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

XKin: an Open Source Framework for Hand Pose and Gesture Recognition Using Kinect

Fabrizio Pedersoli · Sergio Benini · Nicola Adami · Riccardo Leonardi

the date of receipt and acceptance should be inserted later

Abstract This work targets real-time recognition of both static hand-poses and dynamic hand-gestures in a unified open-source framework. The developed solution enables natural and intuitive hand-pose recognition of American Sign Language (ASL), extending the recognition to ambiguous letters not challenged by previous work. While hand-pose recognition exploits techniques working on depth information using texture-based descriptors, gesture recognition evaluates hand trajectories in the depth stream by using angular features and hidden Markov models (HMM). Although classifiers come already trained on ASL alphabet and 16 uni-stroke dynamic gestures, users are able to extend these default sets by adding their personalized poses and gestures. The accuracy and robustness of the recognition system have been evaluated using a publicly available database and across many users. The XKin open project is available online [23] under FreeBSD License for researchers in human-machine interaction.

Keywords Kinect · Hand Pose · Gesture Recognition · Open-source · XKin · Human Computer Interaction

1 Introduction

During the last years efforts have been made in order to achieve more natural interactions between human and computers communication. In this context, gesture recognition capabilities play a key role in the design of innovative Human-Computer Interaction (HCI) systems. Among human body significant activities, the expressive power of hands will assume a pivotal role. In particular, more sophisticated gesture-based interfaces will enable the user to overcome the limitations of keyboard and mouse, increasing efficiency and realism, thus empowering users to interact with a computer in a more intuitive way.

Beyond the automatic interpretation of hand sign languages such as American Sign Language (ASL) [2], a wide range of real scenarios would benefit from novel paradigms of natural interaction, especially medicine, domotics, and the game industry. In addition to these, a gesture-based interaction system could boost users experience in all applications that suffer from the limitation imposed by traditional mouse and keyboard-based interaction, like 3D modeling and CAD/CAM.

Until few years ago the only non intrusive HCI technology able to accomplish the above tasks was vision-based hand-gesture recognition. Due to the introduction on the consumer market of low-cost devices such as the Kinect sensor, today it is possible to improve the tradeoff between performance and price in the design of gesture interfaces. Kinect approach is non intrusive, sensing is passive, silent and permits to overcome the limitation of robustness, speed and accuracy of typical image processing algorithms by combining color images and depth information. In fact the complementary nature of the depth and visual (RGB) information in the Kinect sensor opens up new opportunities to solve fundamental problems in human activity recognition for HCI.

So far the ability of the Kinect of enabling full-body 3D motion estimation has been exploited to track body joints, while little attention has been directed towards small gestures, in particular hand motions, especially because of the difficulty of the task.

Department of Information Engineering
University of Brescia, via Branze 38, 25125 Brescia, Italy
E-mail: firstname.lastname@ing.unibs.it

1.1 Paper aims and organization

In this work we propose the first (and to the best of our knowledge, the only available at the moment) open source solution targeting real-time recognition of hand-poses and gestures with Kinect sensor. As a first advance with respect to state of the art, the proposed method targets recognition of both static posture (*hand-pose* in the following) and dynamic movement (*hand-gesture* in the following) in a unified framework, while most of the existing methods focus either on static signs or on dynamic gestures. As a result, being able to classify both types of hand expressivity allows for understanding the most of the non-verbal language.

Regarding hand-poses, the adoption of multi-scale Gabor filters (as in [6] and [28]) applied on the depth image results in an effective description for recognizing ASL finger-spelling alphabet. Beyond obtaining high recognition accuracy on commonly recognized ASL letters by a Support Vector Machine classifier, we have also extended the recognition capabilities on the whole ASL finger-spelling alphabet, thus including also letters 'j' and 'z', which involve motion and that were not challenged by previous work. For what concerns dynamic hand trajectories, gesture recognition relies on the use of angular features on trajectories in the depth stream and hidden markov models (HMM). We then compare our recognition method on gestures against a geometric template matcher named $1 described in [40]on the same dataset of 16 uni-stroke gestures.

As a second novel contribution, although both hand-pose and gesture classifiers come already trained on two default datasets (i.e. the ASL alphabet, and the 16 uni-stroke dynamic gestures as in [40]), users are able to extend these default sets by adding their personalized poses and gestures. These personalized training procedures are proposed as a possibility for each user for "extending" the default sets of recognizable postures and gestures to those that he/she decide should be recognized by the system beyond the already provided sets, or for improving the recognition performance with respect to the standard non-personalized training.

Last, the proposed solution has been packaged in an open source framework since its first version described in [24], so that everyone can download, try it, and verify the declared performance. The code is freely available to the scientific community on *github* [23] to encourage and support contributions from other researchers and developers in building an open and effective system for empowering natural modalities for human-machine interaction.

Our proposed solution is complying with the following requirements of HCI: first we meet the *responsiveness* [37] criterium, by building a real-time an effective interaction. In addition, the system is usable by more users and not bounded to a particular one (*adaptability* requirement in [37]). Hand-pose in ASL and the 16 uni-stroke gestures in [40] are easy to perform and remember; they also present a clear cognitive association with the performed function (thus meeting *learnability* criterium, as in [37]). Beyond showing high recognition *accuracy* [37], at least for this prototyping phase, the proposed system also respects the *come-as-you-are* paradigm of interaction [37], not posing requirement on the user to wear marker, gloves, long sleeves, fix background or choose a particular illumination.

The rest of the paper is organized as follows. In Section 2 we explore recent advances in the use of Kinect for hand-pose and gesture recognition. Section 3 provides the overall methodology, while Section 4 discusses the details of the hand segmentation stage, which is preparatory for both pose and gesture classification. Hand-pose recognition employing depth information only is discussed in Section 5, while the statistical characterization of dynamic hand-gestures is provided in Section 6. Section 7 describes the user tests which have been carried out to assess performance and potentialities of the proposed framework for Kinect. Finally concluding remarks are gathered in Section 8.

2 Previous work

Despite the recent release of the sensor, some attempts to develop pose and gesture recognition systems employing Kinect have been already made in the past. The Kinect ability of providing full-body 3D motion capture has been initially exploited to track body joints, as in the work by Microsoft [35]. As an example, Biawas et al. [4] proposed a method to recognize human body postures showing fair results. The first operation consisted in the background removal from depth image using an auto-thresholding technique on the depth histogram. Then a grid was placed on the foreground, and the body pose parametrized using depth variation and motion information of each cell of the grid. Finally a multiclass Support Vector Machine (SVM) was used to train the system for the classification.

Further attempts started to shift the attention towards smaller gestures, in particular hand motions. Ren et al. in [30] and [31] developed a hand-pose recognition system which operates in uncontrolled environments and is insensitive to hand-pose variations and distortions, by using both depth and color information from Kinect. Hand segmentation is accomplished using the depth stream and requires the user to wear a black bracelet. For the hand-pose recognition a novel

shape distance metric called Finger-Earth Mover's Distance (FEMD) able to distinguish among 10 hand-poses was developed. In particular FEMD represents the hand shape considering each finger as a cluster, and the dissimilarity distance between two shapes is defined as the sum of work needed to move the earth piles and the penalty on the unmatched fingers.

As an attempt to recognize a larger set of hand-poses, [28] implemented "Spelling It Out", an interactive user interface for American Sign Language (ASL) finger-spelling recognition. Hand-poses corresponding to letters of the alphabet are characterized using appearance and depth images, and are classified using random forests. The best performance on 24 signs of the ASL, on a dataset collected on 5 users, reaches a mean precision of 75%. However, the system overcomes classification problems by offering an easy way for the user to select between ambiguous choices and is integrated with an English dictionary for efficient writing.

Also Uebersax et al. [36] presented an effective system for recognizing letters and finger-spelled words of the American Sign Language (ASL) in real time. The novelty of this method lies in the fact that letter classification, - based on average neighborhood margin maximization, - relies only on depth data coming from a Mesa SR400 TOF camera.

Another approach recognizing ASL letters by working only on depth information is described in [11]. Authors adapted the methodology of body pose estimation used in [35] to the hand, by using Randomized Decision Forests (RDF) for hand shape recognition. According to this scheme, every pixel is classified with a specific hand-pose label, and the final class label is determined by majority vote. At a price of a huge amount of training samples, they achieve high recognition accuracy.

Peris et al. [25] implemented a real-time pose recognition system for both hands reconstructing the 3D volume of the hand shape using a sequence of multi-view depth images acquired by Kinect. The hand detection is performed by using the Robot Operating System [32] and functionalities of OpenNI libraries [21] that extract the hand position in a 3D space as a point cloud. Then left and right hand volume sub-spaces are calculated by the Kernel Orthogonal Mutual Sub-space Method, a powerful algorithm for 3D object recognition that exploits the canonical angles between nonlinear sub-spaces generated by a kernel Principal Component Analysis (PCA).

Mihali et al.[17] proposed a robust hand-pose recognition algorithm that makes use of two Kinect sensors. This setup provides a rich point cloud within which the hand is detected considering the volume included between the closest point and a fixed empirical 3D offset. The volume is then subdivided into 6^3, 8^3, 10^3 evenly distributed voxels. For each sub-division two descriptors are taken, one related to the point count in each voxel, and the other related to presence of one or more pixels in each voxel. Nearest neighbor classifier in combination with majority rule voting scheme is used to recognize the unknown posture.

To improve accuracy in hand-pose recognition, some works started to target the problem of detection and tracking of single hand articulations. Li et al. [13] developed a system based on Kinect that is able to detect hand-poses, to identify single fingers, and to recognize nine postures executed both with one or two hands. Hands are distinguished from the background using a depth range between 0.5 m to 0.8 m and k-means algorithm is employed to obtain two clusters for hand pixels. Fingertips are detected applying the three-point alignment algorithm to points which are both on the hand-contour and on the convex hull. Finger names are determined according to their relative distance, while postures are recognized using a three layer classifier.

Oikonomidis et al. [19] proposed a novel approach to the problem of 3D tracking of hand articulations making use of Kinect. The problem was formulated as an optimization problem to minimize the discrepancy between the 3D structure and appearance of the hypothesized 3D hand-pose model. Optimization is performed by a variant of Particle Swarm Optimization, while hand extraction is achieved by skin color detection followed by depth segmentation.

Liang et al. [14] proposed a method to estimate 6 hand poses evaluating a simplified inverse kinematic of fingers. This method exploits temporal constraints and spatial features of the input depth sequence in order to detect the 3D fingertip position. Hand segmentation is performed through a pixel labeling procedure based on Bayesian inference, while fingertip localization is based on the geodesic extrema extraction and employs a novel path rewriting and K-means clustering.

A key challenge for hand-pose and gesture recognition is that they need to operate in complex scenes with cluttered backgrounds. Doliotis et al. [7] proposed a clutter-tolerant hand segmentation algorithm where 3D pose estimation is formulated as a retrieval problem: given a segmented hand the best matches are extracted from a large database of synthetically generated hand images. However, the need to solve optimization problems for computing the axis of elongation (Minumum Enclosing Ellipsoid) and the local regressions for smoothing the sequence of widths during the hand segmentation process, do not allow the algorithm to achieve real-time performance, by admission of the same authors.

Again Doliotis et al. [8] used Kinect to target dynamic hand-gesture recognition as a natural way of communication between humans and devices. They proposed a method to detect hands in complex and cluttered background, using the scene depth information from the Kinect, and a dynamic programming method, namely Dynamic Time Warping (DTW) for gesture recognition. According to this framework, the gesturing hand is detected by using a motion detection method based on frame differencing and depth segmentation. Trajectory recognition instead is performed using the nearest neighbor classification framework which uses the similarity measure returned by DTW.

With the aim to extend the set of recognizable gestures, Wan et al. [38] proposed a method to recognize five gestures: *start*, *no hand*, *left*, *right*, *back* using Kinect. The procedure consists of: gesture segmentation, feature extraction, trajectory extraction, and classification. A hand is detected through the depth thresholding of the closest object using a fixed empirical value that represents the thickness of the palm. To uniquely identify the gesture start, the algorithm uses a feature vector that takes into account the hand area, the number of non-zero pixels and the farthermost points of the convexity defects to the convex hull. The *left*, *right* and *back* gestures are identified by the trajectory in the (x, z) sub space, and classified by means of linear discriminant analysis. However one severe limitation of this work is the impossibility to detect vertical gestures.

Regarding software solutions, Microsoft released the Kinect device with a Software Development Kit (SDK) [16], which is oriented to skeleton and joint tracking, not considering specific hand movement recognition. As an alternative multi-platform solution, PrimeSense [27] has created NiTE [26], a middleware which provides almost the same features as the Microsoft SDK, but again constitutes a closed source software. In addition to this, NiTE is not a complete solution: in order to work it needs to be included in OpenNI, an open source framework which provides Application Programming Interfaces (APIs) for natural interaction applications. In addition NiTE also requires ad-hoc drivers to communicate with other devices.

Currently the proprietary solution from 3Gear [1] can be considered the state-of-the-art for hand-tracking software. In particular 3Gear focuses on a small number of simple hand poses and gestures (pinching and pointing poses in particular), so that at the moment poses outside of the database may be tracked poorly. The system, as largely described on the work in [39], works best when the user is seated at a desk and the camera (or the Kinect in the most recent release) is looking down at the user's hands. As a big advantage, this configuration allows the user to rest his hands on the desk most of the time. However this setup limits interactivity only to desktop scenarios, excluding for examples medical applications in operating rooms, interactive games (Kinect, Xbox, etc.), and all systems that are not constrained by the usage of a desktop station. By the same authors' admission, 3Gear system is not yet a general purpose hand-tracking system, since it solves the pose estimation problem by querying a precomputed database that relates hand silhouettes to 3D configurations.

Despite these proprietary high-level solutions, the open source community has been recently developing a low-level module for acquiring raw data stream from the Kinect: libfreenect [20]. Libfreenect is developed by the OpenKinect community and exclusively targets Kinect hardware. It represents a simple, clean and easy to use solution that provides the possibility to use the device with PCs running either GNU/Linux, MacOS or Windows. It comes also with many wrappers towards programming languages such as: C++, C#, Python, Java, Matlab and Ruby. Building up on libfreenect, we are able to acquire raw Kinect data streams and build up the first available open-source, reliable, and real-time solution to hand-pose and gesture recognition, as described in the following.

3 Overview of the system

In order to provide a complete solution to hand motions, we developed a system that recognize both poses and gestures. While with *pose* we intend the static appearance of the hand, a *gesture* is meant as a dynamic sequence that comprises the information related to the hand trajectory in time. Dynamic gesture assumes the presence of a sequence *hold-movement-hold* (using the terminology of [15]), which many sign phonologists have claimed to be the sequence of three elements necessary to account for the form of a sign in which the hand moves from one location to another. Examples of different hand-poses are given in Figure 1-a, while examples of hand-gestures are provided in Figure 1-b.

The proposed system consists of three main modules, each specifically dealing with a defined problem: hand segmentation (Figure 2-a), hand-pose classification (Figure 2-b) and gesture recognition (Figure 2-c).

The hand segmentation stage relies only on depth information supplied by the Kinect and provides as output a depth image with the segmented hand, while other objects and background are removed. Depending on the recognition goal, the obtained depth hand image acts as input of either the hand-pose processing chain or the hand-gesture classifier.

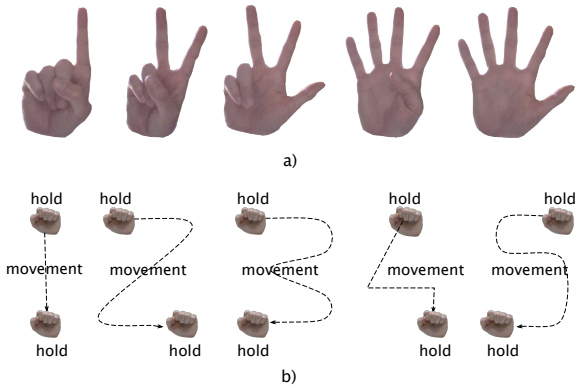

Fig. 1 Examples of a) different hand-poses and b) different hand-gestures. Note that gestures are temporally included within a sequence hold-movement-hold [15], independently by the hand shape assumed during the gesture.

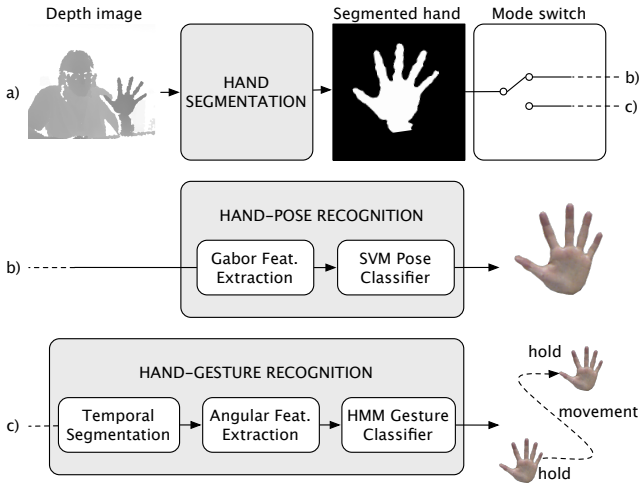

Fig. 2 Overview of the system workflow and its principal modules: a) hand segmentation, b) hand-pose classifier, and c) hand-gesture classifier.

In order to distinguish among a number of different postures, the hand-pose module first extracts from the depth image features based on multi-scale Gabor filtering which provide insights on hand local patterns. Learning and classification are then performed by using a Support Vector Machine (SVM).

The hand-gesture module instead receives as input a continuous stream of depth hand-segmented images. Classification is then performed by means of HMM adopting a description of gesture trajectories based on angular features computed between successive trajectory points extracted from the depth stream.

The switch between the two modalities (pose vs. gesture) depends on the higher level application making use of the framework libraries. As an example, a word processor application for the composition, editing, formatting of written material would make better use of the hand-pose module, to translate into text the performed ASL finger-spelling. Conversely, in medical applications or in domotics, the adoption of gesture-based commands would be more convenient, as an example, for browsing tomographical content, or to act as handy remote controllers, respectively.

Currently the two classifiers are released as independent modules. Therefore there are no technical constraints avoiding the two of them to run simultaneously and returning a sequence of distinguished poses captured while the hand performs a dynamic gesture. Depending on the final application, it will be required to develop a higher logical module for combining the outputs of the two classifiers in the desired manner.

4 Hand segmentation

The process of segmenting the user's hand is depth-based, as shown in Figure 2-a. The role of this procedure is critical since it provides a hand depth image which is a fundamental input for both pose and gesture classifications. This segmentation processing has to be fast in order to be employed in a *real-time* application, and *robust* to illumination changes, skin color variations, and user's position. Furthermore, this method must not pose any particular limitations on the user, in order to meet the *come-as-you-are* paradigm of interaction. The only obvious constraint in order to successfully complete this step is that no large object should be interposed between the user and the Kinect. The hand segmentation process is composed by two principal steps: a mean shift segmentation, followed by a palm detection procedure. Examples of the outputs of the different stages of the hand segmentation process are given in Figure 3.

4.1 Mean shift segmentation

The mean shift algorithm [5] performed on the depth data constitutes the first step of the hand segmentation process. This algorithm is a non-parametric clustering technique which does not require prior knowledge of the number of clusters. Its peculiar ability lies in being attracted by the principal modes (local maxima) of an underlying density function. Aiming at segmenting the depth image, we apply it to the empirical probability density function (pdf) of the depth image. Starting from a full tessellation of the feature space by Gaussian windows with profile as in [5]:

$$k_N(x) = \exp\left(-\frac{1}{2}x\right), \qquad x > 0 \tag{1}$$

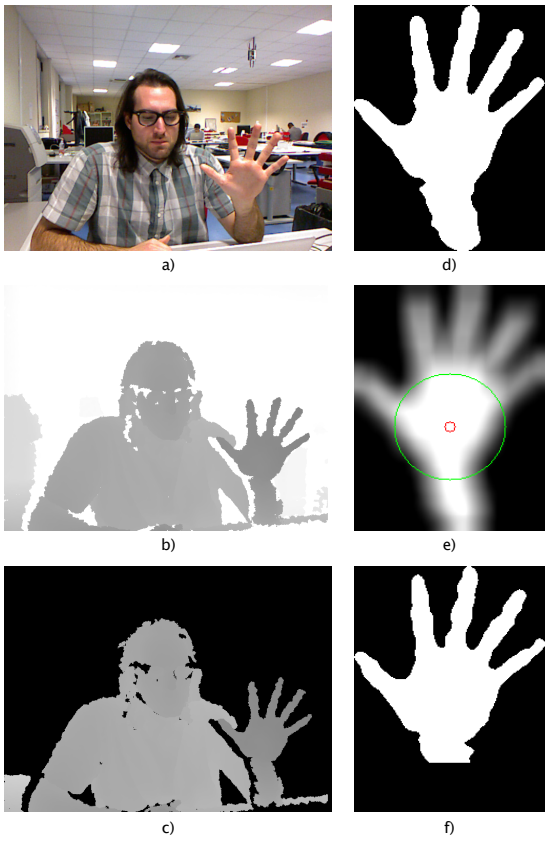

Fig. 3 a) Reference rgb image (not used), b) corresponding depth map, c) depth image after Otsu's thresholding, d) image $\tilde{H}$ containing a roughly segmented hand by mean shift segmentation, e) density image D and palm detection by circle fitting, f) finally segmented hand region (image H).

the algorithm forces the kernel windows to evolve towards the modes of the considered depth density function by iteratively computing the mean shift vector $m(x)$ as the difference between the weighted mean of the density in the kernel window, and x, i.e. the center of the kernels:

$$m(x) = \frac{\sum_i g(\| \frac{x-x_i}{h} \|^2) \cdot x_i}{\sum_i g(\| \frac{x-x_i}{h} \|^2)} - x \qquad (2)$$

where $g(x) = -k'_N(x)$ and h is the bandwidth. The mean shift vector thus always points toward the direction of maximum increase in the density. By associating each pixel with its significant mode, we are able to segment a rough hand region as constituted by all pixels associated to the mode at minimum mean depth. As a result, the returned rough hand region is stored as a binary image $\tilde{H}$, as shown in Figure 3-d.

Because the classic mean shift algorithm is time intensive, in order to meet the requirement of having a real-time interaction, two practical schemes are adopted. First the depth density function is computed after a masking process by Otsu's algorithm [22] for isolating the foreground depths from the background ones (Figure 3-c). Second, the mean shift is not run on all single depth frames, but on a periodical update interval of 1s. Under the realistic assumption that the number of principal modes of the pdf does not abruptly change within the update interval, the number of k principal modes returned by the mean shift algorithm is exploited to run an efficient k-means clustering algorithm on each depth frame contained in the update interval.

4.2 Palm detection

In order to better isolate the hand from the rest of the forearm, the hand image in $\tilde{H}$ is then refined by estimating the largest circle fitting the palm region. By assuming that the palm should have a higher density than the finger and the forearm areas, we first compute the density image D as the convolution between $\tilde{H}$ and a Gaussian kernel G. By thresholding in D high density regions, we select the center for circle fitting as the centroid of the densest one, as shown in Figure 3-e. Then the circle fitting procedure expands a circular window from the region centroid until a large majority of the points inside it belongs to $\tilde{H}$. Then, similarly to the mean shift algorithm, the center is shifted towards the direction that maximizes the density in the window. The expansion and shifting process is iteratively repeated until the largest fitting circle is found. Depending on both the orientation of the principal axes of the rough hand in $\tilde{H}$ and the mutual position of fingers and forearm with respect to the isolated palm, the information provided by the palm detection is exploited to cut the hand region by the wrist, obtaining the final segmented hand region H (as in Figure 3-f).

5 Hand-pose Recognition

When the objective is to distinguish among many non-trivial poses (as those belonging to the ASL alphabet shown in Figure 4), a representative set of features for describing the hand shape is needed. The hand-pose features we use are based on Gabor filtering of the depth image, while the learning and classification stage of the hand pose is performed by a SVM classifier, as shown in Figure 2-b. Operations of feature extraction are fastened by restricting the processing to the small region identified by the minimum bounding box that fully encloses the hand depth image.

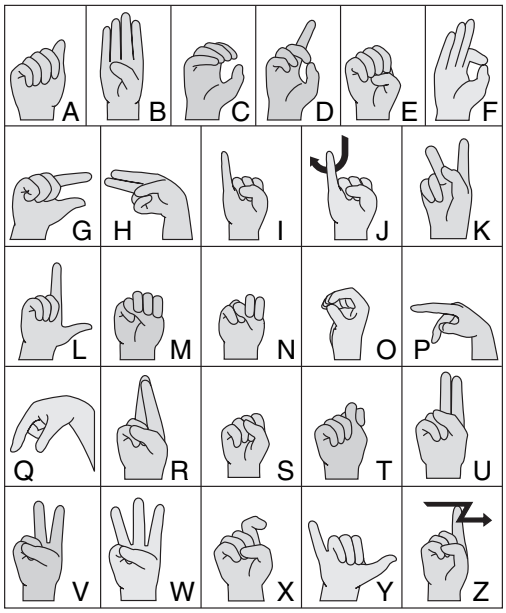

Fig. 4 ASL finger-spelling alphabet (reproduced from [3]).

5.1 Gabor features

The application of Gabor filters with different frequencies and orientations allows to reveal local patterns in images. Their use in image analysis is supported by the fact that they model the response of the receptive fields of the orientation-selective simple cells in the human visual cortex [6]. In 2D-images a Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave, that is:

$$g(x, y, \lambda, \theta, \varphi) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right) \cos\left(\frac{2\pi x'}{\lambda} + \varphi\right) \tag{3}$$

where $1/\lambda$ is the spatial frequency, φ the initial phase, $x' = x\cos\theta + y\sin\theta$, $y' = -x\sin\theta + y\cos\theta$, where θ is the orientation. Note that the spatial frequency and the variance σ^2 are not completely independent: in our experiments we use $\sigma = 0.56\lambda$ (corresponding to bandwidth $b = 1$, see [6] for details).

In order to extract the local patterns in the image, a filter bank consisting of Gabor filters with various scales and rotations is created with a procedure similar to [28], but performed only on depth information. Starting from the isolated hand in image H, the hand bounding box with the depth data is resized to 128×128 and convolved with a bank of Gabor filters at 4 scales (s=1/2, 1/4, 1/8, 1/16) and 4 orientations ($\theta = 0, \pi/4, \pi/2, 3\pi/4$). Filter responses are then averaged by 8×8 overlapping Gaussian functions positioned on the 16 resulting images on a regular grid in order to obtain the resulting feature vector of 1024 elements used for classification.

5.2 SVM classification

Hand-pose recognition is performed by using a SVM classifier. Since the system is supposed to work in a real-time scenario, the classified hand-pose is determined as the most recurrent pose within a classification sliding window on temporal frames, which ensures the system performance even in the most difficult conditions, such as those with fast hand movements or rapid pose changes. An example of the sytem setup in a real-time scenario is given in Figure 5.

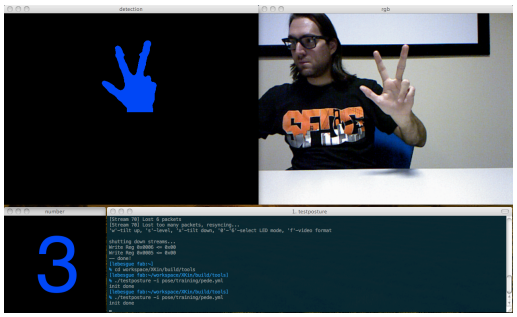

Fig. 5 System set-up in a real-time scenario. In the bottom-left corner, the number corresponding to the currently performed hand-pose (in this case number '3') is shown.

For each hand-pose a multiclass SVM is trained on the combined feature set using the one-against-one approach, thus creating the models for the classification task. For each SVM, the penalty term C and parameter ξ of a standard RBF kernel $K(x, y) = \exp(-\xi \|x - y\|^2)$ are obtained performing cross-validation on the training set via a process of grid search to maximise cross-validation accuracy. The best couples $(\hat{C}, \hat{\xi})$ are then used to learn the training sets and generate the final models.

6 Hand-gesture Recognition

An accurate hand-gesture classification requires both a correct temporal segmentation of the action and an accurate recognition of hand trajectory. For what concerns the temporal segmentation, for each hand-gesture we look for the presence of a *hold-movement-hold* sequence: the three phonological elements necessary to account for a gesture, according to [15]. Therefore, while in our previous work [24] a gesture was included in the temporal interval between two specific poses (start and

a stop-pose, respectively), in the proposed solution gesture segmentation is based on measures of motion activity. In order to evaluate recognition performance we adopt the set of 16 uni-stroke gesture types [40] shown in Figure 6, and perform gesture classification by means of Hidden Markov models working on trajectory angular features of the hand centroid.

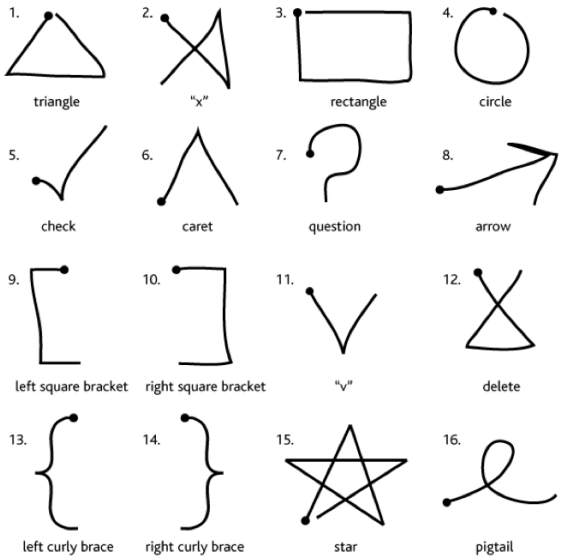

Fig. 6 The set of 16 uni-stroke gestures used in several works, such as in the $1 system (reproduced from [40]).

6.1 Temporal segmentation

According to the *hold-movement-hold* phonological sequence, we assume that a person interposes a punctuating pause in the middle of two distinct gestures. The localization of the temporal endpoints of a gesture relies on spotting the presence of local minima in the total activity of the sequence, as proposed by Le et. al [12]. With respect to other methods for gesture segmentation based on the location of peaks of motion activities [11], the adopted method which works on depth data only, ensures higher performance (recall 94%, precision 98% on CHALEARN dataset [10]).

6.2 Trajectory description

Once the gesture has been segmented, we extract angular features from the hand trajectory, intended as the sequence of points corresponding to the hand centroid in consecutive frames. However, if we regularly sample a real-time acquired gesture sequence, this is constituted by points that are not evenly distributed along the spatial trajectory. This is due to the acceleration introduced by the user when performing a gesture that requires big changes of direction (e.g. for gestures such as 'triangle', 'delete', etc. in Figure 6). If we extracted angular features from one of these trajectories, feature values would not faithfully reflect the characteristics of the ideal gesture evolution. Therefore, a resamplig step similar to that proposed in [40] ensures that the entire sequence of trajectory points is resampled at equal distance.

From the point sequence of the resampled hand trajectory, we extract two angular features which are invariant to position and scale. In fact, the trajectory itself, if taken as a sequence of points corresponding to the hand centroid $\{(x_t, y_t),\ t = 0, 1, \ldots, T-1\}$ in consecutive samples, does not satisfy the property of invariance to translation and scale.

The first proposed feature is given by the sequence of angular variations between the hand centroid in consecutive samples, as shown in Figure 7.

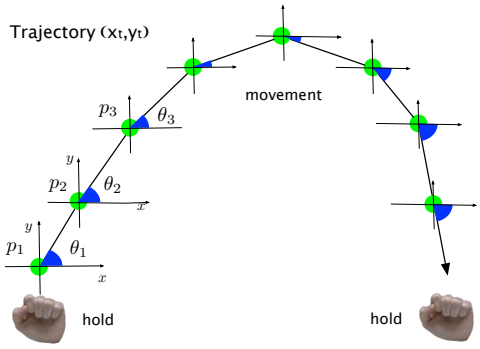

Fig. 7 An example of a gesture trajectory, characterized by a sequence *hold-movement-hold* and the angular variations θ_i.

By computing from the sequence of T centroids the sequence of $(T-1)$ angles between consecutive couples of samples

$$\theta_t = \text{atan2}\,(y_t - y_{t-1}, x_t - x_{t-1}) \quad t = 1, 2, \ldots, T-1 \tag{4}$$

each angle θ_t is quantized to a value θ^q among a number of sixteen possible directions:

$$\theta^q = i \in [1, 2, \ldots, 16] \ \text{ if } \ \theta_t \in \left[\frac{2\pi i}{16}, \frac{2\pi(i+1)}{16}\right] \tag{5}$$

so that the first feature is $\bar{\Theta} = [\theta^q_1, \theta^q_2, \ldots, \theta^q_{T-1}]^T$. The second feature is defined as the integral of the quantized angle differences:

$$\psi_t = \sum_{i=1}^{t} \theta^q_i, \quad t = 1, 2, \ldots, T-1 \tag{6}$$

which introduces a memory element in the trajectory description. As a result, the sequence of trajectory points is represented by a two dimensional feature vector in which each element represents the angle and the summation of difference of all previous angles.

6.3 Classification by HMM

The classification of a gesture trajectory is accomplished by means of hidden Markov models (HMM) [29]. These are discrete state-space stochastic models which work well for temporally correlated data streams, where the observations are a probabilistic function of a hidden state. The idea in this case is to break up a trajectory into different traits of homogenous motion direction, and model each trait by a different HMM state, so that the total number of HMM states corresponds to the number of peculiar traits of the gesture. Possible transitions from one state are: towards the same state, if the hand trajectory endures in the same trait (with high probability between consecutive frames), or towards the state that describes the next trait (with lower probability) in case of changed direction. For example imagining the trivial example of a gesture that consists of a straight horizontal line, the trajectory is made up of only one homogeneous trait, corresponding to a single-state HMM.

Due to the fact that each gesture follows a deterministic path through the HMM states, this choice gives us the possibility to model a trajectory in a way that does not depend on the length of the sequence, allowing us to classify gestures with invariance with respect to the execution speed. In particular we model each gesture with a left-right hidden Markov model $\Lambda = (A, b, \pi)$, endowed with the following distributions:

- the *state transition probability distribution* A is left-right initialized:

$$A = \begin{bmatrix} a & b & c & d & \cdots & e \\ 0 & f & g & h & \cdots & i \\ \vdots & \vdots & \vdots & \vdots & l & m \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

  and initial probability values are uniformly distributed (summing up to one);
- the *observation symbol distribution* b is initially uniformly distributed among all the possible observation symbols, because the system is not aware of the trajectory direction that will be gestured;
- the *initial state distribution* is uniformly distributed among all states.

To train the HMM, the Baum-Welch re-estimation algorithm [29] is applied. Classification is finally performed by means of the forward algorithm [29] by evaluating the maximum likelihood of an unknown hand-gesture sequence against all HMM models.

7 Experimental results

To evaluate the performance of the proposed system, we have carried out a series of experiments on both hand-pose and hand-gesture recognition. The first test considers a hand-pose scenario of 24 different letters of the ASL dataset in [28], as those shown in Figure 4. With respect to hand-gesture recognition, the two performed experiments employ the 16 uni-stroke gestures used in [40] and shown in Figure 6, which we analyze in both single-user and multi-user scenario. We then compare our recognition method on gestures against a geometric template matcher named $1 on the same dataset [40]. In the last experiment, exploiting our unified framework for hand-pose and gesture classification, we extend the recognition capabilities on the whole ASL finger-spelling alphabet, thus including also letters ‘j‘ and ‘z‘, which involve motion and that were not challenged by previous work.

7.1 Test I: Hand-pose recognition

In order to compare the system capabilities against previous work, we carried out the following experiment on the publicly available ASL dataset used in [28]. It consists of 65,000 depth and color hand images acquired from Kinect stream, corresponding to 24 of the 26 ASL letters (omitting non-static letters ‘j‘ and ‘z‘) performed by five subjects, with a good variety in size, background and orientation. In our work we keep only depth images and disregard the color ones. Pugeault et al. [28] report their results on this dataset using both leave-one-subject-out cross-validation and by using half of the set for training and half for testing employing multi-class random forest classification. Therefore we similarly split the entire database randomly in training and testing using a ratio of 0.5, learning the models on the training part and classifying the testing half, and then repeat the experiment using the same cross-validation procedure. Obtained performance achieves 91% of recognition rate for the half vs. half split, and 56% for the leave-one-subject-out cross-validation, respectively. A comparison of the proposed method based on Gabor filtering and SVM classifier with the latter method is provided in Table 1, showing the superiority of our presented

approach. With respect to the work in [28], our better performance are probably justified by the fact that Pugeault et al. do not operate background subtraction before the hand-pose classification stage.

Table 1 Comparison of our method with the approach in [28].

Method	Data	50% vs. 50%	Cross-valid.
[28]	color & depth	75%	47%
Our	**depth**	**91%**	**56%**

Table 2 details the confusion matrix for all letters using the proposed SVM radial classifier. This shows that the most significant amount of confusion comes from letters 'o' (87%), 's' (83%), 't' (84%), 'u' (86%), while the most certain classes are 'b' (96%), 'i' (97%), and 'y' (97%).

For the purpose of comparison with other state-of-the-art methods, we have also tested categorization performance on the same database using the Random Forest Classifier, as performed in [11]. In this experiment we achieve a recognition rate of 79.0% for the half-half configuration, while [11] report 97.8%. Regarding the performance using leave-one-subject-out cross-validation we score a recognition rate of 46.0%, while [11] reports 84.3%. Per letter details can be inspected in the confusion matrix of Table 3, where definitely inferior performance with respect to those declared in [11] (and with respect to those obtained by our method) are probably due to different choices in the classifier parameter configuration.

Last our proposed method, showing average recognition accuracy (ARR) of 91%, slightly outperforms the approach presented in [36], which employs a MESA SR4000 TOF camera, average neighborhood margin maximization (ANMM) on segmented hand depth images, and declares an ARR of 88%, even if on a different database.

7.1.1 Limitations

Such as all classifiers, our proposed pose-recognizer has limitations. First the candidate hand-pose are compared against previously trained SVM classes. Therefore, if the user performs an unknown hand-pose, the system will anyway return the most likely one. Second, the hand segmentation process is critical to the entire process. Despite our approach does not require any manual segmentation or aid by bracelets, gloves, markers or other invasive tool, any error in the palm detection procedure inevitably harms the hand-pose classification. Third, the selected features for hand-pose description are not rotation invariant. Last, in real-time pose classification, the optimal choice of the parameters such as the length of the sliding window on frames used for classification, and the related delay for returning the classification result in a usable interactive interface, still needs an accurate investigation.

7.2 Test II: Hand-gesture recognition

To test the hand-gesture classifier, ten users are asked to perform the 16 gestures in Figure 6 ten times each, by using indifferently his/her right or left hand. Having ten users performing 160 gestures each, this results in a total amount of 1600 gestures available for classification, showing high variability in hand-writing style among users.

Table 4 shows the confusion matrix by using for each gesture a half vs. half split of the database, therefore in a multi-user scenario (i.e. using a unique training set for all users).

Table 4 Confusion matrix for the gesture classification task using angular features and HMM on the dataset with 16 uni-stroke gestures as in [40] in the multi-user scenario (one training set for all users).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**1.0**															
2		**.83**												.17		
3			**1.0**													
4	.27			**.57**					.17							
5					**.33**			.30		.07						.30
6					.07	**.90**										.03
7					.07		**.60**		.10						.23	
8					.03			**.53**							.30	.13
9	.10			.13					**.33**				.43			
10										**.93**				.07		
11		.03			.17						**.80**					
12											.20	**.73**		.07		
13	.17			.17					.30				**.37**			
14					.27					.03		.03		**.67**		
15															**1.0**	
16					.10			.17						.03		**.70**

Inspecting Table 4, it is noticeable that among the 16 gestures there are at least 5 classes which are above 90% of recognition rates (i.e. 'triangle', 'rectangle', 'caret', ']', 'star'), while 3 gestures are far from being acceptable, since below the level of 50% of recognition rates, that are 'check', '[', and '{'. While it was easily predictable that simple uni-stroke gestures such as 'triangle', 'rectangle', 'caret' would correspond to the most certain classes, the high recognition rate on the 'star' gesture is probably due to the fact that no other signs is characterized by such a high number of traits (five) along different directions. Conversely, it looks rather

peculiar that while class ‘]‘ has a high recognition rate (93%), its counterpart class ‘[‘ performs so badly (33%).

Table 5 shows instead the confusion matrix obtained when each user trains the classifier with his/her own training sequences (single-user scenario), averaged on all users. In particular we adopted a leave-one-out procedure for each user gesture, thus using nine of the ten recorded sequences as training sequences, and the remaining one as test set, and repeating the procedure for all sequences.

Table 5 Confusion matrix for the gesture classification task using angular features and HMM on the dataset with 16 uni-stroke gestures as in [40] in the single-user scenario (personalized training set defined by each users).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**.97**		.03													
2		**.98**										.01		.02		
3			**.98**								.01	.01				
4	.22			**.66**					.09				.03			
5		.02			**.58**			.14		.01	.05	.15		.02		.03
6					.06	**.66**		.20							.08	
7							**.52**	.02	.02				.02	.05	.38	
8					.09	.02		**.83**							.05	.02
9	.02			.19					**.54**				.25			
10										**.98**				.02		
11		.10	.01		.01						**.78**	.10				
12		.07	.02								.40	**.50**				.02
13	.24		.01	.22					.17				**.36**			
14		.06								.46	.01	.02		**.46**		
15						.03		.15							**.79**	.02
16					.10	.02		.21		.05		.03		.02	.01	**.56**

Inspecting Table 5, we can observe that the confusion on the least recognized gestures in Table 4 is mitigated in case of the single-user scenario, i.e. when users operate a personalized training on each gesture. In fact out of the 3 gestures below the level of 50% of recognition rates, (‘check‘, ‘[‘, and ‘{‘) only ‘{‘ is still below 50%, while the other two have considerably increased their recognition rates: ‘check‘ from 33% to 58% (+25%) , and ‘[‘ from 33% to 54% (+21%). Other considerable improvements (>10%) are reported for gestures ‘x‘ (+15%), ‘arrow‘ (+30%). Conversely, gesture ‘}‘ which in Table 4 scored 67% has now fallen under 50% (46%), which we admit is still far from being acceptable even for a prototype gesture classifier. One possible reason has to be found in the dimensions of the training set which, in case of personalized learning, are considerably smaller with respect to the training samples in the multi-user scenario (i.e. 9 samples only, instead of 80 in the half vs. half configuration).

Finally we compared our HMM-based gesture classifier with a geometric template matcher named $1 [40], which is a popular uni-stroke recognizer previously used in HCI, which is supposed to outperform other gesture classifiers (such as DTW [8] [18], and the Rubine classifier [33]). In particular we integrated the code of $1 classifier (available here [34]) in our framework, thus replacing our hand-gesture classifier. As training set, $1 needs only one template for each of the 16 uni-stroke gestures. Using the same test set as in our multi-user scenario, we obtained with $1 classifier the confusion matrix reported in Table 6.

Table 6 Confusion matrix for the gesture classification task using $1 system classifier [40].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	**.86**	.02	.10	.02												
2		**.34**	.06					.08		.02	.02	.14			.02	.32
3			**.98**	.02												
4				**1.0**												
5	.10	.02	.68		**.02**						.02	.04	.06			.06
6						**.22**						.78				
7						.02	**.86**					.12				
8						.06		**.20**				.74				
9	.02		.90		.06				**.02**							
10						.02	.90			**.08**						
11	.86		.04								**.10**					
12	.02						.54					**.44**				
13	.16	.04	.26						.06		.32		**.04**			.12
14		.02	.02			.04	.22			.04	.02	.54		**.06**		.04
15								.02							**.98**	
16		.02								.02		.78				**.18**

Inspecting Table 6, it is possible to observe that among the 16 gestures there are only 3 classes which are above 90% of recognition rates (i.e. ‘rectangle‘, ‘circle‘, ‘star‘), while 11 gestures are far from being acceptable, since below the level of 50% of recognition rates. By comparing our HMM-based method (in Table 4) and the $1 classifier (in Table 6) on the same multi-user scenario, our proposed hand-gesture classifier outperforms $1 matcher on 14 of the 16 uni-stroke gestures. Although $1 is a popular and fast system, the last test shows that this approach has some important limitations. Beside the inherent weakness of using a single template, $1 works on the “raw” sequence of points, without representing it in a feature space. Thus the similarity measure is pretty simple, since it depends only on the geometrical similarity between the sequence and the template. Second, the non-uniform scaling of gesture to a squared area can strongly alter the gesture appearance.

7.2.1 Limitations

Such as all classifiers, our proposed gesture recognizer has inherent limitations. First the candidate gestures are compared to previously stored models, and the re-

sult produced is the one that maximizes the likelihood score. Therefore, if the user performs an unknown gesture, the system will anyway return the most likely one. Second, because of the resampling procedure, gestures cannot be differentiated on the basis of the hand speed while gesturing.

7.3 Test III: Whole ASL alphabet

As a novel contribution, exploiting our unified framework for hand-pose and gesture classification, we extend the recognition capabilities to the whole ASL finger-spelling alphabet, thus proposing specific classifiers for letters 'j' and 'z', which involve motion and that were not challenged by previous work. This extension is possible by running simultaneously both hand-pose and gesture recognition modules, and by switching from pose to gesture recognition by means of a simple motion activated mechanism which uses two cutoff thresholds in a twin-comparison fashion [41]. Doing so, ASL letter 'j' is recognizable by the initial presence of a 'j' hand-pose (which is the same as for letter 'i') followed by a 'j'-letter related trajectory. Similarly, ASL letter 'z' is recognized as a combination of an initial 'z' hand-pose (which is the same as for letter 'd') and the following 'z' related trajectory. To test the 'j' and 'z' classifiers, the same ten users of the previous test (non-native to ASL) are asked to perform the 2 letters 'j' and 'z', ten times each. Having ten users performing 20 letters each, this results in a total amount of 200 letters available for classification. Table 7 illustrates the average recognition accuracy for ASL 'j' and 'z' by using a half vs. half database configuration in a multi-user scenario.

Table 7 Average Recognition Rates (ARR) for motion-involving ASL letters 'j' and 'z'.

	ARR
j	**.87**
z	**1.0**

7.3.1 Limitations

With respect to the previous experiment, lower performance in recognition of letter 'j' are probably due to inter-user differences in performing the sign, especially because the initial pose rapidly changes during the trajectory evolution with a hand rotation movement.

Full ASL alphabet recognition (including 'j' and 'z') is further complicated in real-time usage by a severe synchronization issue, due to the fact that both hand-pose and gesture classifier are concurrently running. As an effect, ambiguities might arise for example in the interpretation of a punctuating pause (i.e. a 'still' posture) which can be both classified with the associated hand-pose, or exploited by the gesture classifier to spot a temporal endpoint of a gesture. This synchronization problem is partially mitigated by the use of a twin-comparison method [41], which employs a double threshold mechanism (i.e. two distinct thresholds to be satisfied at the same time: one on frame difference and another on the accumulated frame difference) in order to activate the gesture recognition module. Another limitation due to synchronization is the need to delay the classification output until motion triggers (or inhibits) the gesture classifier for solving the ambiguities between letter couples "i, j", and "z, d" (which both share the same pose, and are only distinguished by the following motion trajectory).

8 Conclusions

We have here developed the first open source package for Kinect sensor device which targets both hand-pose and gesture recognition. By relying only on depth information, we have created an accurate, real-time, and adaptable solution for empowering natural interaction with computer devices, which enables recognition on full ASL finger-spelling alphabet and a rich set of 16 uni-stroke gestures. Beyond these two sets of recognizable hand movements, the developed framework also allows the user to define and train the system with his/her own set of hand poses and gestures. This solution respects the *come-as-you-are* paradigm of interaction, not requiring the user to wear markers, gloves, or long sleeves. The accuracy and robustness of recognition have been tested across many users and compared with other state of the art solutions by using publicly available datasets. The overall high performance and robustness of hand-pose recognition (average recognition rates above 90% on 24 poses) and hand-gesture classification (average recognition rates above 70% on 16 gestures) allow to open up to promising capabilities in the design of innovative natural interaction applications. Moreover the provided libraries represent a complete novelty in the open source community, since similar applications do not exist so far even under the form of proprietary software solutions. By making available the developed API under FreeBSD License, we encourage contributions from other researchers and developers in building an open and effective system for empowering natural modalities of human-machine interaction. Further research attempts will be devoted to extend the current approach to accurate full-body gesture, in

order to learn and recognize human activity and face challenging data such as those in CHALEARN set [9].

References

1. 3Gear systems: Gestural user interfaces. http://www.threegear.com/ (2013)
2. American sign language. http://en.wikipedia.org/wiki/American_Sign_Language (2013)
3. Gibson Hasbrouck & Associates. http://www.gha-pd.com/ (2013)
4. Biswas, K., Basu, S.: Gesture recognition using Microsoft Kinect ™. In: Automation, Robotics and Applications (ICARA), 2011 5th International Conference on, pp. 100 –103 (2011). DOI 10.1109/ICARA.2011.6144864
5. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24**(5), 603–619 (2002). DOI 10.1109/34.1000236
6. Daugman, J.G.: Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America A: Optics, Image Science, and Vision **2**(7), 1160–1169 (1985)
7. Doliotis, P., Athitsos, V., Kosmopoulos, D.I., Perantonis, S.J.: Hand shape and 3d pose estimation using depth data from a single cluttered frame. In: G. Bebis, R. Boyle, B. Parvin, D. Koracin, C. Fowlkes, S. Wang, M.H. Choi, S. Mantler, J.P. Schulze, D. Acevedo, K. Mueller, M.E. Papka (eds.) International Symposium on Visual Computing (ISVC). Springer, Springer (2012)
8. Doliotis, P., Stefan, A., McMurrough, C., Eckhard, D., Athitsos, V.: 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. ACM (2011). DOI 10.1145/2141622.2141647. URL http://doi.acm.org/10.1145/2141622.2141647
9. Escalera, S., Gonzàlez, J., Baró, X., Reyes, M., Lopes, O., Guyon, I., Athitsos, V., Escalante, H.: Multi-modal gesture recognition challenge 2013: Dataset and results. In: Proceedings of the 15th ACM on International conference on multimodal interaction, p. 445–452 (2013). URL http://dl.acm.org/citation.cfm?id=2532595
10. Guyon, I., Athitsos, V., Jangyodsuk, P., Hamner, B., Escalante, H.: Chalearn gesture challenge: Design and first results. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on, pp. 1–6 (2012). DOI 10.1109/CVPRW.2012.6239178
11. Keskin, C., Kirac, F., Kara, Y., Akarun, L.: Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In: ECCV12, pp. VI: 852–863 (2012)
12. Le, T.L., Nguyen, V.N., Tran, T.T.H., Nguyen, V.T., Nguyen, T.T.: Temporal gesture segmentation for recognition. In: Computing, Management and Telecommunications (ComManTel), 2013 International Conference on, pp. 369–373 (2013). DOI 10.1109/ComManTel.2013.6482422
13. Li, Y.: Hand gesture recognition using Kinect. In: Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on, pp. 196 –199 (2012). DOI 10.1109/ICSESS.2012.6269439
14. Liang, H., Yuan, J., Thalmann, D., Zhang, Z.: Model-based hand pose estimation via spatial-temporal hand parsing and 3D fingertip localization. The Visual Computer **29**(6-8), 837–848 (2013). DOI 10.1007/s00371-013-0822-4. URL http://link.springer.com/article/10.1007/s00371-013-0822-4
15. Liddel, S., R.E., J.: American sign language – compound formation processes, lexicalization, and phonological remnants. Natural Language and Linguistic Theory **4**, 445–513 (1986)
16. Microsoft Kinect for Windows. http://www.microsoft.com/en-us/kinectforwindows (2013)
17. Mihail, R.P., Jacobs, N., Goldsmith, J.: Real time gesture recognition with 2 Kinect sensors. In: International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV) (2012)
18. Myers, C.S., Rabiner, L.R.: Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition. The Bell System Technical Journal **60**(7) (1981)
19. Oikonomidis, I., Kyriazis, N., Argyros, A.: Efficient model-based 3D tracking of hand articulations using Kinect. In: British Machine Vision Conference, pp. 101.1–101.11. British Machine Vision Association (2011). DOI 10.5244/C.25.101. URL http://www.bmva.org/bmvc/2011/proceedings/paper101/index.html
20. Libfreenect. http://openkinect.org (2013)
21. Standard framework for 3D sensing. http://www.openni.org (2013)
22. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics **9**(1), 62–66 (1979)
23. Pedersoli, F.: XKin libraries. https://github.com/fpeder/XKin (2013)
24. Pedersoli, F., Adami, N., Benini, S., Leonardi, R.: XKin - eXtendable hand pose and gesture recognition library for Kinect. In: Proceedings of ACM Conference on Multimedia 2012 - Open Source Competition (2012). Nara, Japan
25. Peris, M., Fukui, K.: Both-hand gesture recognition based on komsm with volume subspaces for robot teleoperation. In: IEEE-Cyber (2012)
26. PrimeSense: NiTE. http://www.primesense.com/nite (2013)
27. PrimeSense: Sensing and natural interaction. http://www.primesense.com (2013)
28. Pugeault, N., Bowden, R.: Spelling it out: Real-time asl fingerspelling recognition. In: IEEE International Conference on Computer Vision Workshops, ICCV 2011, pp. 1114–1119 (2011)
29. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2), 257–286 (1989). DOI 10.1109/5.18626. URL http://dx.doi.org/10.1109/5.18626
30. Ren, Z., Meng, J., Yuan, J., Zhang, Z.: Robust hand gesture recognition with Kinect sensor. In: Proceedings of the 19th ACM international conference on Multimedia, MM '11, pp. 759–760. ACM (2011). DOI 10.1145/2072298.2072443. URL http://doi.acm.org/10.1145/2072298.2072443
31. Ren, Z., Yuan, J., Meng, J., Zhang, Z.: Robust part-based hand gesture recognition using kinect sensor. IEEE Transactions on Multimedia **15**(5), 1110–1120 (2013). DOI 10.1109/TMM.2013.2246148
32. Robot Operating System. http://www.ros.org/wiki/ (2013)
33. Rubine, D.: Specifying gestures by example. SIGGRAPH Comput. Graph. **25**(4), 329–337 (1991). DOI 10.1145/127719.122753. URL http://doi.acm.org/10.1145/127719.122753
34. $1 Unistroke Recognizer. http://depts.washington.edu/aimgroup/proj/dollar/ (2013)

35. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11, pp. 1297–1304. IEEE Computer Society, Washington, DC, USA (2011). DOI 10.1109/CVPR.2011.5995316. URL http://dx.doi.org/10.1109/CVPR.2011.5995316
36. Uebersax, D., Gall, J., den Bergh, M.V., Gool, L.J.V.: Real-time sign language letter and word recognition from depth data. In: IEEE International Conference on Computer Vision Workshops, ICCV 2011, pp. 383–390 (2011)
37. Wachs, J.P., Kölsch, M., Stern, H., Edan, Y.: Vision-based hand-gesture applications. Commun. ACM **54**(2), 60–71 (2011). DOI 10.1145/1897816.1897838. URL http://doi.acm.org/10.1145/1897816.1897838
38. Wan, T., Wang, Y., Li, J.: Hand gesture recognition system using depth data. In: Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on, pp. 1063 –1066 (2012). DOI 10.1109/CECNet.2012.6201837
39. Wang, R., Paris, S., Popović, J.: 6d hands: markerless hand-tracking for computer aided design. In: Proceedings of the 24th annual ACM symposium on User interface software and technology, UIST '11, pp. 549–558. ACM, New York, NY, USA (2011). DOI 10.1145/2047196.2047269. URL http://doi.acm.org/10.1145/2047196.2047269
40. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: Proceedings of the 20th annual ACM symposium on User interface software and technology, UIST '07, pp. 159–168. ACM, New York, NY, USA (2007). DOI 10.1145/1294211.1294238. URL http://doi.acm.org/10.1145/1294211.1294238
41. Zhang, H.J., Kankanhalli, A., Smoliar, S.: Automatic partitioning of full-motion video. Multimedia Systems **1**(1), 10–28 (1993). DOI 10.1007/BF01210504. URL http://dx.doi.org/10.1007/BF01210504

Table 2 Confusion matrix for the ASL letter classification task using SVM radial ($\hat{C} = 10$, $\hat{\xi} = 0.001$) on the dataset [28] with 24 letters and five subjects.

	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
a	**.90**																	.02	.02					
b		**.96**																						
c			**.91**		.02										.02									
d				**.90**						.02														
e			.03		**.90**																			
f						**.94**										.03								
g							**.91**	.03																
h								**.95**																
i									**.97**															
k										**.89**										.04				
l											**.94**													
m												**.90**							.02					
n												.05	**.88**						.02					
o					.03									**.87**										
p															**.90**	.04								
q															.07	**.88**								
r																	**.91**			.03				
s	.03													.04				**.83**	.04					
t	.03											.03	.03					.05	**.84**					
u																	.06			**.86**	.07			
v																				.04	**.91**			
w																					.02	**.93**		
x						.02											.02						**.90**	
y																								**.97**

Table 3 Confusion matrix for the ASL letter classification task using a Random Forest Classifier on the dataset [28] with 24 letters and five subjects.

	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
a	**.81**											.04						.02	.03					
b		**.89**				.02																.02		
c			**.83**		.03										.04									
d				**.83**						.02							.02			.03				
e			.03	.02	**.77**				.05			.02				.02								
f		.03	.02			**.87**										.02								
g							**.87**	.06																
h							.02	**.92**																
i									**.91**															
k										**.73**							.04			.03	.07		.02	
l											**.93**													
m	.06											**.68**	.08	.03				.02	.04					
n	.04											.14	**.65**					.02	.04					
o					.04							.02		**.76**		.02		.03						
p										.021		.02	.03		**.76**	.05			.02			.02		
q					.02	.04									.08	**.74**								
r				.05		.04				.04							**.73**			.05	.02			
s	.05				.02				.02			.04		.03				**.73**	.04					
t	.07											.07	.14	.03				.05	**.57**					
u		.03															.10			**.74**	.08			
v		.03				.02											.02			.06	**.76**	.02		
w		.05																			.07	**.79**		
x				.03		.02																	**.79**	
y									.02															**.88**