An Accurate Algorithm for the Identification of Fingertips Using an RGB-D Camera

Marco Maisto, Massimo Panella, *Member, IEEE*, Luca Liparulo, and Andrea Proietti

***Abstract*—RGB-D cameras and depth sensors have made possible the development of an uncountable number of applications in the field of human-computer interactions. Such applications, varying from gaming to medical, have made possible because of the capability of such sensors of elaborating depth maps of the placed ambient. In this context, aiming to realize a sound basis for future applications relevant to the movement and to the pose of hands, we propose a new approach to recognize fingertips and to identify their position by means of the Microsoft Kinect technology. The experimental results exhibit a really good identification rate, an execution speed faster than the frame rate with no meaningful latencies, thus allowing the use of the proposed system in real time applications. Furthermore, the scored identification accuracy confirms the excellent capability of following also little movements of the hand and it encourages the real possibility of successive implementations in more complex gesture recognition systems.**

***Index Terms*—Fingertip tracking, gesture recognition, human–computer interaction, Kinect sensor, RGB-D camera.**

I. Introduction

ONE of the most important field of study in these years is human–computer interaction (HCI), with all the technologies and devices allowing end users to control smart devices with gestures or voice only. Microsoft Kinect has been a pioneer solution in this framework [1]; this sensor is a new game controller technology developed with the collaboration of PrimeSense, an Israeli company that works in 3-D sensing since many years [2]. Kinect is a motion sensor capable of tracking movements, identifying gestures, and recognizing voice; this sensor not only has changed the meaning of controller in computer gaming, but it is also transforming the way we interact with every technological device. A brief survey on the Kinect technology will be reported in Section II. The capability of tracking people or generate depth map is connected to the projector-camera pairing: the former emits a pattern of speckles in the field of view and the latter compares and elaborates the reflected pattern with a reference pattern; their difference will give the depth information.

Many researchers have focused their efforts on designing Kinect applications in several fields of engineering and biomedical applications. Chang *et al.* studied the possibility of rehabilitating two young adults with motor impairments using a Kinect-based system in a public school setting [3]. Lange *et al.* developed and assessed an interactive game-based rehabilitation tool for balance training of adults with neurological injury [4]. Stone *et al.* applied Kinect to the problem of passive fall risk assessment in home environments [5]. Taylor *et al.* studied the use of Kinect and other natural gaming system for developing effective tools to aid rehabilitation in clinical settings [6]. Ross *et al.* assessed the concurrent validity of Kinect against a benchmark reference, a multiple-camera 3-D motion analysis system, in 20 healthy subjects during three postural control tests [7].

Several attempts were also taken to recognize, identify and track fine gestures and movements of the hand [8]–[10]. In spite of this, to the best of our knowledge, to build a robust hand gesture recognition system is still a challenging problem. Existing vision-based approaches are limited by light condition, background aspect, or by the exigency of wear gloves or markers. Introducing 3-D sensors such as Kinect, this task has become easier thanks to the accessibility of the depth map, although this is often not sufficient in order to obtain a really good system to recognize, for example, the gestures of fingers.

A method is proposed by using the so called *AcceleGlove*, an inexpensive micro controller and a set of five MEMS (Micro Electro Mechanical System) dual-axis accelerometers manufactured by Analog Devices. When applied to a character recognition problem, that is characters are designed using hands, good results are reported with a 100% of successes on 21 letters of the alphabet, with a minimal of 78% on the others ones [11]. Similarly, the task is reached using a combination of multiple data inputs: inertial measurement units (IMU) sensors, such as accelerometers and magnetometers, and the related data to normalize the hand angular pose; RGB and depth information to segment the hand and to perform a comparison of an observed gesture in a library of gestures, aiming to get the relevant blob of the hand by filtering the cloud of points previously produced [12].

A lot of attempts can be found in the literature that focus on the identification of the fingertips position with a standard RGB camera, although the known limitations mainly due to the light conditions [13]–[20]. The fingertip identification task has been also studied in many applications pertaining to the haptics field: in such a context, it is crucial to have the possibility of imitating the hand gestures by actually use a hand in front of a camera. For example, the grasping movement simulated with the hand would be very useful. Such a goal has been achieved in [21], where a complex and complete system for hand tracking and rendering in wearable haptics is presented.

Manuscript received January 31, 2013; revised March 19, 2013; accepted March 25, 2013. Date of publication April 24, 2013; date of current version June 07, 2013. This paper was recommended by Guest Editor V. Tam.

The authors are with the Department of Information Engineering, Electronics and Telecommunications, University of Rome "La Sapienza," 00184 Rome, Italy (e-mail: massimo.panella@uniroma1.it).

Digital Object Identifier 10.1109/JETCAS.2013.2256830

Fingertip tracking is also a research field for those applications related to multi-touch interfaces, where the use of the coordinates of fingertips could represent an improvement for the whole system and would allow the users to experience a more advanced and suitable form of interaction [22]–[24]. In applications related to augmented reality and mobile devices, tracking of finger movements and gestures is a challenging problem, since it would improves the user experience itself [25]–[27]. For instance, a gesture based interaction via finger tracking has been proposed in [28], allowing the user to interact with the virtual environment using his own hands and also improving the usability and the quality of the entertainment.

Fingertips can also be detected using depth maps returned from Kinect or, more in general, by depth sensors and RGB-D cameras. A typical application is obtained by first applying a circular filter on the hand to remove the fingers and get only the palm; then, the resulting image is subtracted from the original one to get the segmented finger masks and, supposing that fingertips are the closest objects to camera, the sought points are detected by finding the minimum depth in each finger [29]. The goal can also be reached using a three-point alignment relation, meaning that points that are both on the convex hull and on the hand contour are checked; for each possible candidate point, the distance between such a point and two points in opposite directions is checked; if the distance is larger than a specific value, than the three points are not aligned and the candidate point is labeled as fingertip [30].

A different way to achieve the task is a method based on the geometrical proprieties of the hand, that takes advantages of the shape and the innate geometry of each finger determined by the properties of points belonging to the convex envelope of the hand [31]. Another interesting method is the one proposed by Keskin *et al.* who have created a realistic 3-D hand model that represents the hand with 21 different parts [32]. Suited random decision forests (RDF) are then trained to perform a pixel classification and assign each pixel a hand part; then, using a recognition module based on a support vector machine (SVM), they reach a recognition rate of 99.9%, which is really impressive. Many others authors have also investigated the combination of wearable sensors and Kinect for the finger tracking problem [33], [34].

In addition, there are many software packages that people have placed online to achieve such goals. One of the most popular is a complete software package named "Candescent NUI," which uses Kinect to realize a nice fingertip identification application on desktop computers. Another interesting software package is the CogniMem software, which is built to improve Kinect's finger tracking by using the proprietary "CM1K" pattern recognition chip for Kinect. Other hardware platforms are also used in this context as, for example, the "Jakes Finger Tracking Suite," which uses the Nintendo Wii-mote to capture the spatial position of fingers and uses this information to control a computer.

In this paper, we aim to build a robust and efficient hand gesture recognition system by using a novel method in order to identify and track fine movements of the hand. The proposed approach is based on the analysis of some geometrical features and so, it has the capability to work efficiently in any light conditions (by using an infrared camera) and to overtake the drawback of wearing gloves or markers, since this is often an evident limitation in using such systems. Moreover, we will prove that the proposed algorithm is capable to achieve a very high identification rate even working in real time and without any latency problem, which is a fundamental feature in gesture recognition systems. Furthermore, it could be a starting point for developing a new paradigm for imaging systems considering the 3-D scene analysis and the related multiview vision algorithms and applications. By the way, a very hot topic actually concern enhanced multimedia and pervasive applications in connection with specific algorithms and hardware solutions [35]–[39].

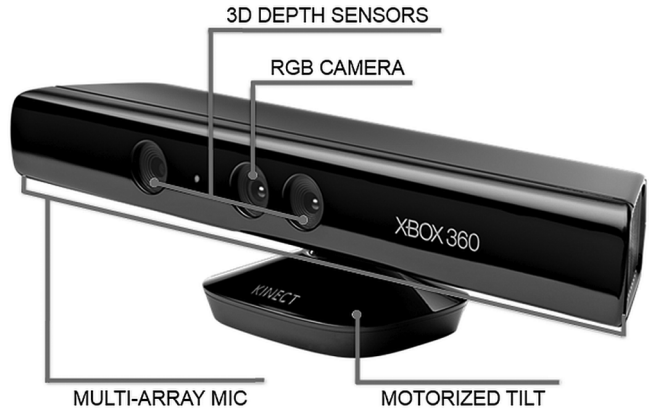

Fig. 1. Microsoft Kinect device.

The paper is organized as follows. A brief survey on the RGB-D technology implemented by the Kinect sensor is reported in Section II. The basic operations pertaining to the preprocessing of raw data sampled by Kinect are introduced in Section III, while the specific details about the different methods proposed in the paper for fingertip tracking and recognition are illustrated in Section IV. The results obtained from the experiments performed in our laboratory are summarized in Section V; they concern measures of speed, precision, accuracy, ability to recognize fine gestures, and a comparative analysis with other techniques in the literature as well. Finally, our considerations about the proposed work are drawn in Section VI and an Appendix is added with some code snippets that can be useful to understand some details of the proposed algorithms.

II. Kinect, What Is It and How It Works

The Kinect sensor shown in Fig. 1 is composed by an RGB camera, which is similar to an autofocus webcam, with an 8-bit VGA resolution of 640 × 480 pixels at 30 frames per second (fps) and using a Bayer color filter. The depth sensor uses an infrared laser projector combined with a monochrome CMOS sensor, allowing to capture data under any light conditions, with 11-bit depth resolution that provides 2048 levels of sensitivity. The combination of the RGB camera in the visible range with the infrared depth sensor realizes the so called "RGB-D sensor." In addition, four microphones are used in a directional array to implement vocal commands; the audio is processed with a 16-bit resolution at 16 KHz sampling rate. Finally, a tilt motor

Fig. 2. An example of speckle pattern generated by Kinect.

is also present, in such a way that Kinect can change its position on both vertical and horizontal axes.

Before the introduction of the Kinect technology, the depth information from an image was achieved by using the well-known time of flying (ToF) measure, that is the time the light impulse takes to cover the camera-to-object distance. Depth information is obtained by Kinect in a very different manner, using a really simple and cheap optical set up. The IR projector irradiates the ambient in the camera's field-of-view with a speckle pattern [40], [41]. The pattern is random in order to favor the reconstruction of the object and to reduce the complexity of the system [42]. In optical systems, a speckle pattern is an intensity field pattern generated by the local mutual interferences of partially coherent beams; the 3-D map of an object is then estimated by examining the laser pattern's shift. Hence, the reconstruction comes out by differences: the projector emits the pattern and the IR camera catches the reflected pattern. By using some parallel computer vision algorithms, which are commonly known as "Light Coding," Kinect succeeds in reconstructing the depth map [43].

As shown in Fig. 2, speckles are associated with a random pattern due to the interference of scattered light: white dots represent phase interference, black dots anti-phase ones; furthermore, the pattern is constant through the z axis. The optical system for 3-D object reconstruction is therefore based on the following elements [42].

- An illuminating unit, which comprehends a coherent light source and a random pattern generator (e.g., diffusion unit), situated in the optical path between the source and the illuminated object.
- Another unit necessary to individuate the light response of an illuminated region and to generate the image data relative to the shift of the pattern (associated with the observed object) with respect to a reference pattern.
- A control unit configured to store image data. Image data can also be processed and analyzed using a reference image, in order to execute a correlation between this and the object under investigation.

The speckles pattern is stored in the control unit and it is obtained in absence of any object. This reference pattern is taken only once a time; then, an image of the object to be reconstructed is taken and relative data are generated. To achieve this, the object is illuminated by the light source and the reflected light is detected by the light sensor through an optical lens. It is important to notice that this process only needs one image to reconstruct the 3-D map, thus significantly reducing the complexity of image processing tasks. Successively, the measured data are compared to the reference data to determinate the relative translation of the pattern features. Usually, this comparison is implemented by using appropriate matching algorithms as for instance correlation, which is quite simple and it could be based on fast Fourier transform (FFT) implementations on a dedicate DSP/FPGA circuit. Correlation peaks will show translations relative to the 3-D information, since the projected pattern is proportionally scaled with respect to the projector-to-object distance, correlation will be invariant to scale and projection. The control system will process the measured data applying a numerical algorithm for 3-D features extraction. This algorithm has a relatively low complexity and allows real time processing with a reduced latency.

III. New Approach to Fingertip Recognition

We propose in this paper a new algorithm that can play a core role in those systems aiming to recognize fine hand movements, considering the specific problem of tracking fingertips. The algorithm was developed using the software development kits (SDKs) provided by PrimeSense, OpenNI and, in particular, the Natural Interaction Middleware (NiTE) [2], [44]. Such SDKs are able to provide both skeleton and hand tracking, in addition to several other features.

The algorithm is proposed with three different variants, whose differences are discussed in detail in Section IV. However, they are all based on a common sequence of basic operations necessary to achieve the fingertip tracking. These operations are illustrated in the following step-by-step description, by means of a real example carried out in our laboratory by using the experimental system described in Section V.

A. Sampling and Segmentation of the Hand

The main object used for our purposes is the "hand-point" control returned by NiTE. It is an object fed with points that relate to one specific hand that is currently defined as the active hand. Unfortunately, pixels of the hand-point mostly lie on the hand contour but its shape is often defined roughly. So, we introduce in the algorithm a preliminary segmentation of the hand image detected by the Kinect sensor. Thanks to the segmentation, it is possible to impose different conditions on the pixels under consideration. First, we build a matrix with depth information; then, for every pixel of the image, its distance from the hand-point along the three axes is checked: if it is less than a threshold, the point will be identified as belonging to the hand. The result of this elaboration is illustrated in Fig. 3. We outline that, in this phase, the user which is making the gestures to be detected by the algorithm can be at any distance to the Kinect device.

B. Noise Canceling

As in every signal processing application, noise is a big problem that could dramatically reduce performances. We try

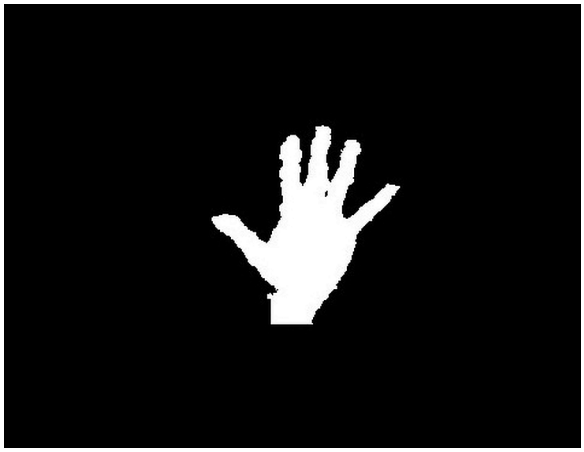

Fig. 3. Result of the hand segmentation.

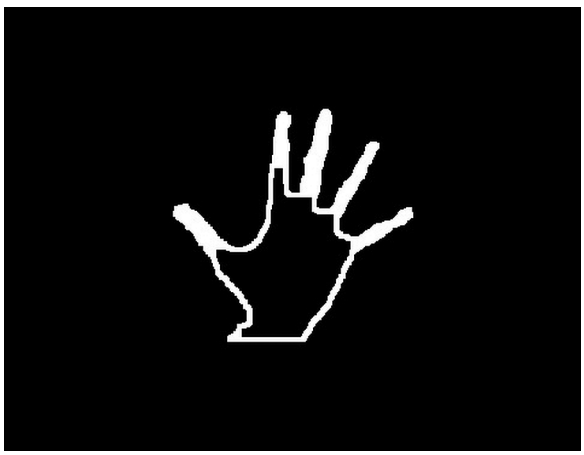

Fig. 4. Noise canceling after the application of the "morphology" operator.

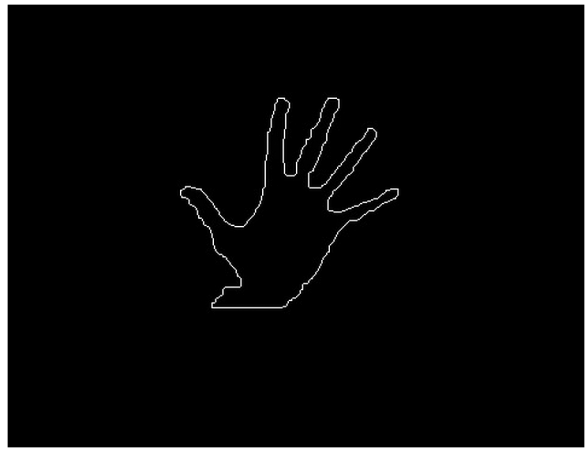

Fig. 5. Detected contour of the hand.

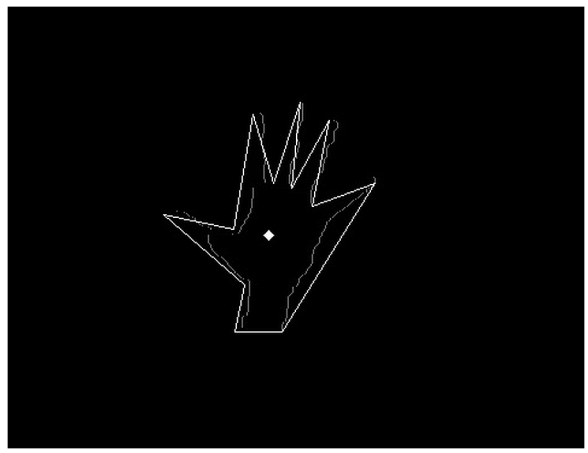

Fig. 6. CoM and the approximated (piecewise linear) contour.

to reduce noise using some functions from the OpenCV library [45], aiming to get a more suitable contour of the hand and have less points to analyze. *cvErode* will do an erosion of pixels in excess through the use of a very little image with a predefined kernel; it can be seen as the computation of local minima on the kernel area. *cvMorphologyEx* is an advanced morphological operator that will compute the geometric shape of the hand. Applying these functions to the set of pixels belonging to the segmented image of the hand, determined in the previous step, we obtain a well-shaped and more defined hand shape, resulting in fewer points to be analyzed. The result obtained at end of this step is shown in Fig. 4.

C. Hand Contour Detection

We use in this step the *cvFindContours* function from OpenCV to find the contour of the hand. This function will return a pointer to a sequence containing all the points regarding the hand contour. By the way, we have found out that the best results are obtained by retrieving only the external contour as well as the points that represent the end of horizontal, vertical and diagonal segments, which are compressed to reduce the number of points to be successively processed. Is it possible to see the result of this elaboration in the Fig. 5.

D. Contour Approximation

Bearing in mind the main goal of our application, it is convenient to work with a piecewise linear contour of the hand shape instead of a smoothed and curvilinear one, although the latter is much more realistic. This can be achieved by using the *cvApproxPoly* function from OpenCV, which approximates one or more curves using the well-known "Douglas-Peucker" algorithm that reduces the number of points representing the considered curve. So, the successive tasks can be performed on an angular contour, as the one obtained and illustrated in Fig. 6, reducing the complexity of the identification.

E. Computation of the Center of Mass

To find out the center of mass (CoM) of the hand the general theory of moments of an image is used [46]. The function *cvMoment* in OpenCV returns image moments up to the third order, allowing the straightforward determination of the CoM. As it will be explained successively, the CoM will be used either when points are to be considered as fingertips or not. The CoM related to the approximated contour obtained in the previous step is shown in Fig. 6. We would like to remark that further moments, particularly the second and the third one, can be computed on the hand contour obtained by Kinect up to this

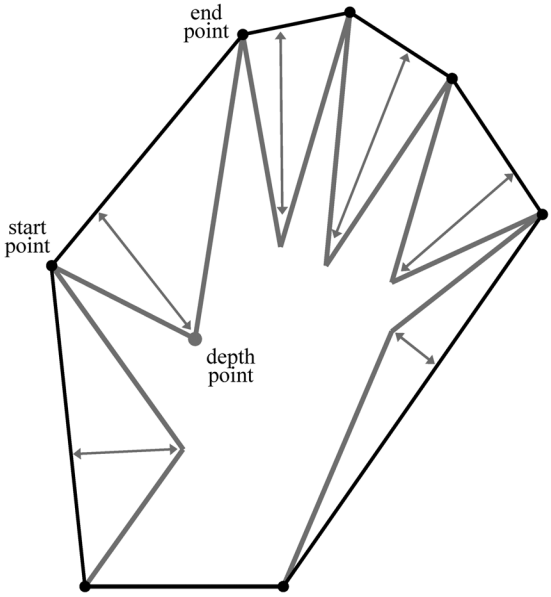

Fig. 7. Convex envelope and convexity defects of the hand: gray line is the approximated contour; black line is the convex envelope; black dots are the points of the contour determining the envelope; gray arrows represent the convexity defects. The highlighted depth point is associated with the convexity defect between thumb and forefinger.

stage. This can be useful in order to identify even the orientation of the hand, as it will be discussed successively.

F. Determination of the Convex Envelope

A convex envelope of a set of points is the smallest convex set that contains those points. The *cvConvexHull* function of OpenCV is used to retrieve the envelope of an object. In our case, we are looking for the envelope of the approximated hand's contour. This function uses the Sklansky's algorithm [47] to find the convex envelope of an n-dimensional polygon. The first step of this method finds the point having the lowest Y-coordinate; then, the remaining points are labeled in a clockwise order. For each point in the cue, it is checked if three points at time form either a "right turn" or a "left turn:" in the first case, the second-to-last point is not part of the convex hull and it has to be removed from the cue; otherwise, this point would be part of the convex envelope (i.e., of the hand in our case). This process goes on for every point in the stack and the resulting set of points will be the convex hull of the object under examination, as illustrated in Fig. 7. We outline that, by this elaboration, the convex envelope is determined by a subset of points of the hand's contour and so, surely there will be points in this set representing the fingertips.

G. Computation of the Convexity Defects

By using the OpenCV function *cvConvexityDefects* applied to the hand's contour and to the convex hull determined in the previous step, we can identify the convexity defects of the contour. In the case of a hand, six different defects should be obtained as illustrated in Fig. 7. Each defect is identified by three points of the contour: the point where the defect starts, the point where the defect ends, the *depth point* that is the farthest from the convex hull within the defect. These points are used successively to isolate and identify fingertips.

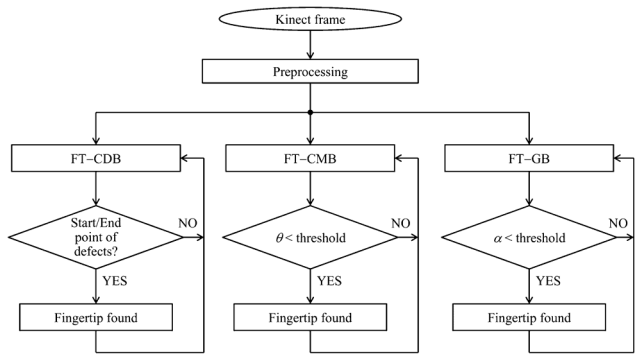

Fig. 8. Flow chart representing the main tasks carried out by the proposed FT algorithm. The preprocessing phase pertains to the operations illustrated in Section III. The thresholds with respect to parameters θ and α are discussed in Sections IV-B and IV-C, respectively.

It is important to notice that we are using an approximated piecewise linear contour of the hand. This also reduces the computational complexity of the proposed approach. In fact, for any vertex in the convex envelope, we may consider the straight lines of the contour that meet at this vertex. These lines start from the points of the hand's contour that are, respectively, prior to and next to the considered vertex (see Fig. 7). Consequently, those points can be considered as a good approximation of the depth points within the closest convexity defects and so, we need not to invoke the *cvConvexityDefects* function. Evidently, this assumption is valid *only if* the considered vertex of the envelope is actually a fingertip.

IV. Identification of Fingertips

The operations introduced so far represent a fundamental preprocessing of the raw data acquired by Kinect. They are the basis of the proposed algorithm for the accurate recognition and tracking of fine fingertip gestures. The general method, which will be referred to as fingertip tracking (FT) algorithm, is illustrated in the following considering three main variations that use different test conditions and subsequent decisions on the considered set of points. Precisely, the first method carries out a simple discrimination about the points that are on the contour of the hand and represent a part of the convexity defects structure, so it is a convexity-defects-based (CDB) variation of the FT algorithm. The second method is a center-of-mass-based (CMB) variation, taking into account the whole geometry of the hand shape. The latter alternative is a geometry-based variation (GB), which performs a simple morphometric analysis of each finger. A flow chart summarizing the whole process of the FT algorithm is shown in Fig. 8.

In the following description we illustrate and discuss the peculiarities of the proposed FT variations, some further details in this regard are also focused in Appendix A. We remark that the core application on these methods focuses on the correct recognition of X and Y coordinates of the representative points, that is pixels on the screen associated with each fingertip. As successively discussed, tracking is easily implemented by repeating the algorithm for every frame acquired by the Kinect sensor,

providing that the algorithm can be executed in a time smaller than the frame period.

A. FT-CDB Method

This method exploits the results returned by the *cvConvexityDefects* function; its application to the contour of a hand usually returns a set of eleven different points among start points, end points and depth points within the whole set of the detected convexity defects: four points regard the fingers joint (the so-called "finger slots"), five regard the fingertips, and two are totally wrong and standing on the wrist. This means that, in the structure containing all these points, we already have what we are looking for and they just need to be isolated and discriminated. As evidenced in Fig. 7, the sought points are the ones associated with the start or the end of convexity defects and hence, fingertips of the hand can be identified directly.

The main drawback of the proposed method is the possible wrong identification of one or two points in the bottom part of the hand. This is because the *cvConvexityDefects* function identifies a convexity defect between the wrist and the thumb (see Fig. 7); so, the algorithm wrongly identifies the start point of convexity at wrist and assigns to it the fingertip label. Obviously, the same situation occurs on the other side of the hand, in the region included between the wrist and the little finger. To avoid this, a threshold on the height of the considered points is created; let Y_{P_i}, $i = 1 \dots 11$, be the Y coordinate of the points associated with the convexity defects. In order to identify the fingertips, we consider only the subset of points satisfying the following condition:

$$Y_{P_i} < Y_C + \epsilon \tag{1}$$

where Y_C is the Y coordinate of the CoM C of the hand, which has been previously computed, with the Y-axis oriented downward. By the several tests carried out to calibrate the FT-CDB algorithm, we found out that using a threshold $\epsilon = 10$ pixels the performances of the algorithm are stable and consistent across a wide set of different operative conditions.

The use of this threshold solves the wrong identification problem, although it may result in an incorrect recognition of fingertips when the hand is pointing down. In this case, some points that actually represent fingertips may be discarded if their height is under the adopted threshold.

B. FT-CMB Method

This method focuses on the points of the convex hull: we outline that points representing fingertips belong to the convex envelope and, at the same time, to the contour of the hand. Let $\{H_1, H_2, \dots, H_N\}$ be the set of N vertices determining the convex hull. For each point $H_i, i = 1 \dots N$, which is a fingertip candidate point, we determine the convexity defects associated with it. As discussed in Section III-G, instead of using the depth points returned by the *cvConvexityDefects* function, we will use equivalently the points of the hand's contour that are prior to or next to H_i; let D_{i-1} and D_i be these points, respectively. Assuming $P = (X_P, Y_P)$ be the generic representation in the Cartesian coordinate system of a point P, the points M_{i-1} and M_i in the middle between H_i and D_{i-1} or D_i can be determined easily

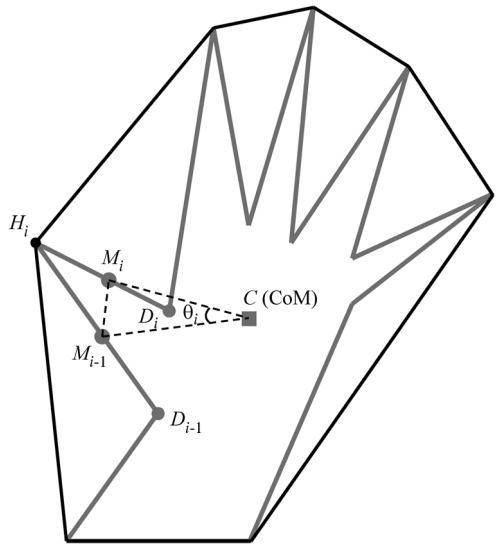

Fig. 9. FT-CMB method: middle points M_i and M_{i-1} associated with the point H_i of the convex envelope and the approximated depth points D_i and D_{i-1} within the related (closest) convexity defects. The angle θ_i associated with the CoM vertex C is checked in order to identify if H_i is a fingertip.

$$M_{i-1} = \left(\frac{X_{H_i} + X_{D_{i-1}}}{2}, \frac{Y_{H_i} + Y_{D_{i-1}}}{2} \right)$$
$$M_i = \left(\frac{X_{H_i} + X_{D_i}}{2}, \frac{Y_{H_i} + Y_{D_i}}{2} \right). \tag{2}$$

As illustrated in Fig. 9, for any point H_i of the convex envelope a triangle can be considered whose vertices are the CoM C and the pair of points M_{i-1} and M_i. The angle θ_i associated with the vertex C is calculated applying the well-known "cosine law"

$$\theta_i = \cos^{-1} \left(\frac{\overline{CM_{i-1}}^2 + \overline{CM_i}^2 - \overline{M_i M_{i-1}}^2}{2\overline{CM_{i-1}}\ \overline{CM_i}} \right). \tag{3}$$

Finally, every angle $\theta_i, i = 1 \dots N$, is compared with a suited threshold in order to determine if the point H_i is a fingertip. Almost stable performances of the algorithm, in any operative situation, are obtained when the condition $\theta_i < 50°$ is satisfied. Since there are points located at the wrist, the points at the top and bottom part of the hand (with respect to the CoM height) are counted in order to avoid a wrong identification when the hand is pointing down. When the top points are more than the bottom ones, i.e. the hand is pointing up, the algorithm works normally and the fingertips are correctly identified and associated with the considered points of the convex hull. Conversely, when the hand is pointing down, the algorithm considers a different threshold on each angle θ_i and fingertips can be identified as well.

Although this second variation of the FT algorithm should work better than the first one, it might still have a problem with the identification of fingertips when the hand is partially closed, for example, when it is showing the "victory" sign. In fact, it is normal that the hand's shape changes when it is partially closed, together with the position of the CoM. Consequently, the normal

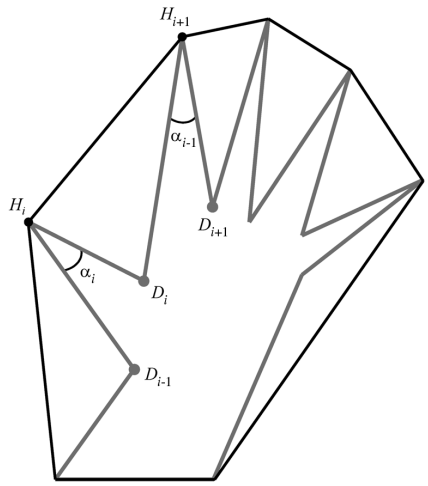

Fig. 10. FT-GB method: some checked angles for the triplets of the hand's contour associated with the vertices of the convex envelope. The angle α_i is checked to verify if H_i is a fingertip, the angle α_{i+1} is checked for H_{i+1}.

workflow of the algorithm is inhibited and it will yield a wrong identification.

C. FT-GB Method

This method uses directly the points of the approximated polygonal contour of the hand. In fact, such points are also the ones that contour the fingers and hence, it is possible to set a geometric restriction taking advantage of the regular shape of fingers. Once again, we consider the points $\{H_1, H_2, \ldots, H_N\}$ determining the convex hull; for each point H_i, $i = 1 \ldots N$, we have a triplet of consecutive points formed by H_i in the middle between the points D_{i-1} and D_i of the approximated hand's contour (the ones prior to or next to H_i, respectively).

As shown in Fig. 10, the angle α_i associated with the vertex H_i is computed for each triplet through the cosine law, similarly to (3). Since fingers are thin and long objects, by imposing a condition on α_i it is possible to identify whether H_i is a fingertip. In this case, a consistent condition for which a point can be considered as a fingertip is $\alpha_i < 45°$, $i = 1 \ldots N$.

Even if this method seems the simplest one, its results are surely the most meaningful since, without any other conditions or parametric assumptions, the FT-GB algorithm is able to find and correctly identify the fingertips irrespectively of the orientation of the hand. This is true because of the stable value of the angle formed by each triplet in every case: even when the hand is partially closed (or it is pointing down), the checked angle is going to maintain a same geometric property and proportion, leading the fingertips to be correctly identified.

Remark: It is worth noticing that a straightforward direction to the improvement of all the proposed algorithms is based on the use of the second order image moment, by which we should be able to identify the orientation of the hand. In fact, through the computation of the second order moment, which characterize the size and orientation of the image, it can be retrieved the angle φ formed between the principal component of the image and the X-axis [46].

Consequently, by elaborating the second order moment of the image under examination (i.e., the hand and the black background, in our case), we can retrieve information about the ellipse that is going to be formed around the hand centered at its centroid. Through some computations it can be retrieved the semi-minor and semi-major axes and the so called "tilt angle" φ, which is the angle formed between the X-axis of the image and the X'-axis of the ellipse related to the hand, as illustrated in Fig. 11. Checking the value of this angle, it is possible to evaluate the orientation of the hand (in terms of cardinal points N, E, S, W) and hence, it can be overtaken the problem related to the wrong identification of the fingertips when the hand is rotating or is not pointing up.

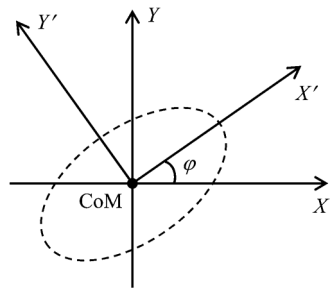

Fig. 11. Tilt angle that can be retrieved by using the second order moment of the image; the ellipse in the image is related to the hand.

The use of this kind of information would surely improve the performances of the proposed algorithms and it should be investigated in future research works. Consequently, given that the computational cost of the whole procedure might increase, there would be a stimulus to develop specific hardware architectures, using for example DSP and FPGA circuits embedded to the smart cameras, in order to maintain the capability of detecting and tracking fingertips in real-time applications.

V. Illustrative Tests

We have done several tests in order to prove the capability of the proposed FT algorithm to recognize and track fine movements of fingertips. The test phases have considered the identification rate, the velocity of execution of the related scripts, the precision and the accuracy of identification. The hardware platform used during the experiments was based on a Intel Core i7 950 @ 3.07 GHz processor, 6 GB of DDR3 RAM @ 935 MHz, NVIDIA GeForce GTX 570 video card with 1280 MB of dedicated memory, Corsair CSSD-F80GB2-A (SATA-SSD) with 78 GB; the software development has been carried out in C++ language on Microsoft Windows 7 Professional 64-bit operating system and Microsoft Visual Studio 2010 Ultimate Edition environment.

A. Identification Rate

Aiming to prove the real identification capability of the proposed methods, we have evaluated the identification rate in three different cases: opened hand pointing up (usual case); opened hand pointing down (unusual case); partially closed hand (unusual shape of the hand). The purpose of such tests is to provide a quantitative measure of the identification rate, evidencing possible limitations and also producing a comparisons among the three methods. The results are summarized in Table I as the percentage of successes in identifying correctly a fingertip for every method, considering 1000 trials that involve different human

TABLE I
IDENTIFICATION RATE (%) OF FINGERTIP GESTURES

Algorithm	FT-CDB	FT-CMB	FT-GB
Open hand pointing up	83	91	98
Open hand pointing down	0	22	68
Partially closed hand	68	4	88

subjects and ambient conditions. As expected, the FT-CDB algorithm fails in the case of an open hand pointing down and the FT-CMB algorithm fails when the hand is partially closed. Conversely, the FT-GB algorithm is able to obtain good results in any condition, thanks to the specific method that is used to identify fingertips.

In more detail, the first two methods achieve good results on the usual case of hand pointing up, around 83% and 91%, respectively. However, the performances fall down when the hand is pointing down and when it is partially closed. The reason of this crash has to be found in two obvious reasons: in FT-CDB we have a threshold on the heights of the points under examination; in FT-CMB the value of the center angle changes because of the modified hand shape. In fact, FT-CMB reaches a better result in identifying fingertips when the hand is pointing down rather than when it is partially closed.

The FT-GB algorithm reaches a really high percentage of successes when the hand is pointing up, nearly the very good results obtained in the similar tests illustrated in [30]. When the hand is pointing down or it is partially closed, the angle on the high extremity of every checked finger (the top of it) is almost the same, but we have a lot of false identifications (on the wrist), thus resulting in a lower identification rate of about 68%. When the hand is partially closed, the checked angle still remains the same and the threshold could be maintained as well, although now we have a little number of false alarms that bring down the identification rate to 88%.

As previously mentioned, some improvements could be obtained in the future by using the second order moment of the image and one should be able to overcome the previous drawbacks. With the information about the orientation of the hand, the limitation on the height of the point in FT-CDB can be surely removed and better and more precise results should be obtained for all the three methods as well.

B. Execution Speed

To evaluate the time necessary for the execution of the proposed FT algorithm and its variations, that is their latency, we have measured the time passed from the beginning of the script, when the frame is acquired by the Kinect sensor, until the end when a response is given about the gesture. The average values obtained over a launch of 1000 iterations of the algorithm are 1.5, 2.5, and 3.5 ms for FT-CDB, FT-CMB, and FT-GB, respectively. In all cases the results prove a satisfactory rapidity of execution for real time applications, since Kinect operates at a target rate of 30 fps and hence, the frame period is about 33 ms.

C. Precision of the Identification

Regarding the evaluation of the precision of the proposed approach, aiming to prove the stability and consistency of the identification, we have considered 1000 iterations of the FT algorithm for each one of the three variations and we have measured the values of both X and Y coordinates estimated by the scripts while a single finger tries to target a fixed point. In other words, we just point the finger to a known point on the screen and we try to maintain the finger fixed on that point for all the iterations, with the hand quite relaxed. The maximum deviation (measured in pixel) was 4, 4, 3 on the X-axis and 9, 10, 8 on the Y-axis in the case of FT-CDB, FT-CMB, and FT-GB methods, respectively.

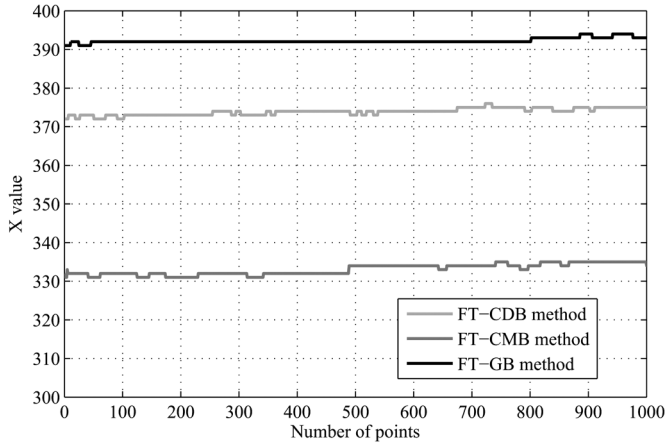

Fig. 12. Estimation of the X coordinate over 1000 iterations: each method is applied using a different reference point (X coordinate) to be estimated.

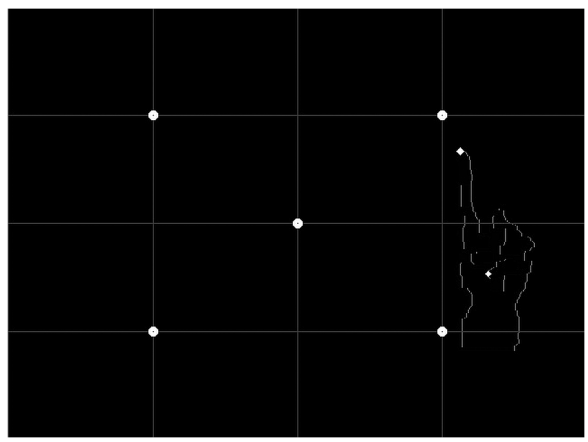

Fig. 13. Reference points used to test the accuracy of the algorithms.

An example of the results coming from these iterations is also shown in Fig. 12. For obtaining a meaningful plot with no overlapping curves, each algorithm is applied using a different reference point and hence, a different X coordinate to be estimated. This test confirms the optimal identification capability of the FT-GB algorithm, since it yields a more stable curve with respect to the other two. Moreover, the regularity of the lines depicted in Fig. 12 is evident with small values of deviation for all the three methods; this proves once more the really good precision of identification. We remember that the hand was quite relaxed during the test, meaning that probably these values can be enhanced if one tries to stand with the hand more fixed and

TABLE II
NUMERICAL RESULTS FOR THE PRECISION OF THE PROPOSED ALGORITHMS

Reference Point	Statistic	FT-CDB	FT-CMB	FT-GB
A (160, 120)	Mean	160.077, 121.890	160.037, 122.324	159.764, 121.932
	Min value	152, 116	156, 117	156, 118
	Max value	177, 134	181, 129	166, 136
	Standard dev	2.477, 3.463	2.253, 1.993	1.339, 2.360
B (160, 360)	Mean	160.085, 360.342	160.003, 359.379	161.260, 359.824
	Min value	156, 354	154, 356	155, 354
	Max value	179, 374	163, 370	165, 364
	Standard dev	3.311, 3.267	0.842, 2.135	2.115, 1.857
C (320, 240)	Mean	320.439, 238.316	321.599, 240.989	320.719, 239.407
	Min value	316, 226	314, 236	317, 235
	Max value	336, 243	340, 247	325, 249
	Standard dev	2.483, 2.859	1.868, 2.075	1.368, 2.254
D (480, 120)	Mean	479.505, 120.503	481.733, 119.980	479.160, 122.044
	Min value	428, 112	475, 112	476, 118
	Max value	484, 133	498, 132	486, 130
	Standard dev	5.237, 3.322	2.690, 3.342	1.929, 1.798
E (480, 360)	Mean	481.126, 358.5088	479.784, 361.525	480.352, 359.446
	Min value	470, 347	452, 345	477, 349
	Max value	489, 370	494, 371	486, 367
	Standard dev	2.439, 2.979	4.860, 3.355	1.704, 2.466

rigid in real scenario. Anyway, the scored precision seems adequate for a possible use of the proposed FT methods in a more complex system that uses the coordinates of fingers to implement some kind of gesture recognition and associate to them any type of command.

D. Accuracy of the Identification

The last test regards the accuracy obtained with the three algorithms. Five reference points are considered in the scene, as illustrated in Fig. 13, and the coordinates of the current point associated with the fingertip are sampled when the fingertip tries to hit each reference point. We considered 1000 iterations of this test for each algorithm and the resulting means and standard deviations were calculated. The numerical results are summarized in Table II where, for each of the three algorithms, it is possible to see the min-max range, the standard deviation and the mean, with respect to the actual value of the reference point. Each entry in Table II is a pair associated with X and Y coordinates of the corresponding point.

We decided to perform this test on five different points in the scene with the aim to determine if the angular distortion of the camera would affect, in some manner, the measure of the considered point; the results just deny this hypothesis. Moreover, these tests about the accuracy of the proposed methods show a really good trend and a really good performance of all the scripts, looking at the distance between real values and the measured ones for all the five reference points.

E. Comparative Analysis With Existing Techniques

The objective comparison of performances obtained by the proposed approach with respect to other techniques already presented in the technical literature is not straightforward. Different gestures, hardware platforms, ambient conditions and error measures are usually adopted in the experimental tests. Anyway, we have analyzed the state-of-the-art performances of the most popular algorithms and we report in the following discussion some comparisons having considered those papers where similar tests had been carried out.

In [23] a touch-screen is adopted for fingertip tracking and hand detection, which is a very different technology with respect to Kinect. However, a test is reported on identification rate that is similar to the one reported in Section V-A. In that paper, the true positive rate of fingertips correctly detected by the algorithm over about 6.600 visible fingertips was 97.3%, whereas the false positive rate of fingertips wrongly detected over about 6.500 detected fingertips was 1.35%. With respect to our tests, in particular the open hand pointing up that should be adopted for a touch-screen control, the FT-GB algorithm obtains a better result with a rate of 98%.

The algorithm proposed in [17] for fingertip tracking uses particle filtering and multi-scale edge extraction techniques. The results reported in that paper exhibit an accuracy (mean deviation) of about seven pixels for translation movements. For a rotation, opening and closing movement, the error increases up to 20 pixels. With respect to the statistical results illustrated in Table II, all the FT variants are able to obtain a higher accuracy of $1/2$ pixels on the mean error with a maximum standard deviation of about three pixels. Moreover, the framework proposed in [17] is capable of online tracking of fingertips at a frame rate of 15 fps on a 2.40 GHz dual core CPU platform, while the proposed FT algorithms sustain the Kinect frame rate of 30 fps using a 3.07 GHz i7 CPU.

The fingertip tracking system proposed in [48] adopts the Kinect technology for allowing the user to input characters by writing freely in the air. In that paper, the precision accuracy of the tracking algorithm is computed as the rate of frames for which the error is under a threshold of 5 or 10 pixels. Such analysis can be compared once again to the results shown in Table II. Taking into account that all the proposed FT variants obtain a mean drift of about 1 pixel and a standard deviation of about

```
while(contour){
   defects = cvConvexityDefects(contour, seqhull);
   for (i = 0, i < defects->total, i++){
   /* check the end point for the first and last point of the sequence */
      if ( (i == 0) || (i == (defects->total -1)) ){
         if (defectArray[i].end->y < center_of_mass + epsilon)
            {...}   /* a fingertip is found on the end point! */
      }   /* end if */
      /* check the start point for the other points in the sequence */
      if (defectArray[i].start->y < center_of_mass + epsilon)
         {...}   /* a fingertip is found on the start point! */
   }   /* end for */
}   /* end while */
```

Fig. 14. C++ code snippet of the identification process in the FT-CDB method.

```
cvConvexHull(PointArray, hull, hullsize);
/* initialize counters for points identified in the upper or lower part of the hand */
upper = 640; lower = 0;
while(contour){
   for (i = 0; i < hullsize; i++){
      idx = hull[i];
      /* create the counter for the points */
      if (PointArray[idx].y < upper){
         upper = PointArray[idx].y;
         up_count++;
         if (PointArray[idx].y > lower){
            lower = PointArray[idx].y;
            low_count++;
         }   /* end if */
         first_thresh = lower - ((lower - upper) * K);  /* K is set to 0.1 by rule-of-thumb */
         second_thresh = lower - ((lower - upper) * W); /* W is set to 0.7 by rule-of-thumb */
         /* create the couples of points: */
         /* idx is the index for the point being considered; */
         /* pdx and sdx are the index for the previous and next point, repsectively */
         for (i = 0; i < hullsize; i++){
            cv::Point u1 = ((PointArray[sdx].x + PointArray[idx].x)/2,
                            (PointArray[sdx].y + PointArray[idx].y)/2);
            cv::Point u2 = ((PointArray[pdx].x + PointArray[idx].x)/2,
                            (PointArray[pdx].y + PointArray[idx].y)/2);
            /* create the vectors between center of mass and the points just found */
            angle = acos((a^2 + b^2 - c^2) / (2 * a * b));
            if (up_count > low_count){
               if ((angle < angle_thresh) && (PointArray[idx].y > first_thresh))
                  {...}   /* a fingertip is found! */
               else if ((angle < angle_thresh) && (PointArray[idx].y > second_thresh))
                  {...}   /* a fingertip is found! */
            }   /* end if */
         }   /* end for */
      }   /* end if */
   }   /* end for */
}   /* end while */
```

Fig. 15. C++ code snippet of the identification process in the FT-CMB method.

three pixels, they should be able to stay under the error of 10 pixels in about 100% of the considered tests. Conversely, the algorithm proposed in [48] achieves a rate close to 92% and the other well-known methods therein considered, that is "nearest point," "curvature fitting," and "template matching" [16], [49], [50], scored a rate of 71%, 81%, and 89%, respectively.

In [25], the markerless camera tracking for hand identification and fingertip tracking is based on the analysis of frames acquired by a simple RGB camera. Considering a comparable test condition with the camera resolution of 640 × 480 pixels at a frame rate of 30 fps, the accuracy reported in [25] to determine the fingertip position varies from 5 to 8 pixels and hence, all the FT variants favorably compare even with respect to this algorithm, in particular the FT-GB method.

VI. Conclusion

In this paper, we propose an algorithm with three different variations to recognize and correctly identify the fine gestures of hands, in particular fingertips, through the use of the Kinect motion sensor. The proposed methods do work in real time without any noticeable lag and they exhibit good performances in the tests that carried out in this regard, also considering similar approaches already proposed in the literature. Good precision,

```
cvConvexHull(PointArray, hull, hullsize);
while(contour){
   for (i = 0; i < hullsize; i++){
      idx = hull[i];
      /* v1 and v2 will be the vector to which we will apply the cosine theorem */
      cv::Point v1 = ((PointArray[sdx].x - PointArray[idx].x), (PointArray[sdx].y - PointArray[idx].y));
      cv::Point v2 = ((PointArray[pdx].x - PointArray[idx].x), (PointArray[pdx].y - PointArray[idx].y));
      angle = acos((v1.x * v2.x + v1.y * v2.y) / (norm(v1) * norm(v2)));
      if (angle < angle_thresh)
         {...}   /* a fingertip is found! */
   }  /* end for */
}  /* end while */
```

Fig. 16. C++ code snippet of the identification process in the FT-GB method.

high level of accuracy and an adequate identification rate, especially for the FT-GB algorithm, are encouraging to develop a more complex gesture recognition system based on the possible poses of fingertips and their fine movements.

This system can be improved by the help of a neural network, which should be capable of learn new poses and store them in a database, allowing users to create their own customizable gesture database. The proposed approach can also be improved by merging depth data with RGB information. Through the use of combined RGB-D data a better result can be reached because of the additional inputs that can be processed like the color of pixels. Another improvement might be done by actually using more systematically the information related to higher image moments than the first one. In this way, it should be possible to overtake the problems seen in the FT-CDB and FT-CMB algorithms and hence, to achieve a better performance on the three as well.

Appendix A
Pseudo Code of the Three Algorithms

For the sake of illustration, we report in the following some C++ code snippets related to each one of the algorithms proposed in the paper. More precisely, they regard the critical part of the code concerning the identification process, which is different in the algorithms. Evidently, all the parts in common will not be reported, in order to avoid the description of all the code that would result in an unusable and unclear exposition.

The frames related to FT-CDB, FT-CMB, and FT-GB algorithms are illustrated in Figs. 14, 15, and 16, respectively.

References

[1] "Kinect," Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/Kinect

[2] PrimeSense [Online]. Available: http://www.primesense.com

[3] Y.-J. Chang, S.-F. Chen, and J.-D. Huang, "A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," *Res. Develop. Disabil.*, vol. 32, no. 6, pp. 2566–2570, 2011.

[4] B. Lange, C.-Y. Chang, E. Suma, B. Newman, A. Rizzo, and M. Bolas, "Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor," in *Proc. IEEE Int. Conf. Eng. Med. Biol. Soc.*, 2011, pp. 1831–1834.

[5] E. Stone and M. Skubic, "Evaluation of an inexpensive depth camera for passive in-home fall risk assessment," in *Int. Conf. Pervasive Comput. Technol. Healthcare*, 2011, pp. 71–77.

[6] M. Taylor, D. McCormick, T. Shawis, R. Impson, and M. Griffin, "Activity-promoting gaming systems in exercise and rehabilitation," *J. Rehabil. Res. Develop.*, vol. 48, no. 10, pp. 1171–1186, 2011.

[7] A. Ross, Y.-H. Pua, K. Fortin, C. Ritchie, K. Webster, L. Denehy, and A. Bryant, "Validity of the Microsoft Kinect for assessment of postural control," *J. Rehabil. Res. Develop.*, vol. 36, no. 3, pp. 372–377, 2012.

[8] H. Guan, C.-S. Chua, and Y.-K. Ho, "3-D hand pose retrieval from a single 2D image," in *Proc. Int. Conf. Image Process.*, 2001, vol. 1, pp. 157–160.

[9] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla, "Filtering using a tree-based estimator," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2003, vol. 2, pp. 1063–1070.

[10] Z. Zafrulla, H. Brashear, H. Hamilton, and T. Starner, "A novel approach to American sign language (ASL) phrase verification using reversed signing," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2010, pp. 48–55.

[11] J. Hernandez-Rebollar, R. Lindeman, and N. Kyriakopoulos, "A multi-class pattern recognition system for practical finger spelling translation," in *Proc. IEEE Int. Conf. Multimodal Interfaces*, 2002, pp. 185–190.

[12] P. Trindade, J. Lobo, and J. Barreto, "Hand gesture recognition using color and depth images enhanced with hand angular pose data," in *Proc. IEEE Int. Conf. Multisensor Fusion Integrat. Intell. Syst.*, 2012, pp. 71–76.

[13] K. Oka, Y. Sato, and H. Koike, "Real-time fingertip tracking and gesture recognition," *IEEE Comput. Graph. Appl.*, vol. 22, no. 6, pp. 64–71, 2002.

[14] J. Alon, V. Athitsos, Y. Quan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1685–1699, Sep. 2009.

[15] C.-C. Lee, C.-Y. Shih, C.-C. Yu, W.-R. Lai, and B.-S. Jeng, "Vision-based fingertip-writing character recognition," *J. Signal Process. Syst.*, vol. 64, pp. 291–303, 2011.

[16] D. Lee and S. Lee, "Vision-based finger action recognition by angle detection and contour analysis," *ETRI J.*, vol. 33, no. 3, pp. 415–422, 2011.

[17] M. Do, T. Asfour, and R. Dillmann, "Particle filter-based fingertip tracking with circular hough transform features," in *IAPR Mach. Vis. Appl.*, 2011.

[18] Y. Liao, Y. Zou, H. Zou, and Z. Liang, "Fingertips detection algorithm based on skin colour filtering and distance transformation," in *Int. Conf. Quality Software*, 2012, pp. 276–281.

[19] A. Chaudhary, J. Raheja, and S. Raheja, "A vision based geometrical method to find fingers positions in real time hand gesture recognition," *J. Software*, vol. 7, no. 4, pp. 861–869, 2012.

[20] H. Liang, J. Yuan, and D. Thalmann, "3-D fingertip and palm tracking in depth image sequences," in *Proc. 20th ACM Int. Conf. Multimedia*, 2012, pp. 785–788.

[21] V. Frati and D. Prattichizzo, "Using Kinect for hand tracking and rendering in wearable haptics," in *IEEE World Haptics Conf. WHC*, Jun. 2011, pp. 317–321.

[22] G. Hackenberg, R. McCall, and W. Broll, "Lightweight palm and finger tracking for real-time 3-D gesture control," in *IEEE Virtual Reality Conf.*, Mar. 2011, pp. 19–26.

[23] P. Ewerling, A. Kulik, and B. Froehlich, "Finger and hand detection for multi-touch interfaces based on maximally stable extremal regions," in *Proc. 2012 ACM Int. Conf. Interactive Tabletops Surf.*, 2012, pp. 173–182.

[24] S. Azenkot, J. Wobbrock, S. Prasain, and R. Ladner, "Input finger detection for nonvisual touch screen text entry in Perkinput," in *Proc. Graphics Interface*, 2012, pp. 121–129.
[25] T. Lee and T. Hollerer, "Handy AR: Markerless inspection of augmented reality objects using fingertip tracking," in *IEEE Int. Symp. Wearable Comput.*, Oct. 2007, pp. 83–90.
[26] J. Hannuksela, M. Barnard, P. Sangi, and J. Heikkila, "Adaptive motion-based gesture recognition interface for mobile phones," in *Proceedings of the 6th International Conference on Computer Vision Systems*. Berlin, Germany: Springer-Verlag, 2008, pp. 271–280.
[27] J.-H. An, J.-H. Min, and K. Hong, "Finger gesture estimation for mobile device user interface using a rear-facing camera," in *Future Information Technology*, ser. Communications in Computer and Information Science. Berlin, Germany: Springer, 2011, vol. 185, pp. 230–237.
[28] W. Hurst and C. V. Wezel, "Gesture-based interaction via finger tracking for mobile aumented reality," *Multimedia Tools Appl.*, vol. 62, pp. 233–258, 2013.
[29] J. Raheja, A. Chaudhary, and K. Singal, "Tracking of fingertips and centers of palm using Kinect," in *Proc. Int. Conf. Computat. Intell., Modell. Simulat.*, 2011, pp. 248–252.
[30] Y. Li, "Hand gesture recognition using Kinect," in *Proc. IEEE Int. Conf. Software Eng. Service Sci.*, 2012, pp. 196–199.
[31] Y. Wen, C. Hu, G. Yu, and C. Wang, "A robust method of detecting hand gestures using depth sensors," in *Proc. IEEE Int. Work. Haptic Audio Visual Environ. Games*, Oct. 2012, pp. 72–77.
[32] C. Keskin, F. Krac, Y. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Nov. 2011, pp. 1228–1234.
[33] A. Manaf and R. Sari, "Color recognition system with augmented reality concept and finger interaction: Case study for color blind aid system," in *Int. Conf. ICT Knowl. Eng.*., 2012, pp. 118–123.
[34] F. Cordella, F. D. Corato, L. Zollo, B. Siciliano, and P. V. D. Smagt, "Patient performance evaluation using Kinect and Monte Carlo-based finger tracking," in *IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatron.*, Jun. 2012, pp. 1967–1972.
[35] M. Panella and G. Martinelli, "An RNS architecture for quasi-chaotic oscillators," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 33, no. 1–2, pp. 199–220, 2003.
[36] R. Parisi, A. Cirillo, M. Panella, and A. Uncini, "Source localization in reverberant environments by consistent peak selection," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Honolulu, HI, Apr. 2007, vol. I, pp. 37–40.
[37] A. Chaudhary, J. Raheja, K. Das, and S. Raheja, "A survey on hand gesture recognition in context of soft computing," in *Advanced Computing*, ser. Communications in Computer and Information Science. Berlin, Germany: Springer, 2011, vol. 133, pp. 46–55.
[38] M. Panella and G. Martinelli, "Neural networks with quantum architecture and quantum learning," *Int. J. Circuit Theory Appl.*, vol. 39, no. 1, pp. 61–77, 2011.
[39] M. Panella and G. Martinelli, "The quantum approach leading from evolutionary to exhaustive optimization," *J. App. Sci.*, vol. 12, no. 19, pp. 1995–2005, 2012.
[40] Z. Zalevsky and J. Garcia, "Range mapping using speckle decorrelation," U.S. Patent 7433024 B2, Oct. 7, 2008.
[41] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," U.S. Patent 0118123 A1, May 13, 2010.
[42] Z. Zalevsky, A. Shpunt, A. Maizels, and J. Garcia, "Method and system for object reconstruction," U.S. Patent 0177164 A1, Jul. 15, 2010.
[43] "PrimeSense: Beyond natal-digital foundry meets the men behind the 3-D camera," Eurogamer [Online]. Available: http://www.eurogamer.net/articles/digitalfoundry-primesense-article
[44] OpenNI, the standard framework for 3-D sensing [Online]. Available: http://www.openni.org
[45] OpenCV-open source computer vision [Online]. Available: http://opencv.org
[46] M. Teague, "Image analysis via the general theory of moments," *J. Opt. Soc. Am.*, vol. 70, no. 8, pp. 920–930, 1980.
[47] J. Sklansky, "Measuring concavity on a rectangular mosaic," *IEEE Trans. Comput.*, vol. C-21, no. 12, pp. 1355–1364, Dec. 1972.
[48] Z. Feng, S. Xu, X. Zhang, L. Jin, Z. Ye, and W. Yang, "Real-time fingertip tracking and detection using Kinect depth sensor for a new writing-in-the air system," in *Proc. 4th International Conference on Internet Multimedia Computing and Service*. New York: ACM, 2012, pp. 70–74.
[49] L. Jin, D. Yang, L. Zhen, and J. Huang, "A novel vision based finger-writing character recognition system," *J. Circuits, Syst., Comput.*, vol. 16, no. 3, pp. 421–436, 2007.
[50] Z. Pan, Y. Li, M. Zhang, C. Sun, K. Guo, X. Tang, and S. Zhou, "A real-time multi-cue hand tracking algorithm based on computer vision," in *IEEE Virt. Real. Conf.*, 2010, pp. 219–222.

Marco Maisto was born in Aprilia (Rome), Italy, in 1986. He received the B.Sc. degree in telecommunications and the M.Sc. degree in communications engineering from University of Rome "La Sapienza," Rome, Italy, in 2009 and 2012, respectively.

In 2006 he joined the Universidad Complutense de Madrid during his participation to the Erasmus Project. During his academic studies and the subsequent research he has gained expertise in real time C++ programming for video stream processing and Microsoft Kinect applications.

Massimo Panella was born in Rome, Italy, in 1971. He received the five-year Dr.Eng. degree with honors in electronic engineering and the Ph.D. degree in information and communication engineering from the University of Rome "La Sapienza," Rome, Italy, in 1998 and 2002, respectively.

He is currently Assistant Professor of Circuit Theory, Neural Networks and Pervasive Systems at University of Rome "La Sapienza" and his research activity pertains to circuit theory, computational intelligence, and pervasive computing for modeling, optimization and control of complex systems.

Luca Liparulo was born in Campobasso, Italy, in 1983. He received the B.Sc. degree in telecommunications and the M.Sc. degree in communications engineering, in 2008 and 2011, respectively, from the University "La Sapienza, " Rome, Italy, where he is currently a Ph.D. degree student in information and communication engineering.

His research activity is related to the application of computational intelligence techniques in complex systems optimization.

Andrea Proietti was born in Rome, Italy, in 1986. He is currently a Ph.D. degree student in information and communication engineering at the University of Rome "La Sapienza," Rome, Italy. He received the B.Sc. degree in electronic engineering and the M.Sc. degree in electronic engineering for industry and innovation with honors from the University of "Roma Tre," Rome, Italy, in 2009 and 2011, respectively.

His skills and his research activity concern control electronics, measurements, data acquisition from sensors networks, multimedia and mobile applications.