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

Head-Mounted Gesture Controlled Interface for Human-Computer Interaction

Alvise Memo · Pietro Zanuttigh

Received: 8 April 2016 / Accepted: date

Abstract This paper proposes a novel human-computer interaction system exploiting gesture recognition. It is based on the combined usage of an head-mounted display and a multi-modal sensor setup including also a depth camera. The depth information is used both to seamlessly include augmented reality elements into the real world and as input for a novel gesture-based interface. Reliable gesture recognition is obtained through a real-time algorithm exploiting novel feature descriptors arranged in a multi-dimensional structure fed to an SVM classifier. The system has been tested with various augmented reality applications including an innovative human-computer interaction scheme where virtual windows can be arranged into the real world observed by the user.

Keywords Head mounted display · gesture recognition · human-computer interface · augmented reality · depth data.

1 Introduction

Human-computer interaction has been based on the traditional interface with monitor, keyboard and mouse for a long time. Various virtual reality schemes have been proposed in order to replace this model, but being excluded from the real world to enter the virtual environment is often a huge limitation. Furthermore gloves and other input devices exploited for this task have proved to be cumbersome to use. This work proposes an innovative solution for the

This research is part of the *Robotic 3D video* and *3D Cloud Vision* projects of the University of Padova.

All authors
Department of Information Engineering, University of Padova, Padova, Italy
Tel.: +39-049-8277774
Fax: +39-049-8277699
E-mail: alvise.memo@gmail.com, zanuttigh@dei.unipd.it

interaction with the computer allowing a natural interaction with the machine without being excluded from the real world.

From the visualization point of view this is solved by using an augmented reality system capable to insert the computer interface elements in the representation of the real 3D world in front of the user. This challenging target is achieved by combining the data acquired by a standard video camera with the 3D information from a Time-of-Flight (ToF) sensor. The calibration of the devices and the 3D information coming from the ToF sensor allows to properly insert the virtual elements into the real world as if they were objects in the 3D space and not just graphics elements overlaid over the video stream. This approach combined with the usage of a stereoscopic head-mounted visualization system enables the presentation to the user a single 3D world containing both the real and virtual elements.

The proposed approach aims at improving the interaction part using a fast hand gesture recognition algorithm to control the device interface. The 3D information from the ToF camera is a very valuable representation that helps in this task. Most approaches for hand gesture recognition from depth rely on a setup where the camera has a fixed position facing towards the user, in our case the camera is attached to the user head thus making the problem more challenging. The proposed gesture recognition algorithm can work in both camera setups, but has been tuned for the ego-vision environment.

This paper presents a novel complete system that allows a more natural interaction with the real world by exploiting data from multiple sensors: a ToF sensor, a standard video camera and an inertial sensor. This allows both acquiring the colored three-dimensional representation of the scene in front of the user and having an immediate description of its relative motion. The interaction is made possible by an advanced gesture recognition system that exploits a computing inexpensive real-time approach based on depth data and targeted to setups where the camera is head-mounted and can move. This module has been firstly presented in [24], in this work we extend the previous conference publication by including the gesture recognition module into a complete AR system.

More in detail on the gesture recognition module, it exploits two new sets of ad-hoc features, one representing the local curvature along the hand contour and the other the thickness of the hand region close to each contour point. An efficient solution for the construction of large synthetic training datasets is also presented and experimental results demonstrate that training on synthetic data can be sufficient for a reliable gesture classification with Support Vector Machines (SVM), thus avoiding the cumbersome construction of large real data training datasets. Finally, a new kind of 3D structure for the combination of multiple feature descriptors is proposed and proved to generalize well to the real data case.

Some examples applications of the proposed system are presented, with a particular focus on the presentation of an Augmented Reality (AR) system capable to seamlessly include a computer interface into the real world.

The paper is organized in the following way: Section 2 presents the related works, then Section 3 presents the architecture of the proposed system. Section 4 presents the gesture recognition module. Some applications of the proposed system are shown in Section 5. Section 6 contains the experimental results and finally Section 7 draws the conclusions.

2 Related works

Multi-modal human computer interaction is an inter-disciplinary research exploiting results from different fields from human-computer interaction to computer vision, computer graphics and many others. A complete review of the field can be found in [15], in this section we will focus on the works more related to the architecture of the proposed system. There has been a very few proposal of complete systems containing all the elements of this work but the various building blocks have been widely studied.

A complete system containing both the visualization and interaction module is presented in [14]. This work analyzes in detail the full pipeline required for the user interaction but the proposed input modalities are based on head movements and speech only. Hand gesture recognition is instead exploited in [9]. This system uses an RGB camera with three single-point 3D sensors to acquire the hand position that is then used in AR applications. The hand is identified from the single point 3D measures but the hand shape analysis used for gesture recognition is based on color data alone, thus limiting the accuracy of the gesture recognition module. Another system combining gesture recognition and AR is [3]. In this case the 3D sensor used for hand gesture recognition is not wearable but is placed on a tripod in front of the user, a choice that limits the applicability of the method to small office environments. In [21] and [22] a touch-less interactive augmented reality system based on a wearable device is proposed. A dynamic hands and feet gesture recognition system based on template matching is proposed and used to allow the interaction with augmented reality games.

Many different augmented reality systems have been proposed in the literature. A very good review of the first attempts to combine virtual and real elements into AR systems is contained in [2], while a more recent review of the advances in this field is [8]. Most of these works focus on the estimation of the user position and head orientation or on the positioning of the virtual elements but do not present a complete system including also the input module.

An important aspect for systems like the considered one is the joint calibration of the various acquisition and visualization devices. The works of [43] and [1] address the calibration of vision systems with inertial sensors.

Hand gesture recognition is a widely studied problem. The recent introduction of depth cameras has opened the way to many new approaches exploiting this kind of information but most approaches focus on the desktop setup with a fixed depth camera facing the user. Several approaches follow the standard pipeline consisting in firstly extracting a set of relevant features from the depth data and then applying a suitable machine-learning technique in order to recognize the performed gesture. An example of this family is the approach of [18] that exploits silhouette and cell occupancy features to build a shape descriptor that is then fed to a classifier based on action graphs. The method of [38] rely on volumetric shape descriptors and on Support Vector Machines (SVM) in order to recognize dynamic gestures from depth sequences. Volumetric descriptors and SVM are used in order to get the hand pose in [35]. Hand pose estimation from depth data is a widely studied problem, see [34] for a complete review.

In the proposed approach we instead focused on the direct recognition of the gestures without getting the pose and by working on a single frame. This problem has been tackled by comparing the histograms of the distance of hand edge points from the hand center in order to recognize the gestures in various works like [31], [30] and [11]. This technique leads to good results even if a precise alignment of the histograms is needed for optimal performances. To solve this problem (and hand segmentation) [31] and [30] use a black bracelet but this is a usability limitation. Different types of features can also be combined together as in the approach of [12]. Among the various features exploited in [12] there is also the curvature of the hand contour, but notice that the multi-resolution algorithm used for the descriptors extraction in this work is different and more computationally demanding than the one proposed in this paper. The approach has been extended with different classification and feature selection schemes [26] and by exploiting the data from a LeapMotion sensor [23]. The approach of [29] exploits a convex shape decomposition method in order to recognize the palm, fingertips and hand skeleton and then use this data for gesture recognition. In [16] the hand point cloud is described using Viewpoint Feature Histograms (VFH) and Hidden Markov Models are used for the classification of static gestures while dynamic data is handled with Dynamic Time Warping (DTW). A novel descriptor representing the local distribution of 3D samples acquired by the depth sensor is proposed in [41] and exploited for hand gesture recognition. Another recent work [10] exploits a Random Forest Classifier trained on synthetic data to recognize the various hand parts and then uses this information for hand gesture recognition. Notice that the estimation of the pose, skeleton or the recognition of hands parts is very useful for gesture recognition, but approaches exploiting these representations are typically more complex and computationally demanding. For the considered ego-vision system where a simple, fast and robust approach is needed we preferred a solution directly recognizing the gesture without going through intermediate representations.

The use of depth cameras has also been studied in the field of ego-vision, where the sensor is typically mounted over the user head. The approach of [33] consider both the problem of the hand segmentation in ego-vision systems and of gesture recognition, even if the main focus of the work is on the first task. For the segmentation a Random Forest classifier is used on a superpixel segmentation, while the gestures are classified using Exemplar SVMs. Another approach for hand gesture recognition with an head mounted camera has been

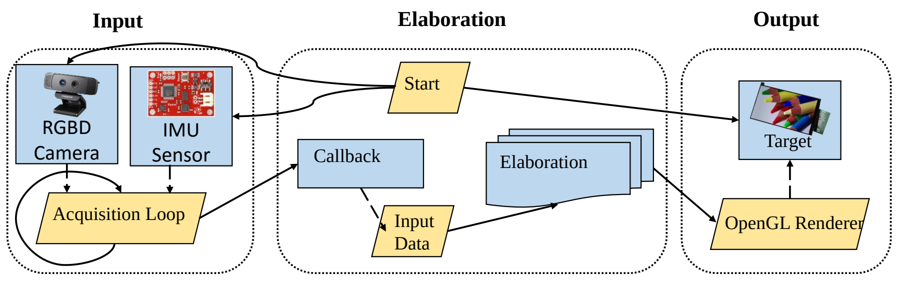


Fig. 1 Block diagram showing the main components of the proposed system.

presented in [25]. This approach is based on the analysis of a skeleton extracted from the hand but the experimental evaluation is limited to 3 static gestures and 5 dynamic ones. The recognition of dynamic gestures has been considered in [5], that deals also with the problem of removing the camera motion in ego-vision systems. The work of [37] deals with action recognition from a wearable depth camera. This work exploits a model that automatically mines discriminative states of the considered actions exploiting Multiple Kernel Learning. The problem of hand detection and segmentation in ego-centric vision has instead been considered in [6]. Another approach dealing with hand detection and tracking in ego-centric data is [4]. Hand detection and action recognition using the data from Google glasses is considered in [17]. Finally hand pose estimation using a wearable RGB-D camera is the topic of [32] that exploits synthetic training examples and multi-class rejection-cascade classifiers.

3 System Architecture

The proposed system extends the classic Head Mounted Display (HMD) paradigm by introducing additional sensors and functionalities enabling gesture recognition and augmented reality features. The overall architecture is shown in Fig. 1: the system can be divided into three main blocks, i.e., the 3D acquisition subsystem, the processing block and finally the visualization unit. Fig. 2 shows the employed hardware setup: the main elements are a high resolution monitor with a pair of aspheric lenses, a low-cost inertial measurement unit (IMU) and a RGB-D camera with a wide angle lens for the RGB optics.

3.1 3D acquisition subsystem

The data acquisition module combines the data from the 3 inputs (RGB, ToF and IMU) and provides them to the processing module. The inertial sensor is a SparkFun Razor 9-DOF IMU that operates at a frequency of *50Hz* and features a 3 axis accelerometer, a 3 axis magnetometer and a 3 axis gyroscope. The raw signals are filtered by using the Direction Cosine Matrix (DCM)

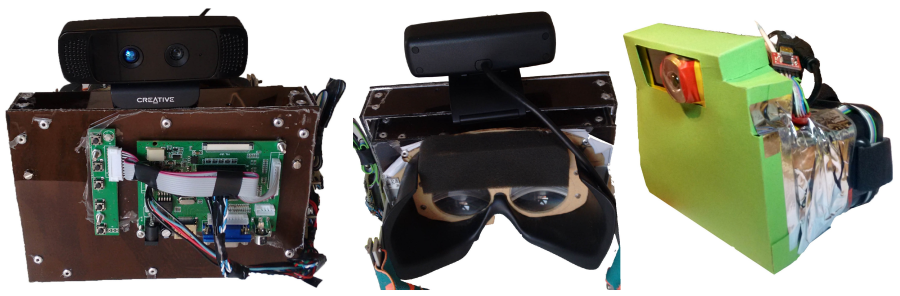


Fig. 2 Hardware setup of the proposed system.

algorithm [28] that takes care of reducing sensor noise and numerical errors. In particular the magnetometer has been calibrated in its final position in the complete setup, in order to account for the magnetic distortions of the other components.

The color and depth data are provided by a Creative Senz3D RGB-D camera. This camera performs well for short-range depth acquisition, but has a limited depth range and an high noise level in the far range. Even so, the very compact size and low weight of the device makes it suited for head mounted applications. Notice that a possible alternative is to use a stereoscopic setup, but the two cameras together with the Senz3D would increase the setup complexity and weight. Removing the depth camera and computing the depth map from the stereo setup is instead more computationally expensive and allow to obtain a less accurate depth map. Another limitation of the camera is the field of view (around 60 degrees). In order to solve this issue a wide angle lens is used for the color data, thus increasing the field of view to 120 degrees. The color and depth cameras have then been jointly calibrated using the calibration algorithm of [42].

3.2 Processing module

The processing module deals with multiple tasks. It is composed of a coordinator and several application components. The various tasks run on separate processes and are completely asynchronous in order to make the system fast and robust and to open the way for future improvements. The coordinator initializes and set up the whole system, controls the various input and output modules and keeps track of failures and performance issues. It collects and synchronize all the acquired data and make them available to the various components. The application components process the incoming data and extract the required information. Among them a key component is the gesture recognition algorithm that will be described in detail in Section 4. The processing module also acts as a bridge for the various applications that require augmented reality elements (e.g., web-pages, images, videos, or applications screen allowing the interaction with the real objects) and provides the corresponding data to the visualization module in order to add them to the rendered image. It also collects the output of the gesture recognition module and uses it to control the various applications. Finally it contains also the game engine required by gaming applications, e.g., the demo of Section 5.

3.3 Visualization module

The visualization module has the task of presenting to the user the augmented reality environment, that is a combination of virtual and real elements. The rendering of the virtual elements is performed with a standard OpenGL engine. As shown by Fig. 3 the main rendering loop works in the following way: the input data (color, depth and IMU information) is loaded and used to generate a textured point cloud in the user's 3D coordinate system. Then the point cloud is rendered two times from the viewpoints of the two eyes (the visualization and input system are calibrated) taking into account also the aspheric lens distortion. Finally the augmented reality elements are added to the rendering of the real world. The proposed rendering engine is very fast (the latency is about $10ms$) and by running on a separate thread it ensures that there is no delay in the images shown to the user.

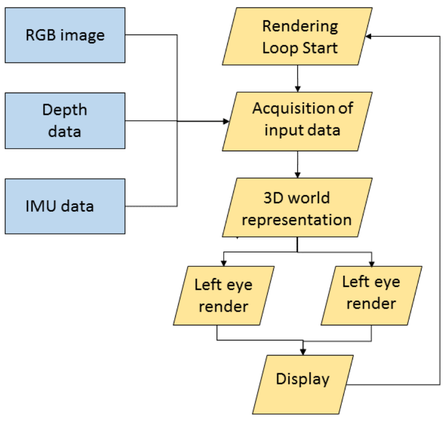


Fig. 3 Rendering pipeline exploited by the visualization system.

This procedure presents some interesting elements not present in standard approaches. Firstly the view of the real world is acquired from the RGB color camera, but notice that, since a single camera is available, 3D information is required to generate the two virtual views that are shown to the two eyes. The construction of the two virtual views needs to be performed carefully in order to obtain a satisfactory visual quality as discussed in several studies on stereoscopic image quality [20,39]. As previously introduced, the calibration information is used to create a colored point cloud that is then rendered from the viewpoints of the two eyes (an hole-filling algorithm is used to fill the regions eventually occluded from the viewpoints of one of the eyes). A critical issue is that many applications require the 3D visualization of the complete world around and the depth camera has a limited range of around $5m$. The rendering algorithm assigns to all the points without a valid depth value a fixed canonical value and these elements are then rendered as a flat surface, while objects actually measured by the depth camera results in a rendered 3D shape. This trick has proved to produce a good visual quality with only barely noticeable artifacts. Another critical point is that the lenses that focus the display at the short distance from the eye of the user, introduce a large geometric distortion. In order to compensate it, this distortion component needs to be firstly estimated. For this purpose a setup composed by a high-quality DSLR (Digital Single-Lens Reflex) camera seeing through the lens towards a small checkerboard has been used. Firstly the camera focus and zoom are fixed to achieve the best image quality for the small checkerboard. Then the camera is unmounted and, keeping the same settings, separately calibrated with a larger checkerboard using the approach of [42]. Finally the lens is placed back in position and the pictures of the small checkerboard seen through the target lens are acquired and undistorted using the camera calibration data. This process allowed to separate the lens distortion from the distortion component due to the camera: on the undistorted images, the only distortion remaining is the one introduced by the lenses. It is finally possible to apply again the checkerboard calibration algorithm of [42] finding the lens distortion parameters. These values are used to pre-distort the displayed image, that once saw through the lenses become undistorted and thus more realistic for the user.

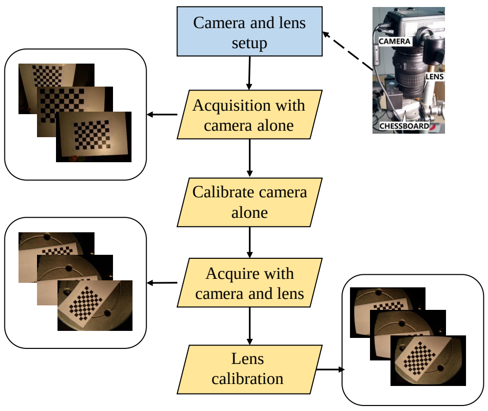


Fig. 4 Flowchart of the procedure used for the calibration of the optical system.

4 Gesture Recognition Module

In order to allow a natural interaction between the user and the virtual environment we included a gesture interface where the user can interact with the system with the bare hand without using any tool. This task is very challenging using color data alone, but recent research has demonstrated that depth data allow a very reliable solution [40]. Furthermore depth data directly provide the position of the hand in 3D space, a fundamental information for the applications of Section 5. The considered setup is quite challenging since the head-mounted camera is moving and furthermore the hand can easily move out of the field of view. For these reasons we decided to exploit a single frame approach that is independent from the camera position and motion and does not require a continuous tracking of the hand. In this way if the camera gets out of the field of view the gesture recognition can restart as soon as the hand gets back in the visible region.

The proposed approach encompass three main steps as depicted in Fig. 5. In the first step the hand is recognized and segmented from the rest of the scene. Then two sets of relevant features are extracted from the silhouette of the hand on the depth map. The first set of features captures the local curvature of the hand contour, while the second exploits a distance transform representation to get the thickness of the hand region corresponding to each contour point. Finally, the extracted features are used to build a three-dimensional vector representation that is fed to a classifier based on Support Vector Machines (SVM) trained exploiting a synthetic dataset.

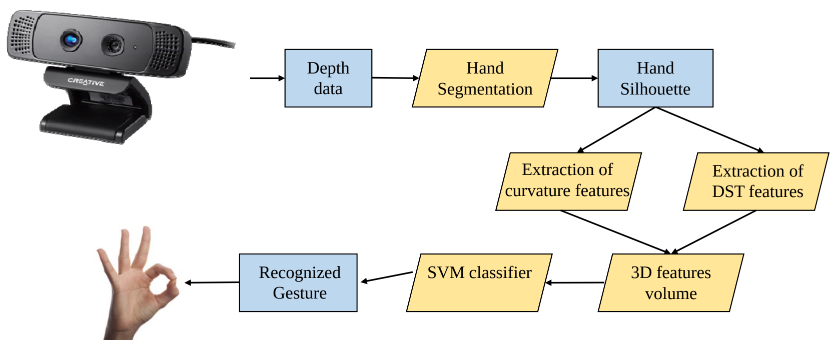


Fig. 5 Pipeline of the proposed hand gesture recognition approach.

4.1 Hand identification and segmentation

The first step consists in the extraction of the samples corresponding to the hand region from the depth map. The process assumes that the hand is close to the camera and starts with a thresholding of the depth map that removes all the samples with a distance bigger than a pre-defined threshold that depends on the selected application (e.g., in the ego-vision system we set it to $0.5m$). In this way an initial hand mask is obtained (see Fig. 6b): the set of all the points in this initial mask will be denoted with $\mathcal{H}'$. The approach can be made more robust by checking the compatibility of the color of the samples with the skin color [12], at the expenses of an increase in computation time. Typically after this operation the hand remains in the scene together with the first part of the arm and some other isolated objects or spots due to the noise of the sensor. The longest contour, that typically corresponds to the hand, is extracted from the binary mask and a second mask $\mathcal{H}''$ containing the area inside this contour is generated. This mask contains only the hand without other objects but internal contours or holes inside in the hand region will be filled (e.g., the area between the thumb and index of the "Ok" gesture in Fig. 6). For this reason the masks $\mathcal{H}'$ and $\mathcal{H}''$ are intersected in order to get the mask $\mathcal{H}'''$:

$$\mathcal{H}''' = \mathcal{H}' \cap \mathcal{H}'' \tag{1}$$

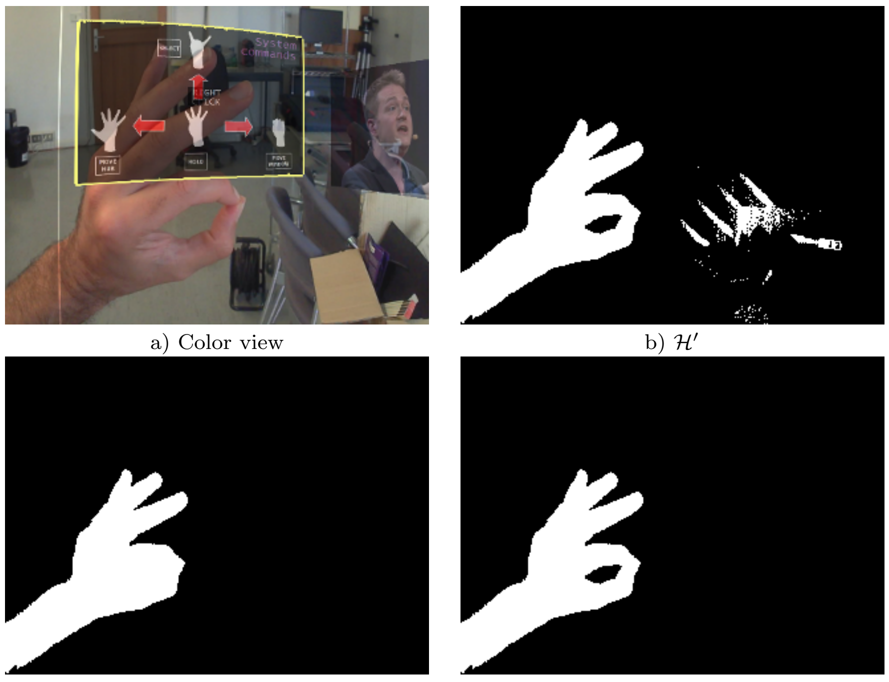
which contains only the hand points together with the wrist and the first part of the forearm. A distance transform (DST) is applied to the binary mask $\mathcal{H}'''$ in order to find the hand center: the maximum of the DST assumed to correspond to the center. More precisely a new data structure $D'(\mathbf{x}) = D'(u,v)$ containing the distance from each pixel $\mathbf{x} = (u,v)$ to the closest point not belonging to the mask is constructed, i.e.,

$$D'(u,v) = \min_{(j,k)\notin\mathcal{H}'''} (|u-j| + |v-k|) \tag{2}$$

where the Manhattan (i.e., d_1) distance has been used as the distance metric (it provided the same performances of the Euclidean one and proved to be faster to compute). The approach of [7] has been used for the computation of the distance transform, an example of the results of the algorithm is shown in Fig. 7a. The maximum of $D'(u,v)$,

$$D'_{max} = \max_{(u,v)} \quad D'(u,v) \tag{3}$$

is computed and the point corresponding to the maximum of the DST is selected as the palm center C (if multiple pixels with the same highest distance transform value are present their barycenter is selected as the hand center). After locating the palm center, a circle of radius $R = 3D'_{max}$, centered on C_H, is computed and all the points outside the circle are excluded from the mask thus obtaining the final hand mask $H(u,v)$. Notice that this final step allows to remove the forearm if it has been included in the hand shape. The distance transform is also re-computed using this updated mask, thus obtaining a new set of distance values $D(u,v)$ that will be used for the construction of the distance features (see Section 4.2). The orientation of the hand is then obtained by computing the first two moments of the binary mask (see Fig. 7b for an example). Finally the hand region is re-sampled to 100×100 pixels in order to normalize the data with respect to different hand sizes. The external contour of the mask H is then computed, leading to a sequence of points $\mathbf{b}'_1, ..., \mathbf{b}'_k$. The sequence is uniformly re-sampled into a sequence $\mathbf{b}_1, ..., \mathbf{b}_n$ with a constant number of samples n. This allows to make the approach invariant to the number of pixels in the contour (for the results we used $n = 500$). The sequence $\mathbf{b}_1, ..., \mathbf{b}_n$ will be used in the feature extraction step. The starting point $\mathbf{b}_1$ of the contour is selected as the one corresponding to the computed orientation, in order to make the approach also rotation invariant. This allows to recognize gestures independently of the hand orientation, but also makes all the gestures with the same hand pose equivalent, if gestures like pointing left or right need to be disambiguated the problem can be solved in two steps by firstly recognizing the gesture and then disambiguating between the two sub-cases with the moments information.



a) Color view b) $\mathcal{H}'$

c) $\mathcal{H}''$ d) $\mathcal{H}'''$

Fig. 6 Computation of the initial hand mask: a) Color view with augmented reality elements from the ego-vision system; b) Binary mask $H'(u, v)$ from the thresholding operation; c) Area $H''(u, v)$ inside the longest contour; d) Intersection $H'''(u, v)$ of the two masks.

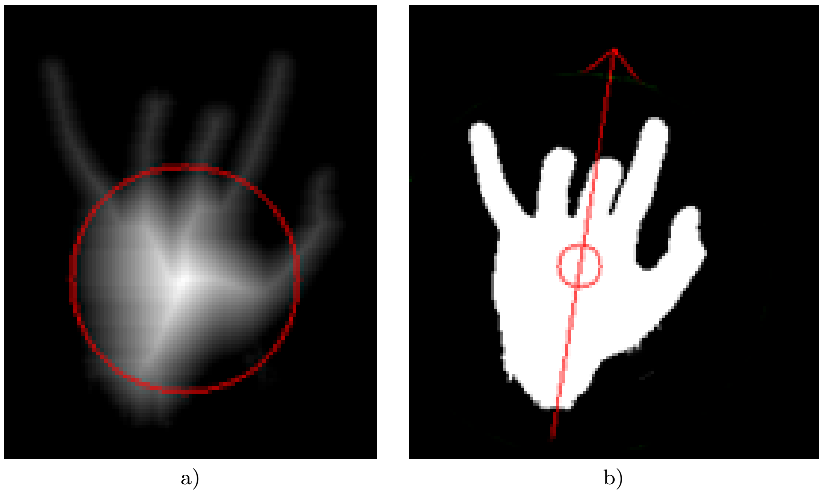


Fig. 7 a) Distance transform computed on a sample hand mask (the circle has a radius equivalent to the maximum of the distance transform); b) hand center and orientation obtained from the binary moments.

4.2 Features extraction

Two different sets of features are then extracted from the information computed in Section 4.1. The first set describes the curvature of the contour $\mathbf{b}_1, \ldots, \mathbf{b}_n$. The contour is analyzed and for each sample $\mathbf{b}_i$ the curvature is computed as the angle made between the set of the preceding samples and the set of the subsequent ones. More precisely, let $\mathbf{p}_i$ be the barycenter of the k preceding samples and $\mathbf{s}_i$ the one of the k subsequent samples (for the proposed system we used $k = 5$), i.e.,:

$$\mathbf{p}_i = \frac{1}{k} \sum_{j=1}^{k} \mathbf{b}_{i-j} \tag{4}$$

$$\mathbf{s}_i = \frac{1}{k} \sum_{j=1}^{k} \mathbf{b}_{i+j} \tag{5}$$

Finally, the curvature c_i associated to the i-th sample is computed as the angle between the vectors $(\mathbf{b}_i - \mathbf{p}_i)$ and $(\mathbf{s}_i - \mathbf{b}_i)$, i.e.:

$$c_i = \arccos\left[(\mathbf{b}_i - \mathbf{p}_i) \cdot (\mathbf{s}_i - \mathbf{b}_i)\right] \tag{6}$$

As shown in Fig. 8, the algorithm computes the angle from the straight direction made by the contour in proximity of the i-th sample and thus each value c_i is included in the range $[-\pi, +\pi]$ (notice that also the sign of the curvature is considered). Averaging over the k closest pixels in both directions makes the approach more stable with respect to the noise on the contour, a relevant issue due to the high noise level of the employed depth camera. The

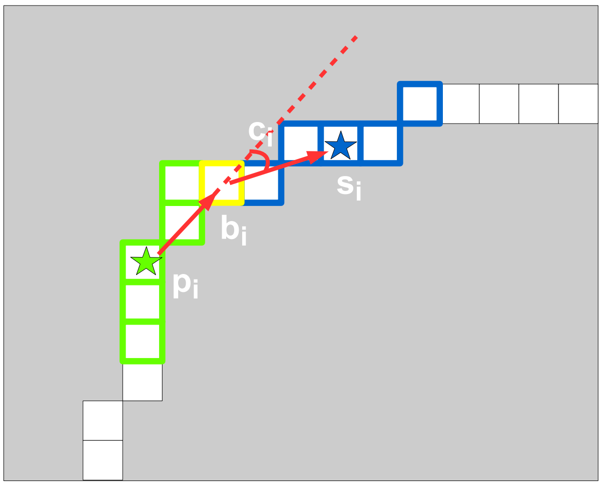


Fig. 8 Computation of the curvature value c_i at the location of sample $\mathbf{b}_i$. Point $\mathbf{p}_i$ (shown with a green star) is the barycenter of the preceding samples and $\mathbf{s}_i$ (shown with a blue star) is the barycenter of the subsequent ones. (*Best viewed in color*)

result of this computation is a sequence of values $\mathbf{c} = (c_1, \ldots, c_n)$ representing the local curvature at each location of the contour.

The second descriptor represents the thickness of the hand region associated to each contour point. This descriptor is based on the previously computed distance transform $D(u, v)$ on the hand mask $\mathcal{H}$. For each contour sample $\mathbf{b}_i$, the values of $D(\mathbf{x})$ along the direction $\mathbf{n}_i$ perpendicular to the contour are analyzed (see Fig. 9). The first maximum of the DST along the direction $\mathbf{n}_i$ going towards the center of the hand is then selected (the search stops when a maximum is found) and used as the feature value, i.e.,:

$$d_i = \max_k D(\mathbf{b}_i + k\mathbf{n}_i) \tag{7}$$

where k is the displacement from the contour in pixels that grows until a maximum is found (i.e., the values of D start to decrease).

At the end of this process a sequence of values $\mathbf{d} = (d_1, \ldots, d_n)$ is obtained, each value d_i giving the distance value associated to the i-th sample. As shown in Fig. 9, higher values correspond to the palm area and to groups of fingers joined together, while the DST takes smaller values in the fingers region.

4.3 Gesture Classification

After computing the two feature descriptors, their values are fed to the classification algorithm. In order to perform the recognition, feature data are firstly rearranged in a three dimensional structure and then fed to a multi-class one-against-one SVM classifier. The employed classifier has been trained on a syn-

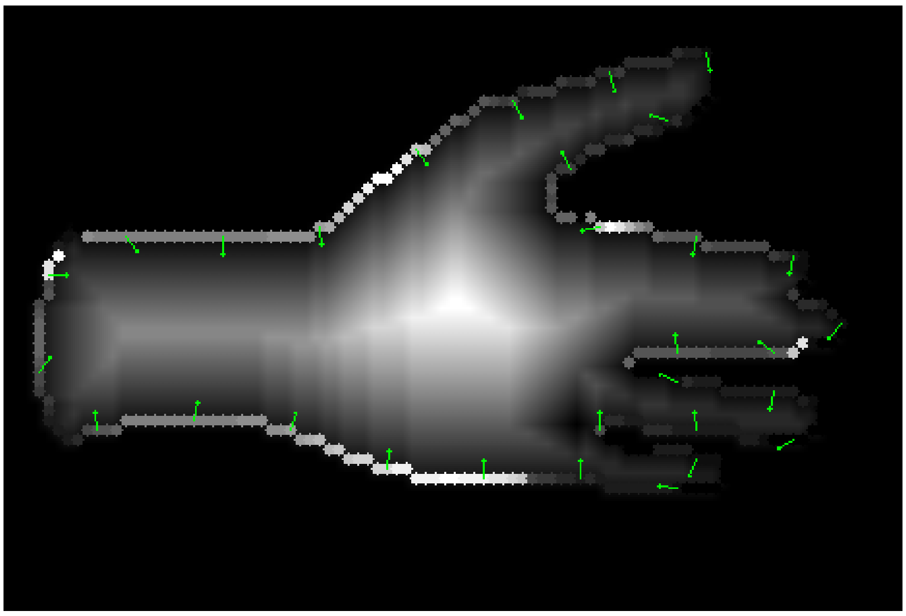


Fig. 9 Computation of the distance-based features: the green arrows represent the direction along which the maximum DST value is chosen and the color of each edge pixel is proportional to the value of the DST in the corresponding maximum.

thetic dataset constructed using a rendering system developed ad-hoc for this work.

For each input frame, a feature vector is computed starting from the contour curvature and distance-based features extracted as described in Section 4.2. An example of the content of the two feature vectors for a sample gesture is shown in Fig. 10, where the vectors **c** and **d** are shown by plotting the value of each element against its index. A first possibility is to feed the Support Vector Machine with a simple concatenation of the two vectors $\mathbf{c} = (c_1, \ldots, c_n)$ and $\mathbf{d} = (d_1, \ldots, d_n)$. The effectiveness of this basic solution relies on how precisely the hand orientation is estimated for each frame, or at least on the invariance of the orientation information with respect to different executions of the same gesture. Due to the not too precise estimation of the hand orientation this approach provided sub-optimal results in our tests. For this reason the data have been rearranged in order to better capture significant correlations between the two descriptors. A possible workaround for the orientation issue is to plot on a plane the discrete parametric curve given by (c_i, d_i), where the parameter i ranges from 1 to n. Fig. 11 shows an example of this representation, i.e., it displays the curvature and distance couples (c_i, d_i) taking the index i as an implicit parameter. In this case, since the shape of the curve does not depend on which contour sample is chosen as the starting point, the orientation estimate plays no role in the descriptor computation. On the other hand, using only this type of features without exploiting the information about the variations of the curvature or of the distance with respect to the position of the samples may result in a significant drop in the prediction accuracy of the classifier, especially if a relatively good orientation estimate is available.

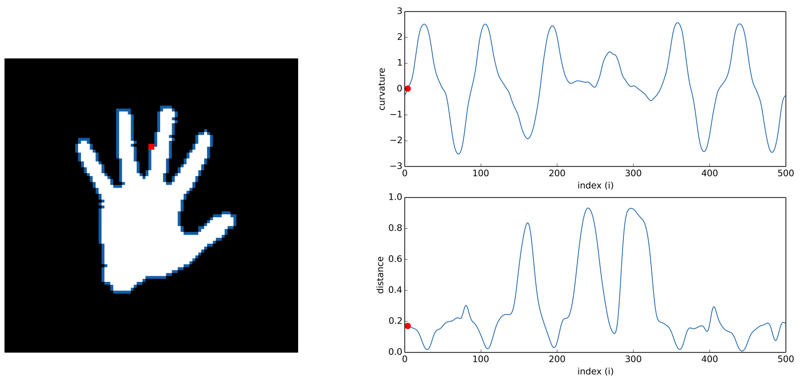


Fig. 10 Curvature and distance features for a sample gesture. The feature values have been plotted starting from the red point and proceeding in clockwise order along the hand contour.

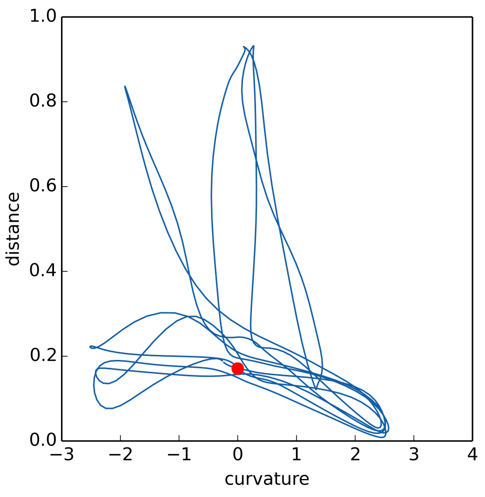


Fig. 11 Plot of the the parametric (c_i, d_i) curve for the sample gesture of Fig. 10a.

The idea exploited in this work is to make explicit the parameter i, or equivalently to look at the triples (c_i, d_i, i) for $i = 1, \ldots, n$ as coordinates in a three-dimensional space. An array representation of a quantized and smoothed version of the resulting 3D plot is computed and used as the feature vector to be given in input to the classifier. More precisely, the ranges of possible values of $c_i \in [0, 2\pi]$ (curvature values are shifted from $[-\pi, \pi]$ to $[0, 2\pi]$ in order to simplify the representation) , $d_i \in [0, 1]$ and $i \in [1, n]$ are divided into three set of quantization intervals, i.e., $I_c = \{1, \ldots, n_c\}$, $I_d = \{1, \ldots, n_d\}$ and $I_i = \{1, \ldots, n_i\}$ respectively. A quantization function is then used mapping each triple (c_i, d_i, i) to the corresponding combination of quantization intervals, represented by a triple of integers (j, h, k) in $I_c \times I_d \times I_i$. A three dimensional array $\mathbf{A}$ of size $n_c \times n_d \times n_i$ is created where the entry $\mathbf{A}(j, h, k)$ is set to 1 if there exists at least one triplet (c_i, d_i, i) which is mapped to $(j, h, k) \in I_c \times I_d \times I_i$ by the quantization function, and it is set to 0 otherwise. Finally, a multi-dimensional Gaussian filter is applied to the array, using suitable standard deviations σ_c, σ_d, and σ_i for each dimension along which the filter is applied. In order to handle the boundaries, before applying the filter, the array is extended outside its borders with zeros along the first two dimensions (those relative to the intervals I_c and I_d), while wrapping is performed along the third dimension (the one accounting for intervals I_i). An example of the resulting representation is given in Fig. 12, where four slices along the third dimension of the 3D array are shown. Notice how the Gaussian smoothing makes the proposed approach more stable with respect to the sensor noise and to inaccuracies in the computed contour. The plots in the figure are computed from the same gesture of Fig. 10, and the different intensities of blue are used to represent the element values ranging from 0 (white) to 1 (dark blue).

By varying both the granularity of the quantization intervals I_i and the amount of Gaussian smoothing applied along the third dimension (i.e., varying σ_i), it is possible to implicitly reduce the weight that the position information of the feature values (and consequently the orientation information) has in describing the different gestures. The number of intervals n_c, n_d and the standard deviations σ_c, σ_d relative to the first and second dimension have also an impact on the relative importance between the two type of features.

In the proposed system $n_c = 20$, $n_d = 20$ and $n_i = 10$ have been used to perform quantization. The standard deviations of the Gaussian filter have been set to $\sigma_c = 1.5$, $\sigma_d = 1.5$ and $\sigma_i = 1.0$. An additional smoothing has been applied separately to both curvature and distance vectors $\mathbf{c}$ and $\mathbf{d}$ as a preprocessing step before computing the three-dimensional array. In particular, a 1-dimensional Gaussian filter with standard deviation $\sigma = 2.0$ has been used to smooth curvature values. For the distance vectors a standard deviation $\sigma = 4.0$ has instead been used. The quantization and filtering parameters have been experimentally determined and provided the optimal performance in our setup, as expected using a finer quantization and a lower amount of smoothing increase the accuracy of the descriptors but reduces the robustness to noise and other issues. If a different depth camera or a different set of gestures are used these values can be changed in order to find the optimal trade-off between robustness and accuracy.

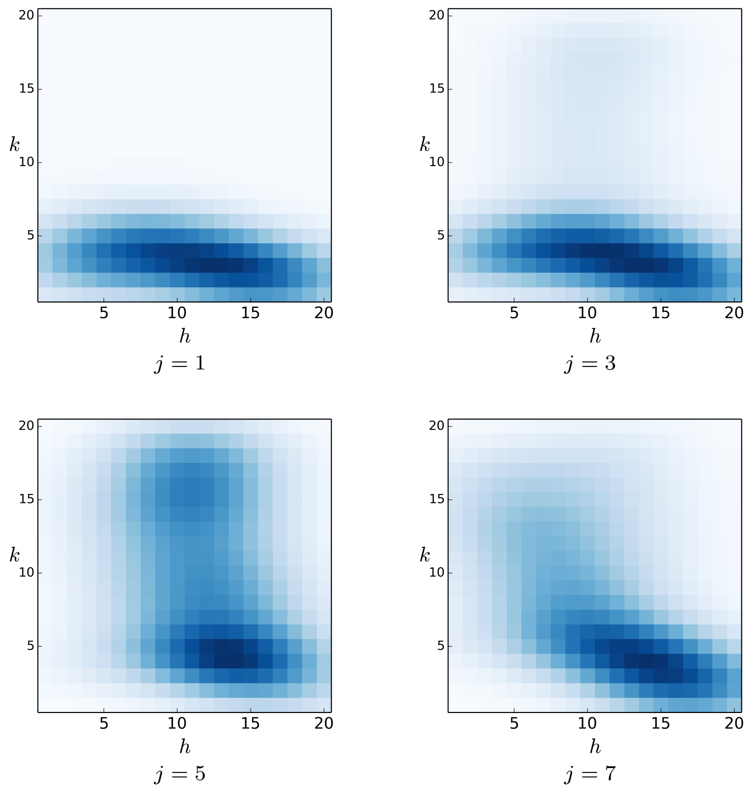


Fig. 12 Example of the three-dimensional array **A** corresponding to the gesture in Fig. 10 for some sample values of the index j. The employed parameters are $n_c = 20$, $n_d = 20$, $n_i = 10$, $\sigma_c = 4.0$, $\sigma_d = 1.5$, $\sigma_i = 1.0$.

In Fig. 13 the computed features are reported for three different repetitions of 4 different gestures (gestures G1, G3, G5 and G6 of the test dataset used to evaluate the system). For each repetition, four slices of **A** are displayed corresponding to values $j = 1, 3, 5, 7$, notice how the basic shape of the descriptor remains the same along the various repetitions. By looking also at the different gestures in the figure it is possible to realize that the shapes are characteristic of the performed gesture and allow a very good discrimination between the different gestures.

As previously introduced the classification is performed with a multi-class Support Vector Machine. We used the implementation from the scikit-learn library [27]. The employed kernel is the Radial Basis Function (SVM) with parameters $C = 1$ and $\gamma = 0.0025$. The multi-class classification has been performed using the one-against-one approach. The classifier has been trained by computing the feature vector A for each sample of the training dataset as previously described.

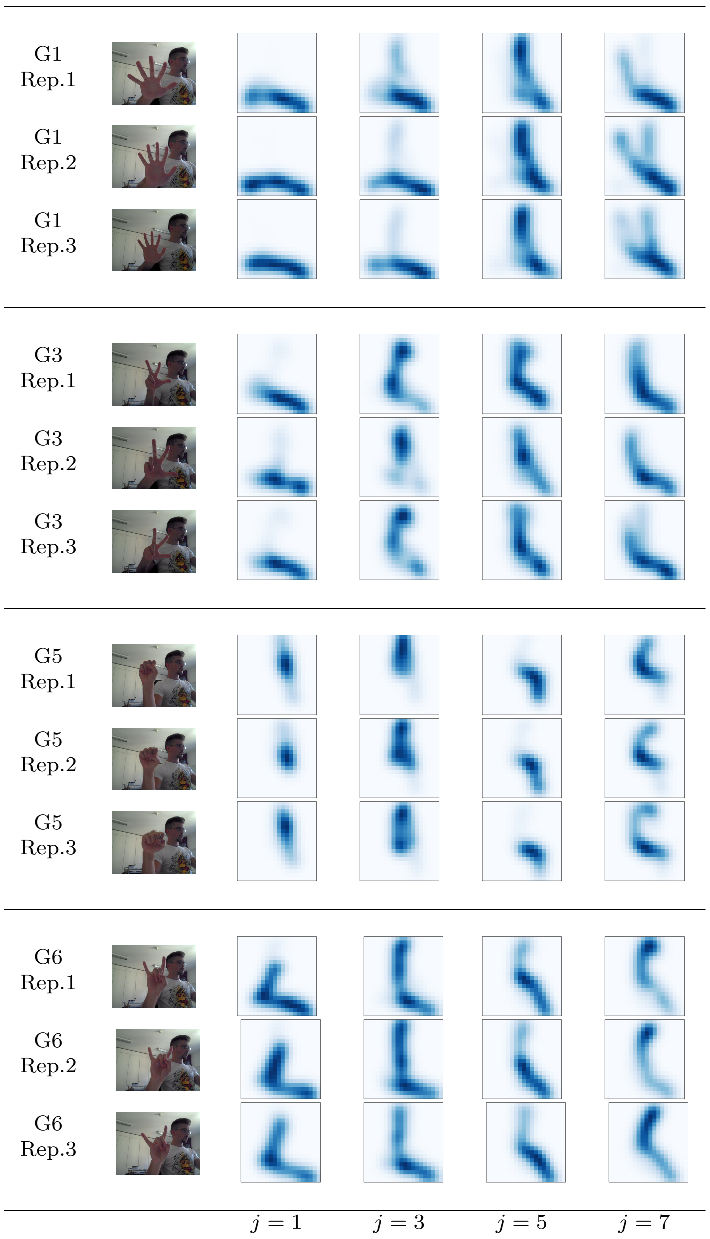


Fig. 13 The figures shows 4 different slices of the feature array **A** for 3 different repetitions of 4 different gestures (i.e., gestures G1, G3, G5 and G6 from Fig. 22). Notice how the slices are similar for different repetitions of the same gesture and very different among different gestures.

A large training dataset allows a more accurate classification but on the other side its construction requires a huge amount of manual work. In order to solve this problem we performed the training on a computer generated dataset that simulates the data that would have been acquired in a real setup. The rendering of an accurate hand shape is a challenging task because of its anatomy and because of the large variety of poses that it can assume. A rendering system exploiting the linear blend skinning technique [19] and a fast OpenGL renderer derived from the one used in the visualization system has been developed (it is available at `http://lttm.dei.unipd.it/downloads/handposegenerator/`). The various hand poses have been generated by rendering a 42-DOF skeleton (see Fig. 14) attached to a textured 3D model derived from the one used in the open source library *LibHand* v0.9 [36]. Depth information is finally exported in order to simulate the data provided by the depth camera in the real setup and, as it is possible to see from Fig. 15, the synthetic depth images are a good representation of the real ones. For each gesture we considered several different hand positions and orientations, different inter-distances of the fingers and we generated a large dataset of 35200 training frames. There are 3200 different samples for each of the 11 considered gestures (see Section 6) corresponding to different orientation and small variations in the position of the fingers. In particular 5 different inclinations in the x-axis direction, 4 in the y-axis one and 5 for the z-axis have been considered, thus obtaining 100 possible orientations. Different small perturbations of the positions of the fingers (e.g., a bit closer one to the other) have also been considered and 32 different variations of each pose have been generated. By combining all possible positions and orientation, $100x32 = 3200$ samples for each gesture have been obtained and used to train the classifier.

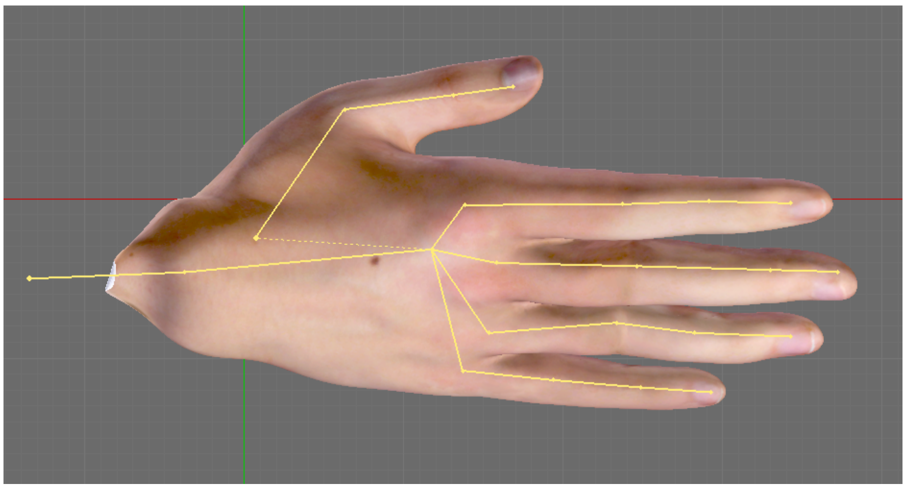


Fig. 14 42-DOF hand skeleton used by the rendering library.

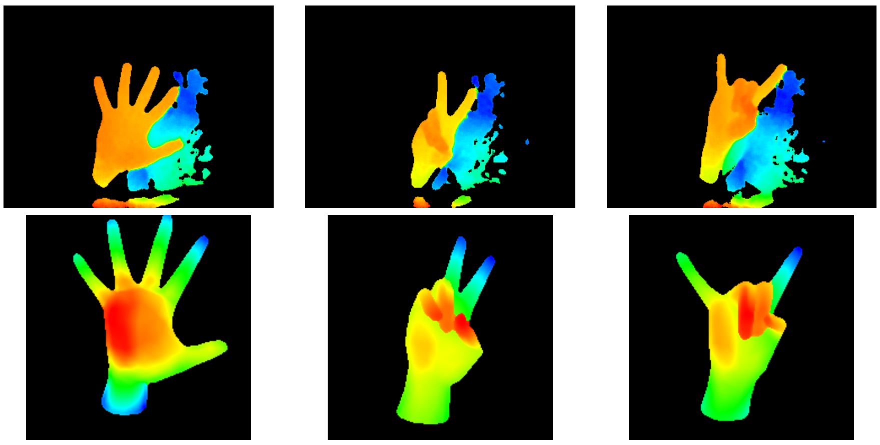


Fig. 15 Comparison of real data acquired by the ToF sensor (first row) and synthetic depth maps (second row) on 3 sample gestures.

5 Applications

The proposed system can be used in a wide range of contexts and some sample applications have been built. The first considered environment is a virtual reality one: Fig. 16 shows a simple video-game exploiting the considered system. In this case the main difference with respect to traditional systems is the gesture interface. Furthermore, compared to other gaming platforms controlled with gestures like Microsoft's Kinect, the use of the head mounted sensor allows to move around while playing the game instead of being locked in front of the sensor.

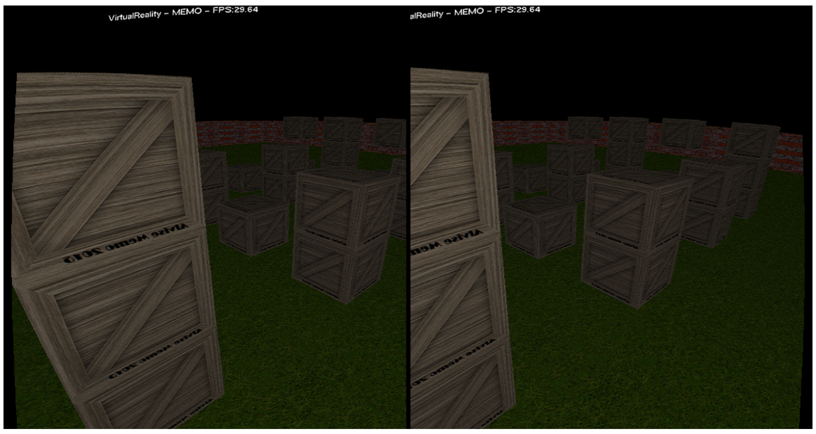

VirtualReality - MEMO - FPS:29.64
alReality - MEMO - FPS:29.64

Fig. 16 Example of a virtual reality 3D game running inside the proposed system. The figure (as the others in this section) show the two rendered views presented to the two eyes.

It is also possible to create augmented reality games that combine together real and virtual elements. In particular the depth sensor calibrated with the visualization system allows to create virtual elements properly placed and interacting with the real environment and the user. An example is the virtual sphere in the example of Fig. 17. This is an example of the novel interaction schemes made possible by 3D data and by the gesture recognition system: the demo application allowed the creation of a sphere centered exactly on the user hand, that remains attached to the hand when it moves in 3D space. In this demo when a selected gesture is performed the hand orientation is computed and the sphere is thrown.

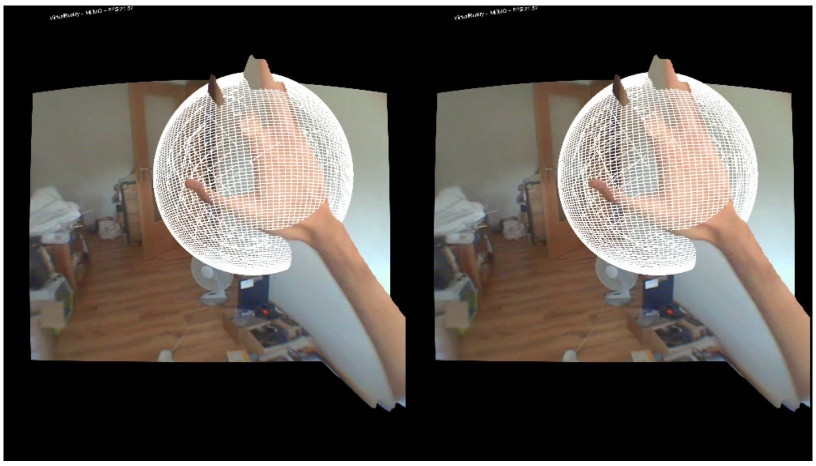


Fig. 17 Augmented reality game including virtual and real elements. Notice how the virtual sphere is properly placed in the 3D location corresponding to the center of the hand.

A more interesting application for the proposed system is the construction of human-computer interfaces. The proposed system allows the interaction with a computer while the user is moving around in the real world, thus opening the way to novel augmented reality applications. A traditional operating system interface with multiple desktops can be combined with the proposed system as shown in Fig. 18 and in the submitted video. This demo exploits both the RGB and depth data from the sensor: the color data allows the user to stay focused in the reality he is in, while the depth data allows to place a set of customizable displays on the preferred positions in the 3D space of the real world and to adjust them according to the user head position and to the real objects interacting with the computer interface. In particular the various windows instead of lying in the computer screen can be virtually placed in any location of the real world. For example if an user has to interact with a machinery, the window containing the interaction controls or the instructions to use the machinery can be placed directly over the object in the real world (a couple of examples are shown in Figs. 19 and 20). The use of the RGB-D camera allows a more precise localization than in standard augmented reality applications exploiting visual features.

The gesture interface of Section 4 can be used not only to interact with the virtual desktops using the hand as a mouse and the gestures for the commands but also to move the virtual desktops around in the 3D space. Fig. 21 shows an example of how the user is able to move the displays around with a simple gesture. The displays are also fully customizable through the gesture interface and the user can change their size, position, orientation, transparency and distance from his eyes. The gestures recognized by the system are translated into movements or actions depending on the setup. For movements, as the field of view of the ToF depth sensor is not too large, the following system is used: when the gesture is recognized, a starting point is saved and provides an anchor referring to which the consequent positions are evaluated in order to allow the movement in the corresponding direction.

The gestures are acquired in real time for each frame and stored in a queue. Only gestures that have been recognized in a number of recent frames in the queue are considered, thus avoiding issues due to erroneous recognitions in isolated frames. In detail, let us assume that the target is the recognition of h different gestures $G1, G2, .., Gh$. We can introduce an indicator function $l_{i,g}$ that is equal to 1 if the proposed classification algorithm detects gesture g at time instant i and to 0 otherwise. In order to increase the robustness of the approach at each time instant t we consider the last k frames and we compute the number of times $C_g = \sum_{i=0}^{k-1} \mathbf{l}_{t-i,g}$ each gesture is detected. Then we denote with $C_{M1} = \max_g C_g$ the number of times the most frequent gesture has been detected and with C_{M2} the frequency of the second most common one. Finally we detect a gesture and send the corresponding command if it has been recognized more frequently than the second candidate with a 30% margin, i.e., if the following condition is satisfied:

$$C_{M1} > \frac{C_{M2}}{0.7}. \tag{8}$$

Moreover the system can be used in a wide spectrum of environments with the main goal of enabling users to increment the information that is immediately available. Some examples are:

- Medical applications where augmented information coming from diagnostic data can help the operator (for example X-ray or MR data over the live view of the patient).
- Sports applications where the movements of the player or the trajectories of the ball can be used to perform a more accurate training and evaluation of the the athlete.
- An engineer could use it for having multiple virtual monitors while working on some machinery.
- For architects it can be used to enable live preview of their creations (for example by overlaying the new project over a building that needs to be renovated), thus evaluating the comfort and feasibility.

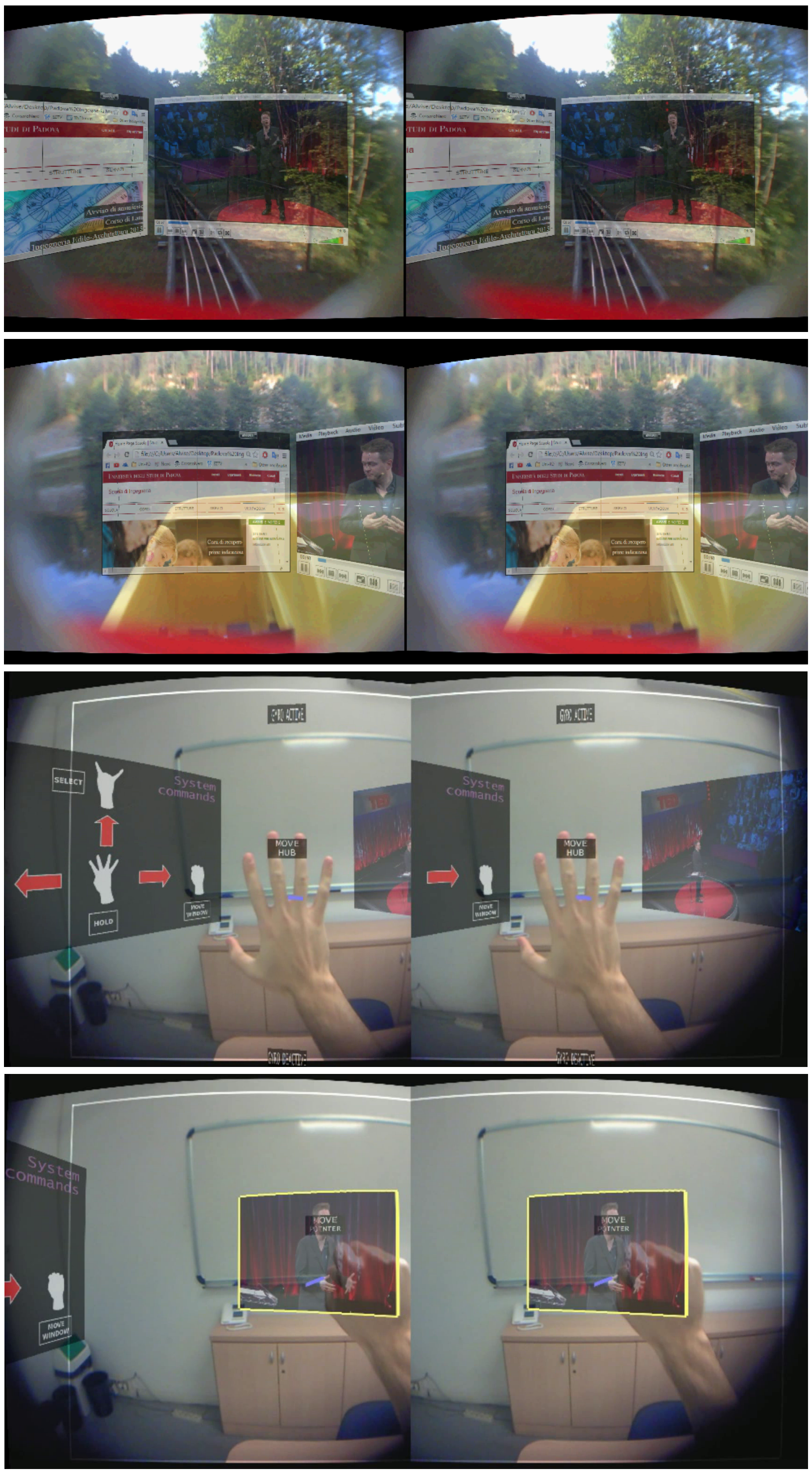


Fig. 18 Examples of virtual desktops placed in the real world environment.

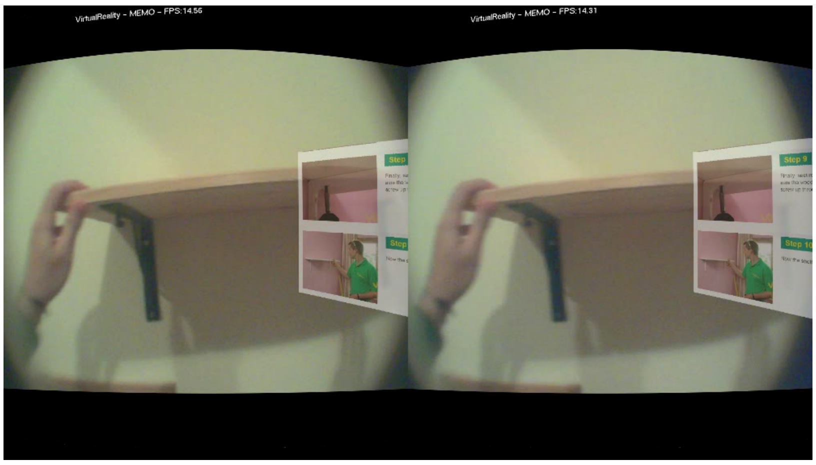

VirtualReality - MEMO - FPS:14.56
VirtualReality - MEMO - FPS:14.31
Step 9
Step 10

Fig. 19 Example of augmented reality application where the proposed system is used to help the user mounting a shelf.

6 Experimental evaluation

This section presents an evaluation of the accuracy of the gesture recognition module and then a brief report on the usage of the complete system.

6.1 Gesture recognition module

The gesture recognition module has been widely tested in different environments inside the proposed ego-vision system, but in order to perform a numerical evaluation of its performances an hand gesture dataset has been acquired with the Creative Senz3D camera. The dataset contains 11 different gestures performed by 4 different people and is available at `http://lttm.dei.unipd.it/downloads/gesture2` . A sample color and depth frame for each gesture is shown in Figure 22, notice how the dataset contains different gestures with the same number of raised fingers, gestures with fingers very close each other and with fingertips touching each other. Each gesture has been repeated by each user 30 times for a total of 1320 acquisitions.

Table 1 shows the results obtained by training the classifier on the synthetic dataset as discussed in Section 4.3 and then performing the testing on real data contained in the acquired dataset. Most of the gestures are correctly recognized and an average accuracy of 90% has been obtained. This is a quite remarkable result considering that the training has been performed on a synthetic dataset without any real acquisition. The extracted features are stable across different repetitions of the same gesture but at the same time they are able to discriminate different gestures. Looking more in detail at the results in Table 1 it is possible to notice how the accuracy is above 80% for all the considered gestures and above 90% for most of them. The most critical ges-

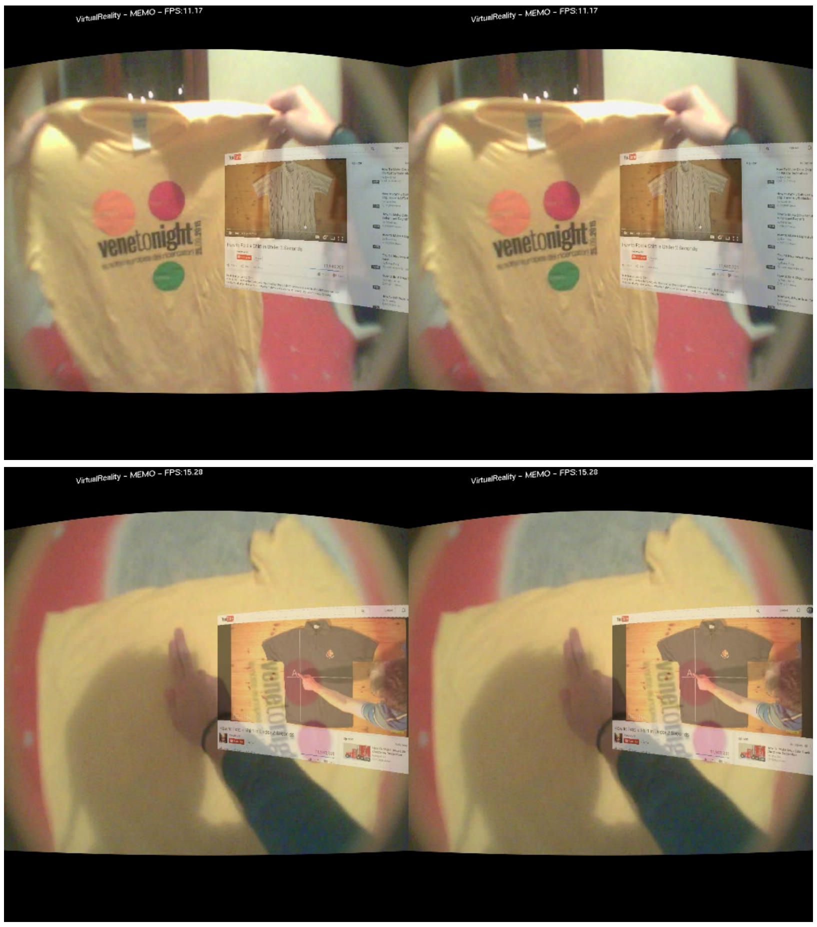

VirtualReality - MEMO - FPS:11.17
VirtualReality - MEMO - FPS:15.28

Fig. 20 Example of augmented reality application where the proposed system is used to help the user choosing a shirt of the correct size.

tures are G2, G7 and G9. G2 is sometimes confused with G3 since they differ only for the position of the thumb that in some acquisitions is not very easy to detect from the silhouette. G7 and G11 both have a single raised finger and sometimes are confused each other. This is a typical problem for many gesture recognition approaches due to the limited accuracy of the orientation information and of the not too precise measures from the depth camera. The three-dimensional data structure allows to improve performances on such configurations, but there is still room for further improvements. Finally G9 is another challenging gesture due to the touching fingers, it is recognized 82% of the times, a good result considering that many approaches based on fingertips detection are typically unable to handle this gesture.

With regard to the effectiveness of the proposed features arrangement, it is interesting to look at Table 2 which reports, in the first row, the per-gesture

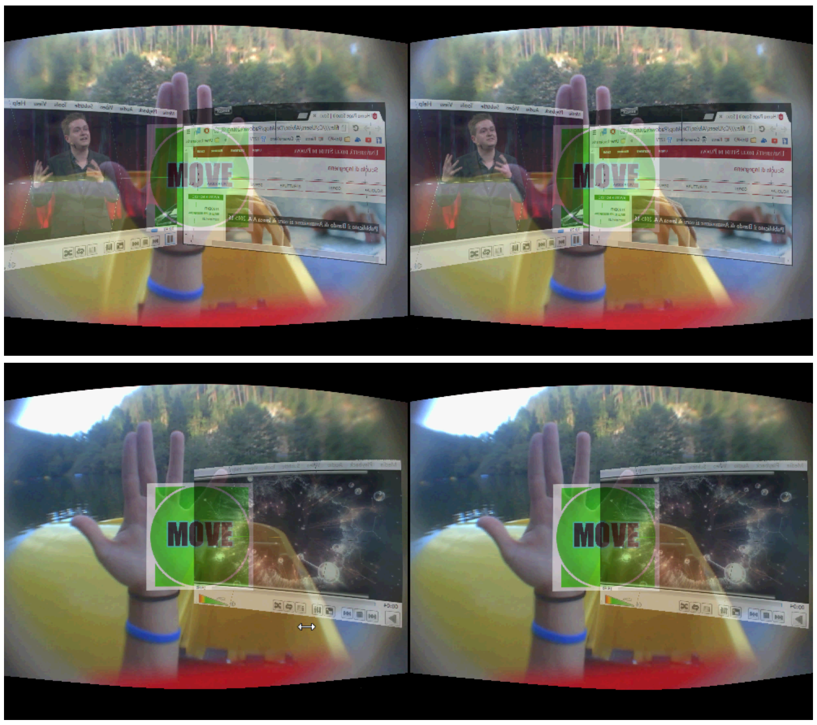


Fig. 21 The virtual displays can be moved around in the 3D space using the gesture interface.

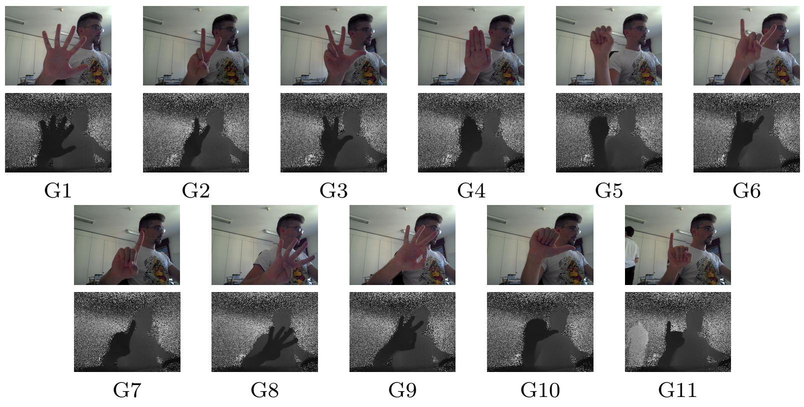


Fig. 22 Sample color and depth frame for each gesture in the experimental results dataset.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11
G1	**0.94**	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.00
G2	0.00	**0.84**	0.12	0.00	0.00	0.03	0.00	0.00	0.01	0.00	0.00
G3	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
G4	0.00	0.00	0.00	**0.88**	0.09	0.00	0.02	0.00	0.00	0.00	0.02
G5	0.00	0.00	0.00	0.06	**0.88**	0.00	0.00	0.00	0.00	0.06	0.00
G6	0.00	0.03	0.00	0.00	0.00	**0.88**	0.00	0.00	0.04	0.00	0.05
G7	0.00	0.00	0.01	0.00	0.00	0.00	**0.80**	0.00	0.00	0.00	0.19
G8	0.02	0.02	0.00	0.00	0.00	0.00	0.00	**0.93**	0.04	0.00	0.00
G9	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.05	**0.82**	0.00	0.01
G10	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	**0.92**	0.00
G11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**

Table 1 Performance of the proposed approach on the acquired dataset. The Table shows the confusion matrix for the 11 different gestures contained in the dataset. Due to rounding effects some rows can sum up to 0.99.

accuracies obtained by feeding the SVM with a simple concatenation of the two features vector **c** and **d**. These results are compared with those achieved by the proposed approach (that exploits the three-dimensional structure to combine the two types of features), which are listed in the second row. The table clearly shows how the proposed scheme allows to significantly improve the recognition accuracy while the effectiveness of using the **c** and **d** vectors alone or concatenated depends on how well the hand orientation is estimated and small variations in these estimates may lead to poor classification results.

We also compared the proposed approach with competing strategies from the literature. In [12] an effective approach for static hand gesture recognition exploiting different types of features has been proposed. The approach of [12] is very effective when real data is used for training but is not able to exploit the synthetic data of our training set as well as the proposed work. Among the various descriptors proposed in [12] the best performing ones are the distance features, that allow to obtain an accuracy of 75%, relatively good but quite far from the 90% of the proposed approach. Even by combining multiple descriptors together did not allow to improve the performance w.r.t. to the distances alone. We also tested the descriptors based on the convex hull concavities proposed in [13], but this approach lead to a limited accuracy of 65% on the considered dataset.

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	All
$\mathbf{c}+\mathbf{d}$	**0.97**	0.53	0.84	0.72	0.66	0.77	0.77	0.28	0.17	**0.93**	0.37	0.64
3D array	0.94	**0.84**	**1.00**	**0.88**	**0.88**	**0.88**	**0.80**	**0.93**	**0.82**	0.92	**1.00**	**0.90**

Table 2 Per-gesture accuracies achieved by exploiting the concatenation feature vector $\mathbf{c}+\mathbf{d}$ (first row) and by the proposed three-dimensional feature array (second row). Results in the first row were obtained using the best configuration returned by a grid search over a number of parameters controlling sub-sampling and smoothing of **c** and **d**.

The proposed approach has also very limited computational requirements (see Table 3). On the $320x240$ depth maps produced by the Senz3D depth camera, the hand segmentation requires $3.1ms$ on average on a standard PC (an Intel i5-2430 CPU running at a $2.4Ghz$). The extraction of the features can be computed in only $0.83ms$ while SVM classification takes around $1ms$ on average. Summing up the three steps, the average total execution time is less than $5ms$ for each analyzed frame. This corresponds to a frame rate of $200fps$ that makes the approach very well-suited for the proposed system.

Step	*Execution Time*
Hand Segmentation	3.1 ms
Feature Extraction	0.83 ms
SVM Classification	1 ms
Total	**4.93 ms**

Table 3 Average execution time of the various steps of the proposed approach. The test have been performed on a standard PC with an Intel i5 processor.

6.2 Applications evaluation

The sample demos presented in Section 5 show the strength points of the developed system but there is plenty of room for further developments thanks also to the modular architecture of the proposed system that makes easy the construction of new applications.

The system has been tested by different people and in different settings and the testers, even with a short learning time, ended up with a positive review. It has also been tested in some very challenging environments: e.g., a trip on a paddle boat (Fig. 23a) and a roller-coaster ride (Fig. 23b) with high speed, bumps, sun light directly to the camera and moving objects all around. The system proved to be reliable also in these challenging settings. In particular the first test, performed in a paddle-boat was very successful with a smooth and nice visualization and it showed how the system is able to properly work in this environment. The roller coaster environment is very challenging due to the high speed and strong vibrations of the roller coaster. In this case the camera frame rate is not always sufficient to accurately handle the very fast motion but it is still possible to use the system. In particular the single frame gesture recognition algorithm was working also in this case. Another limitation is that the chosen depth camera is not suited for all the range of environments (specially the acquisition range is limited), but all the available depth acquisition devices and techniques have their issues as size, weight, range or reliability (e.g., 3D data from stereo vision systems is far less reliable). The system has been tested also at the *European Researchers' Night* with a large number of people (around 70) of different ages and different skill levels. The users have been asked frequently for eye-strain, headache or nausea, and while the usage time was limited to few minutes each, no problems were reported. The overall experience was highly appreciated, and while kids loved the games, adults could test a possibility for their future human-computer interface.

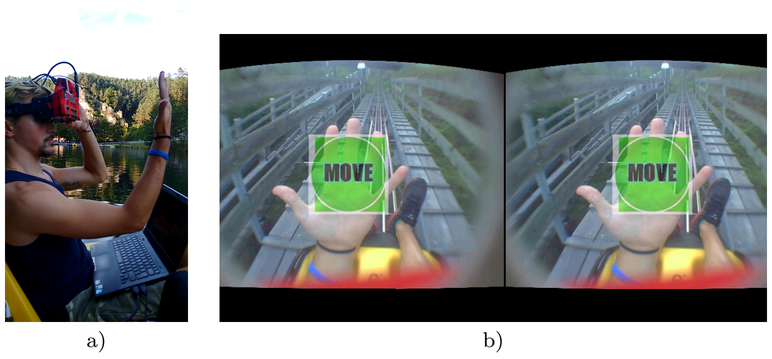


Fig. 23 Examples of the system in challenging environments: a) paddle boat; b) roller coaster.

7 Conclusion

In this paper a novel human-computer interaction system has been proposed. The exploitation of a depth camera has allowed to place the virtual elements into the desired locations in the real world and to use a novel interaction scheme based on hand gesture recognition. Critical issues like the calibration of the visualization and acquisition sub-systems have been considered and solved. Concerning gesture recognition, two new different feature descriptors have been developed ad-hoc for this problem, based on the contour curvature and on the shape thickness represented through a distance transform. By arranging them in a smoothed three-dimensional data structure we obtained a representation that proved to be discriminative and at the same time stable across different repetitions of the same gesture. The critical problem of the classifier training has been solved by developing a rendering application able to create realistic synthetic depth maps. The system has been tested in different augmented reality applications and in challenging outdoor environments and demonstrated to be very reliable. Experimental results showed that the proposed system enables an effective augmented reality experience to the test users. The gesture interface has been tested on a challenging dataset and obtained a 90% recognition accuracy in real-time. Further research will be devoted to the development of novel applications exploiting the proposed system and to the study of novel feature descriptors in order to improve the performances of the gesture recognition module. Furthermore the inclusion of a dynamic gesture recognition algorithm capable to account also for the temporal dimension besides the static pose will be considered.

Acknowledgements The authors would like to thank Ludovico Minto and Fabio Dominio for their help in developing some components of the gesture recognition system

References

1. Armesto, L., Tornero, J., Vincze, M.: Fast ego-motion estimation with multi-rate fusion of inertial and vision. The International Journal of Robotics Research **26**(6), 577–589 (2007)
2. Azuma, R.T.: A survey of augmented reality. Presence **6**(4), 355–385 (1997)
3. Bai, H., Lee, G., Billinghurst, M.: Using 3d hand gestures and touch input for wearable ar interaction. In: CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14, pp. 1321–1326. ACM, New York, NY, USA (2014)
4. Bambach, S., Lee, S., Crandall, D.J., Franchak, J.M., Yu, C.: Tracking hands of interacting people in egocentric video (2015)
5. Baraldi, L., Paci, F., Serra, G., Benini, L., Cucchiara, R.: Gesture recognition in egocentric videos using dense trajectories and hand segmentation. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 702–707. IEEE (2014)
6. Betancourt, A., López, M.M., Regazzoni, C.S., Rauterberg, M.: A sequential classifier for hand detection in the framework of egocentric vision. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 600–605. IEEE (2014)
7. Borgefors, G.: Distance transformations in digital images. Computer vision, graphics, and image processing **34**(3), 344–371 (1986)
8. Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., Ivkovic, M.: Augmented reality technologies, systems and applications. Multimedia Tools and Applications **51**(1), 341–377 (2011)
9. Colaço, A., Kirmani, A., Yang, H.S., Gong, N.W., Schmandt, C., Goyal, V.K.: Mime: Compact, low power 3d gesture sensing for interaction with head mounted displays. In: Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, UIST '13, pp. 227–236. ACM, New York, NY, USA (2013)
10. Dinh, D.L., Lee, S., Kim, T.S.: Hand number gesture recognition using recognized hand parts in depth images. Multimedia Tools and Applications pp. 1–16 (2014)
11. Dominio, F., Donadeo, M., Marin, G., Zanuttigh, P., Cortelazzo, G.M.: 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, pp. 9–16. ACM (2013)
12. Dominio, F., Donadeo, M., Zanuttigh, P.: Combining multiple depth-based descriptors for hand gesture recognition. Pattern Recognition Letters pp. 101–111 (2014)
13. Dominio, F., Marin, G., Piazza, M., Zanuttigh, P.: Feature descriptors for depth-based hand gesture recognition. In: Computer Vision and Machine Learning with RGB-D Sensors, pp. 215–237. Springer International Publishing (2014)
14. Hanheide, M.: A cognitive ego-vision system for interactive assistance. Ph.D. thesis, University of Bielefeld (2006)
15. Jaimes, A., Sebe, N.: Multimodal humancomputer interaction: A survey. Computer Vision and Image Understanding **108**(12), 116 – 134 (2007). Special Issue on Vision for Human-Computer Interaction
16. Kapuscinski, T., Oszust, M., Wysocki, M., Warchol, D.: Recognition of hand gestures observed by depth cameras. Int J Adv Robot Syst **12**, 36 (2015)
17. Kumar, J., Li, Q., Kyal, S., Bernal, E., Bala, R.: On-the-fly hand detection training with application in egocentric action recognition. In: Proceedings of HANDS CVPR Workshop, pp. 18–27 (2015)
18. Kurakin, A., Zhang, Z., Liu, Z.: A real-time system for dynamic hand gesture recognition with a depth sensor. In: Proc. of EUSIPCO (2012)
19. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 165–172. ACM Press/Addison-Wesley Publishing Co. (2000)
20. Liu, Y., Yang, J., Meng, Q., Lv, Z., Song, Z., Gao, Z.: Stereoscopic image quality assessment method based on binocular combination saliency model. Signal Processing **125**, 237–248 (2016)

21. Lv, Z., Halawani, A., Feng, S., Li, H., Réhman, S.U.: Multimodal hand and foot gesture interaction for handheld devices. ACM Trans. Multimedia Comput. Commun. Appl. **11**(1s), 10:1–10:19 (2014)
22. Lv, Z., Halawani, A., Feng, S., Ur Réhman, S., Li, H.: Touch-less interactive augmented reality game on vision-based wearable device. Personal and Ubiquitous Computing **19**(3-4), 551–567 (2015)
23. Marin, G., Dominio, F., Zanuttigh, P.: Hand gesture recognition with leap motion and kinect devices. In: Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 1565–1569. IEEE (2014)
24. Memo, A., Minto, L., Zanuttigh, P.: Exploiting silhouette descriptors and synthetic data for hand gesture recognition. In: Proceedings of Smart Tools and Apps in computer Graphics (2015)
25. Michel, D., Papoutsakis, K., Argyros, A.A.: Gesture recognition supporting the interaction of humans with socially assistive robots. In: Advances in Visual Computing, pp. 793–804. Springer (2014)
26. Nanni, L., Lumini, A., Dominio, F., Donadeo, M., Zanuttigh, P.: Ensemble to improve gesture recognition. International Journal of Automated Identification Technology (5), 47–56 (2013)
27. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
28. Premerlani, W., Bizard, P.: Direction cosine matrix imu: Theory. DIY DRONE: USA pp. 13–15 (2009)
29. Qin, S., Zhu, X., Yang, Y., Jiang, Y.: Real-time hand gesture recognition from depth images using convex shape decomposition method. Journal of Signal Processing Systems **74**(1), 47–58 (2014)
30. Ren, Z., Meng, J., Yuan, J.: Depth camera based hand gesture recognition and its applications in human-computer-interaction. In: Proc. of ICICS, pp. 1 –5 (2011)
31. Ren, Z., Yuan, J., Zhang, Z.: Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. pp. 1093–1096. ACM (2011)
32. Rogez, G., Khademi, M., Supančič III, J., Montiel, J.M.M., Ramanan, D.: 3d hand pose detection in egocentric rgb-d images. In: Computer Vision-ECCV 2014 Workshops, pp. 356–371. Springer (2014)
33. Serra, G., Camurri, M., Baraldi, L., Benedetti, M., Cucchiara, R.: Hand segmentation for gesture recognition in ego-vision. In: Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices, pp. 31–36. ACM (2013)
34. Supancic, J.S., Rogez, G., Yang, Y., Shotton, J., Ramanan, D.: Depth-based hand pose estimation: data, methods, and challenges. pp. 1868–1876 (2015)
35. Suryanarayan, P., Subramanian, A., Mandalapu, D.: Dynamic hand pose recognition using depth data. In: Proc. of ICPR, pp. 3105 –3108 (2010)
36. Šarić, M.: Libhand: A library for hand articulation (2011). URL `http://www.libhand.org/`. Version 0.9
37. Wan, S., Aggarwal, J.: Mining discriminative states of hands and objects to recognize egocentric actions with a wearable rgbd camera. In: Proceedings of HANDS CVPR Workshop, pp. 36–43 (2015)
38. Wang, J., Liu, Z., Chorowski, J., Chen, Z., Wu, Y.: Robust 3d action recognition with random occupancy patterns (2012)
39. Yang, J., Lin, Y., Gao, Z., Lv, Z., Wei, W., Song, H.: Quality index for stereoscopic images by separately evaluating adding and subtracting. PloS one **10**(12), e0145,800 (2015)
40. Zanuttigh, P., Marin, G., Dal Mutto, C., Dominio, F., Minto, L., Cortelazzo, G.M.: Time-of-Flight and Structured Light Depth Cameras: Technology and Applications, 1 edn. Springer International Publishing (2016). DOI 10.1007/978-3-319-30973-6. URL `http://www.springer.com/book/9783319309712`
41. Zhang, C., Tian, Y.: Histogram of 3d facets: A depth descriptor for human action and hand gesture recognition. Computer Vision and Image Understanding (2015)

42. Zhang, Z.: A flexible new technique for camera calibration **22**(11), 1330–1334 (2000)
43. Zhou, S., Fei, F., Zhang, G., Liu, Y., Li, W.J.: Hand-writing motion tracking with vision-inertial sensor fusion: Calibration and error correction. Sensors **14**(9), 15,641–15,657 (2014)