with an accuracy of about $92,5\%$, even if, as expected, in this more challenging situation the accuracy is slightly lower than in the previous case. The curvature features have very similar performances (92%). Also in this case, elevation features are the least performing descriptor, since most gestures have the fingers very close to the hand plane; their accuracy is 43.5%. Better results can be obtained by using area based features, that allows to obtain an accuracy of 60%, lower than distance or curvature but able to distinguish the majority of the gestures.

By combining distance and curvature features, it is possible to reach an accuracy of 98.5%. These two descriptors are, again, very informative and also rather complementary, so their combination gets quite close to the optimum in this simple database. Although the performances of distance and curvature are better than the other two descriptors, note that each of the different descriptors captures different aspects of the hand pose that are relevant in different gestures. In order to obtain even better accuracy, it is hence necessary to combine multiple descriptors together. By further adding the area features, a small improvement in the accuracy can be obtained, rising it to 99%. Finally, by using all the 4 feature types the accuracy remains at 99%. The improvement obtained by adding the last two set of features on this database is rather limited, but consider that performances with distance and curvature data are already very close to the optimum.

The last three rows of Table 1 compare the results with the ones from Ren et al. (2011b). It is evident that the proposed recognition scheme outperforms

the compared approach: even the best performing version of the work of Ren et al. (2011b) has an accuracy of 94%, that corresponds to having 6 times more errors than the proposed approach. Furthermore, note that Ren et al. (2011b) exploits a black bracelet that all the people wear in order to locate and align the hand shapes, while our approach does not exploit this aid and does not require to wear any glove, bracelet or other sort of marker.

Table 1: Performance of our approach. The proposed approach is compared with (Ren et al., 2011b). The work of (Ren et al., 2011b) presents the results of two different versions, one using near-convex decompisition (FEMD-a) and one exploiting a thresholding decomposition (FEMD-b). Results for the compared method are available only for the first database since the software of (Ren et al., 2011b) is not publicy available.

Type of features	Database of (Ren et al., 2011b)		Our Database	
	Accuracy *users training*	Accuracy *generic training*	Accuracy *users training*	Accuracy *generic training*
Distance features	96,0 %	92,5 %	83,0 %	70,4 %
Elevation features	85,5 %	43,5 %	70,8	47,5 %
Curvature features	97,5 %	92 %	92,9 %	88,3 %
Area features	84,5 %	60 %	71,7 %	54,2 %
Dist.+curv.	99,5 %	98,5 %	95,0 %	89,6 %
Dist.+curv.+area	100 %	99 %	96,1 %	92,9 %
Dist.+curv.+elev.+area	100 %	99 %	97,6 %	93,8 %
FEMD-a(Ren et al., 2011b)	90.6%		N.A.	
FEMD-b(Ren et al., 2011b)	93.9%		N.A.	

The second database is more challenging, since it includes a larger number of gestures, which are also more complex and more difficult to distinguish. Recall that distance descriptors are able to distinguish most of the gestures on the first database, while on the second one they reach an accuracy of $83,0\%$ with the *users training* and $70,4\%$ with the *generic training*. The lower performances are due to the presence of different gestures with the same number of raised fingers. Also consider that, while in the other database the hands were all acquired in very ideal conditions (e.g., same distance, hand almost perpendicular to the camera, people with similar hands), here a more realistic setting has been used with a more limited control on the position and orientation of the hand, and the people have hands with very different characteristics. Curvature descriptors are the best descriptor on this database, with an accuracy of 92.2% and $88,3\%$ for the two types of training respectively. Note that curvatures do not rely on the computation of the hand orientation or on the positions of the centroid and palm plane. For this reason, is more performing in complex configurations where the estimation of these parameters is not always highly accurate. Elevation features allow to obtain an accuracy of $70,8\%$ if the users are involved in the training, while in the other case accuracy drops to $47,5\%$. Finally, area features have an accuracy of $71,7\%$ (*users training*) and $54,2\%$ (*generic training*), slightly better than the elevation features.

With the *users training*, by combining distance and curvature features the accuracy is $95,0\%$. Note how distance features have lower performances but they are able to give an improvement to the results of curvature features alone. A further improvement can be obtained by adding also area features,

raising up the accuracy to $96,0\%$. Finally, by including all the 4 types of feature, an accuracy of $97,6\%$ can be obtained, better than the ones of the various subset of features.

When the users are not involved in the training the performances are lower but by combining multiple descriptors they dramatically improve. With distance and curvature features together the accuracy is $89,6\%$, by adding also area features it raises up to $92,9\%$,and finally by including all the 4 types of feature an accuracy of $93,8\%$ can be obtained.

In order to allow for a more accurate analysis, the confusion matrix for the recognition with all the 4 types of features on the second dataset is shown in Fig. 9 while a larger set of confusion matrices is available at `http://lttm.dei.unipd.it/paper_data/gesture`. Note that the proposed scheme can also be used to reliably analyze the pose and trajectory of the hand in dynamic environments, some sample videos are available at `http://lttm.dei.unipd.it/paper_data/gesture`.

The proposed approach does not require complex computations and is able to run in real-time. In particular the current implementation (that has not been fully optimized) is able to achieve about $10fps$. From a computational point of view the most demanding steps are in the initial detection phase (i.e., the hand detection takes about $46ms$ and the extraction of palm and fingers regions about $25ms$). The computation of the palm plane takes about $4ms$. Feature extraction takes about $38ms$, mostly spent on the curvature descriptors ($28ms$). The other demanding computation is area descriptors that require about $10ms$ while distance and elevation features require a negligible computation time. Finally SVM classification uses $1ms$ for a total

running time of $114ms$ for each frame.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12
G1	**20/20** (26/28)	0/20 (1/28)	0/20 (1/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G2	0 (0/28)	**20/20** (**28/28**)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G3	0/20 (0/28)	0/20 (0/28)	18/20 (**1**)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	2/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G4	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	**20/20** (**1**)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G5	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	**20/20** (**1**)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G6	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (1/28)	**20/20** (27/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G7	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	1/20 (0/28)	0/20 (0/28)	0/20 (0/28)	16/20 (**1**)	3/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G8	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (1/28)	**20/20** (27/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G9	0/20 (0/28)	0/20 (0/28)	3/20 (2/28)	0/20 (0/28)	0/20 (0/28)	0/20 (2/28)	0/20 (0/28)	0/20 (0/28)	17/20 (24/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)
G10	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	2/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	18/20 (**1**)	0/20 (0/28)	0/20 (0/28)
G11	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	2/20 (0/28)	0/20 (0/28)	0/20 (0/28)	2/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	16/20 (**1**)	0/20 (0/28)
G12	0/20 (0/28)	0/20 (0/28)	0 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	0/20 (0/28)	**1** (**1**)

Figure 9: Confusion matrix for the proposed approach on our database with joint usage of all the 4 proposed feature types. Each entry contains both the output of the classifier for the generic training case (validation) and the output of the classifier for the training with users case (testing, between parenthesis).

7. Conclusions

This paper shows an effective way of exploiting depth information for hand gesture recognition, with a limited and not always required color in-

formation aid for hand identification only. It is worth noting how the palm and finger regions can be reliably extracted from depth data. Our approach remarkably does not require any manual segmentation or aid by bracelets, gloves, markers or other invasive tools.

The main idea of this paper is the usage of different features extracted from depth data capturing relevant and complementary properties of the hand gestures. The proposed features are the distances of the fingers from the hand centroid, the elevation of the fingers from the palm, the curvature of the hand shape and the planarity of the palm area. Each of the employed features is able to supply for the lack of information suffered by the remaining features for certain gestures. Although some kind of features alone allow for reasonable hand gesture recognition performances, the experimental results reported in Table 1 show that their combined usage lead to an higher accuracy.

Further research will be devoted to the introduction of new features into the proposed approach in order to better represent the fingers when they are folded. Also the introduction of color-based features will be considered. Since many gestures are characterized by a dynamic time evolution and the proposed approach is already able to follow the trajectory and orientation of the hand over time, we are planning to extend the proposed approach from the analysis of single frames to the analysis of video sequences considering also time-dependent features.

References

Ballan, L., Taneja, A., Gall, J., Gool, L.V., Pollefeys, M., 2012. Motion capture of hands in action using discriminative salient points, in: Proc. of the European Conference on Computer Vision (ECCV), Firenze.

Biswas, K., Basu, S., 2011. Gesture recognition using microsoft kinect, in: Automation, Robotics and Applications (ICARA), 2011 5th International Conference on, pp. 100 –103.

Chang, C.C., Lin, C.J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27.

Doliotis, P., Stefan, A., McMurrough, C., Eckhard, D., Athitsos, V., 2011. Comparing gesture recognition accuracy using color and depth information, in: Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments(PETRA'11), pp. 20:1–20:7.

Fitzgibbon, A., Fisher, R.B., 1995. A buyer's guide to conic fitting, in: In British Machine Vision Conference, pp. 513–522.

Garg, P., Aggarwal, N., Sofat, S., 2009. Vision based hand gesture recognition. World Academy of Science, Engineering and Technology 49, 972–977.

Herrera, D., Kannala, J., Heikkilä, J., 2012. Joint depth and color camera calibration with distortion correction. IEEE Trans. Pattern Anal. Mach. Intell. 34, 2058–2064.

623 Keskin, C., Kirac, F., Kara, Y., Akarun, L., 2011. Real time hand pose
624 estimation using depth sensors, in: ICCV Workshops, pp. 1228 –1234.

625 Keskin, C., Kıraç, F., Kara, Y.E., Akarun, L., 2012. Hand pose estima-
626 tion and hand shape classification using multi-layered randomized deci-
627 sion forests, in: Proc. of the European Conference on Computer Vision
628 (ECCV), pp. 852–863.

629 Kumar, N., Belhumeur, P.N., Biswas, A., Jacobs, D.W., Kress, W.J., Lopez,
630 I., Soares, J., 2012. Leafsnap: A computer vision system for automatic
631 plant species identification, in: Proc. of the European Conference on Com-
632 puter Vision (ECCV).

633 Kurakin, A., Zhang, Z., Liu, Z., 2012. A real-time system for dynamic hand
634 gesture recognition with a depth sensor, in: Proc. of EUSIPCO.

635 Li, Y., 2012. Hand gesture recognition using kinect, in: Software Engineering
636 and Service Science (ICSESS), 2012 IEEE 3rd International Conference on,
637 pp. 196 –199.

638 Manay, S., Cremers, D., Hong, B.W., Yezzi, A., Soatto, S., 2006. Integral
639 invariants for shape matching. IEEE Transactions on Pattern Analysis and
640 Machine Intelligence 28, 1602 –1618.

641 Oikonomidis, I., Kyriazis, N., Argyros, A., 2011. Efficient model-based 3d
642 tracking of hand articulations using kinect, in: Proceedings of the 22nd
643 British Machine Vision Conference (BMVC 2011).

644 Pedersoli, F., Adami, N., Benini, S., Leonardi, R., 2012. Xkin - extendable
645 hand pose and gesture recognition library for kinect, in: In: Proceedings of

ACM Conference on Multimedia 2012 - Open Source Competition, Nara, Japan.

Pugeault, N., Bowden, R., 2011. Spelling it out: Real-time asl fingerspelling recognition, in: Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, pp. 1114–1119.

Ren, Z., Meng, J., Yuan, J., 2011a. Depth camera based hand gesture recognition and its applications in human-computer-interaction, in: Proc. of Int. conference on Information, Communications and Signal Processing (ICICS), pp. 1 –5.

Ren, Z., Yuan, J., Zhang, Z., 2011b. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera, in: Proc. of the 19th ACM international conference on Multimedia, ACM, New York, NY, USA. pp. 1093–1096.

Suryanarayan, P., Subramanian, A., Mandalapu, D., 2010. Dynamic hand pose recognition using depth data, in: Proc. of Int. Conference on Pattern Recognition (ICPR), pp. 3105 –3108.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, IEEE. pp. I–511.

Wachs, J.P., Kölsch, M., Stern, H., Edan, Y., 2011. Vision-based hand-gesture applications. Commun. ACM 54, 60–71.

668 Wan, T., Wang, Y., Li, J., 2012. Hand gesture recognition system using
669 depth data, in: Consumer Electronics, Communications and Networks
670 (CECNet), 2012 2nd International Conference on, pp. 1063 –1066.

671 Wang, J., Liu, Z., Chorowski, J., Chen, Z., Wu, Y., 2012. Robust 3d action
672 recognition with random occupancy patterns, in: Proc. of the European
673 Conference on Computer Vision (ECCV).

674 Wen, Y., Hu, C., Yu, G., Wang, C., 2012. A robust method of detecting
675 hand gestures using depth sensors, in: Haptic Audio Visual Environments
676 and Games (HAVE), 2012 IEEE International Workshop on, pp. 72–77.

677 Zabulis, X., Baltzakis, H., Argyros, A., 2009. Vision-based hand gesture
678 recognition for human computer interaction, in: The Universal Access
679 Handbook. Lawrence Erlbaum Associates, Inc. (LEA). Human Factors and
680 Ergonomics. chapter 34, pp. 34.1 – 34.30.