Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping

Article *in* IEEE Transactions on Instrumentation and Measurement · November 2015

DOI: 10.1109/TIM.2015.2498560

2 authors, including:

Ana-Maria Cretu
Carleton University
86 PUBLICATIONS **374** CITATIONS

SEE PROFILE

Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping

Guillaume Plouffe and Ana-Maria Cretu, *Member, IEEE*

***Abstract*—The paper discusses the development of a natural gesture user interface that tracks and recognizes in real-time hand gestures based on depth data collected by a Kinect sensor. The interest space corresponding to the hands is first segmented based on the assumption that the hand of the user is the closest object in the scene to the camera. A novel algorithm is proposed to improve the scanning time in order to identify the first pixel on the hand contour within this space. Starting from this pixel, a directional search algorithm allows for the identification of the entire hand contour. The *K*-curvature algorithm is then employed to locate the fingertips over the contour, and dynamic time warping (DTW) is used to select gesture candidates and also to recognize gestures by comparing an observed gesture with a series of pre-recorded reference gestures. The comparison of results with state-of-the-art approaches shows that the proposed system outperforms most of the solutions for the static recognition of sign digits and is similar in terms of performance for the static and dynamic recognition of popular signs and for the sign language alphabet. The solution deals simultaneously with static and dynamic gestures as well as with multiple hands within the interest space. An average recognition rate of 92.4% is achieved over 55 static and dynamic gestures. Two possible applications of this work are discussed and evaluated: one for interpretation of sign digits and gestures for a friendlier human-machine interaction, the other one for the natural control of a software interface.**

***Index Terms*—hand gesture recognition; static gestures; dynamic gestures; RGB-D sensor; human-computer interaction;**

I. Introduction

RECENT years witnessed an increased research interest in the development of natural, intuitive user interfaces. Such interfaces have to remain invisible to the users, allowing them to unobtrusively interact with an application, without the need of specialized and costly equipment. They have to support a natural interaction and be adaptable to the user without imposing elaborate calibration procedures. At the same time, they have to fulfill their role in real-time, with accuracy, and provide robustness against background clutter. These various requirements and their inherent complexity still provide significant challenges for researchers.

As hand gestures constitute a powerful inter-human communication modality, they can be considered as well an intuitive and convenient mean for the communication between human and machines. This justifies the interest of the research community in the development and advancement of hand gesture technologies. One of the most important abilities of an efficient natural user interface is therefore its ability to recognize in real-time hand gestures.

The principal components of a hand recognition system are: data acquisition, hand localization (e.g. segmentation and tracking), hand feature identification and gesture recognition based on identified features. The classic solution for data acquisition is color cameras that have already been employed successfully for gesture recognition tasks [1, 2, 3]. These solutions are however sensitive to clutter, lighting conditions and skin color. Video capture has the extra challenge related to the speed of movement. In terms of 3D motion capture at the level of the fingers, possible solutions include optical marker systems, accelerometers, magnetic trackers and data gloves. These require extensive calibration, limit the natural movement of the fingers and are generally very expensive. The vision-based markerless motion capture became possible largely due to the introduction on the market of the Kinect, a low-cost, off-the-shelf sensor that gathers positional information about an individual's motion. Data captured by this sensor, in RGB-D (color and depth information) form, is often used as a source for hand gesture detection and recognition. While a good survey for the gesture recognition in general is available in [4] and one focused on depth images in [5], the following literature review summarizes briefly the solutions that make use of RGB-D data for the purpose of hand gesture recognition.

Different approaches can be used for hand localization in acquired data, according to its nature. For depth data, the classical solution is depth thresholding, either empirical or automated. Empirical solutions choose by trial and error the limits of the most probable search space and concentrate the computation effort for the hand localization within it. Among the automated solutions are: the detection of the closest point to the camera [6-10] based on the assumption that the hand is the closest object in the scene, and the use of other reference elements in the scene, such as the head position [11-13] or the facial color information [3] to identify the most possible location of the hand. An exhaustive solution is to scan the entire visibility space of the Kinect using overlapping depth intervals until the hands are found [14]. This can be however a very lengthy process. In terms of color images, possible

Manuscript received April 20, 2015. This work supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

The authors are with the Department of Computer Science and Engineering, Université du Québec en Outaouais, Gatineau, QC J8X 3X7, Canada (e-mail: ana-maria.cretu@uqo.ca).

solutions for hand segmentation include skin color detection [1, 3, 12, 15-17] and Haar-like features [2]. Finally certain approaches in the literature benefit from both depth and color information [7, 15-22]. Some systems impose the intervention of the user to calibrate the system prior to its use; for example in [20], the location of the hand is recuperated from a wave gesture executed by the user. Others impose constraints, such as the fact that the user wears a colored glove [22] or a black belt on the wrist of the gesturing hand [7, 18, 23] in order to simplify the hand localization procedure and its segmentation. These solutions impede on the invisibility of the interface to the user. Finally, certain authors [14] favor the middle of the screen as an area of higher likelihood for hand detection, therefore avoiding the search along the image sides in order to improve the response time. Such a system constrains the user to adapt its behavior to the interface.

After the hand localization, the next step is the selection of appropriate features that allow for the recognition of gestures. Among the solutions encountered in the literature for depth data are: hand contour, hand palm center [6, 14], finger properties (e.g. count, direction) [24], hand trajectory [11, 25], and gradient kernel descriptors [21], while for color data, popular choices are Haarlet features [20], Gabor features [26, 27], SIFT [21], SURF [15], Hu moment invariants [17], HOG [28, 29] descriptors, and shape distance metrics, such as the finger-earth mover's distance [7, 18, 19]. Certain authors [21, 28, 30] combine features using a bag-of-visual words approach prior to recognition. For the recognition of gestures from the selected features, possible techniques proposed in the literature are neural networks [11, 20], SVMs [15, 28, 30], nearest neighbor classifiers [8, 10], random forests [26], fuzzy approaches [1], and hidden Markov models [15, 27, 29]. Alternatively, a simple method that takes into consideration the position of middle and end point of a gesture trajectory with respect to the initial points is proposed in [17]. Most of the existing solutions for gesture recognition are designed for use either on static [8, 10, 20, 21, 24, 26, 31, 32] or on dynamic gestures [11, 28, 33, 34]. Overall there are only very few solutions that work simultaneously on both static and dynamic gestures [12, 25, 27]. Moreover, most of the current solutions are able to handle a single hand [1, 7, 15, 18, 21, 26, 29, 31, 32] or two hands [14, 17, 24]. To the best of our knowledge there are no solutions that can track and recognize gestures from multiple hands, as the one proposed in this paper. The results, in most cases in form of average recognition rates, are presented on datasets of selected static and/or dynamic gestures varying from 6 [32], 10 [8], 12 [10], 14 [17, 24], 24 (or 26) (gestures of the American sign language (ASL) alphabet; two gestures in this alphabet are dynamic and some authors work on static gestures only) [9, 21] and up to 42 gestures [27]. Most results are reported for a single hand [7, 9, 27], a few for two hands [17, 24].

The objective of this work is to develop an improved, low complexity and real-time solution for the recognition of both static and dynamic gestures executed by one or multiple hands from depth data returned by a Kinect sensor. The gain in performance and speed comes from multiple improvements in the various stages of the solution, including a faster algorithm to search the first pixel of the hand contour; the recognition of gestures independent of their position in the field of view of the Kinect; the use and adaptation of DTW not only as a validation tool [14], but also as a selector for candidate gestures; the normalization of disparity values to improve the recognition rate in the case of multiple hand detection; the use of constraints on the contour to ensure it is closed; and the use of contour information for recognition only in cases when no fingertips are identified, in order to improve the computation time. Moreover, the proposed approach offers an interface that allows the user to train the system with his/her desired gestures and test rapidly its performance. According to a recent review on gesture recognition [5], this is one missing aspect of current solutions: the capacity to learn a gesture set from the user. The interface is invisible to user in the sense that it does not require any constraints or calibration procedure. Due to the fact that our system is based only on depth information, cluttered backgrounds, lighting conditions, clothing or skin color have little impact on the reported performance.

Gesture recognition has multiple applications in the context sign language interpretation [21, 26, 27], smart home interactive control [33], control of a software interfaces [18, 19, 35, 37] and of virtual environments [15], human-robot interaction [20, 36] and robot hand control [22]. Among the possible applications of this work, two are evaluated in the context of this paper: one for the real-time interpretation of sign language digits and gestures, the other one in the context of natural control of a software application.

II. Proposed Solution for Hand Detection and Gesture Recognition in Depth Data

The proposed framework for hand detection and gesture recognition is illustrated in Fig. 1. Raw data coming from the Kinect is used to recuperate depth information on all the pixels of an image. A novel, faster algorithm is then proposed to identify each point of a closed contour identified within a given depth interval.

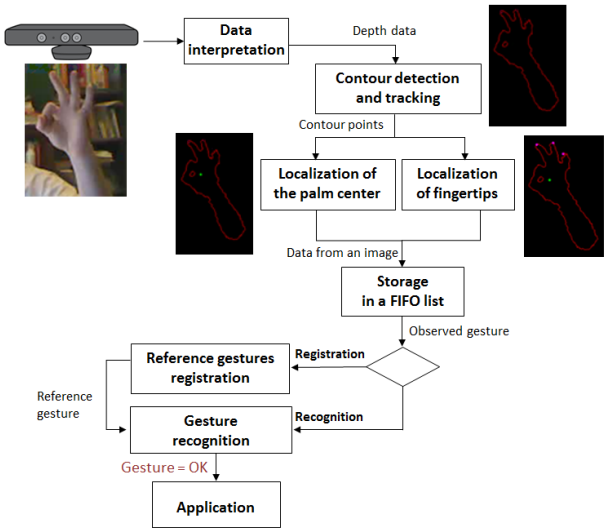


Fig. 1. Proposed framework for hand detection and gesture recognition

Starting from the recuperated contour points, the center of the palm is identified as the center of the largest circle circumscribed in the contour. The fingertips are localized by employing the *k*-curvature algorithm, which is based on the change in the slope angle of the tangent line at selected points over the contour. During the execution, the hand information (contour, palm center and fingertips) is stored in a fixed-size FIFO list. This list contains data on reference gestures that can be recorded by a user, according to the specific needs of the application. The same list is also used by the dynamic time warping algorithm in order to recognize previously recorded reference gestures.

A. Contour Detection and Tracking

From the 16-bit raw data collected by the Windows SDK 1.8 for Kinect, the last 13 bits represent the depth information. A 3-bit shift operation is performed to only retain this information. Each time a new frame becomes available from the Kinect, the depth information is recuperated and used to detect one or multiple hand contours.

Detection of the Interest Space - In order to guide the search for the hand contour, the proposed solution starts by identifying the minimal distance (closest) pixel to the Kinect, as illustrated in Fig. 2, based on the assumption that the hand is the closest object to the camera in the scene. The depth of this pixel is considered to represent the minimal depth for inspection and a constant (its value corresponding to the interval of inspection. i.e. an empirically determined value of 20cm in this work) is added in order to calculate the maximum depth within which a search is performed for the hands. The search for the hand(s) occurs within this limited space.

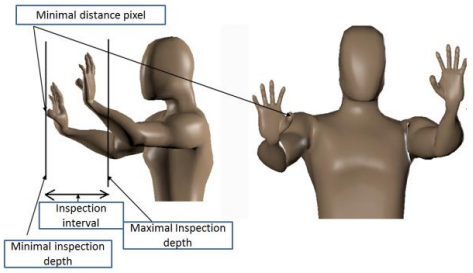


Fig. 2. Inspection interval with respect to minimum distance pixel

This approach is simple and does not impose significant limitations for the user, as we naturally tend to keep the hands in front of body when we gesticulate. In order not to have to be constrained by the closest pixel, a possible solution is to use other elements in the Kinect image as a reference, such as using of the skeleton tracking feature of Kinect to identify the hand position. However, due to the fact that this joint is among the least stable joints to track using the Kinect SDK [38], this solution was not retained in this work.

Search of the First Pixel of a Contour- A search is performed within the inspection interval to identify the first pixel that has at least one neighboring pixel that is not included in the inspection interval. To optimize the search, [14] proposes to inspect every other 5 pixels (e.g. blocks of 5×5) and does not take into consideration the pixels around the edges of the image (20% of pixel lines). In this work, the search algorithm is improved as it follows: A pixel is considered valid if it is present in the inspection interval. The search takes place in blocks of 20×20 pixels. If a pixel is valid and if it doesn't possess any neighboring pixel that is valid, the search continues pixel by pixel towards an invalid neighboring pixel until a contour pixel is found, as illustrated in Fig. 3.

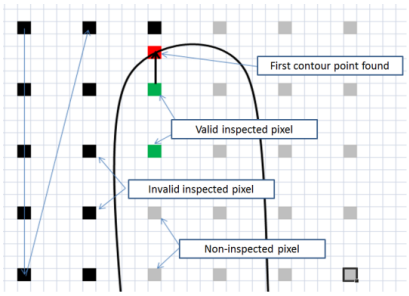


Fig. 3. Proposed algorithm for the detection of the first point of the contour inspecting each block of 20×20 pixels

If a pixel is valid and none of the neighboring pixels are invalid, the search continues with the next block. This algorithm results in a faster search. Because the Kinect is used with the resolution of 640x480 pixels, each image contains 307200 pixels. In the worst case, the proposed algorithm covers maximum 768 pixels (307200/400) to confirm an image that does not contain a hand. An improvement of 99.75% (or 100-(768/307200)×100) is therefore achieved for an image of 640x480 with respect to a full search. The proposed solution improves the search time by 15.25% with respect to the solution proposed by [14] and, unlike the latter, it does not restrict the search to the central part of the image.

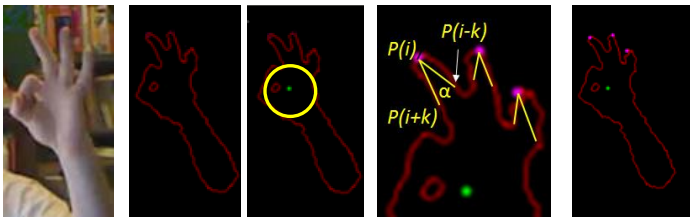
Contour Points Detection - Once the initial contour pixel is found, a directional search is performed to identify the full contour of the hand, as in [14]. In the context of this work, the considered directions are: up-left, up-right, down-left and down-right. A search direction is favored if it has been used for the previous pixel as well. The algorithm includes also a solution to backtrack if an unknown valid configuration is encountered. To further improve this approach, in this work, a constraint is added to only validate closed contours, as illustrated in the pseudo-code of the proposed algorithm shown below:

Algorithm 1 : Contour Point Detection

Input:
The first point, P_1, of a list of contour points belonging to a potential hand contour, **C**
Output:
List of contour points P_i belonging to an identified contour **C**

For each potential hand contour **C**
From the first point P_1 found, trace in order each point of contour and stop when the **C**.length > 700 or when the next point is already present in the contour list.
//if the last point is near the first point
if ((**C**.length > 10 and (|LastPoint.x – $\mathbf{P_1}$.x| <= 15 pixels and |LastPoint.y - $\mathbf{P_1}$.y| <= 15 pixels and |LastPoint.z- $\mathbf{P_1}$.z| <= 15 pixels)) then
the Contour **C** is accepted
Display the contour points in red
else
the Contour **C** is rejected
end

The detected contour using this procedure for the OK sign in Fig. 4a is shown in Fig. 4b.



(a) (b) (c) (d) (e)

Fig. 4. (a) Initial OK gesture, (b) detected contour, (c) largest circle contained in the palm, in yellow and identified palm center, in green, d) angle calculation for fingertip direction computation and (e) identified fingertips marked by magenta dots.

In order to identify more than one hand, once the closed contour points of the first hand are discovered, the algorithm continues the search for the first pixel of the next contour and repeats the process until it reaches the end of the image. The pseudo-code above is therefore repeated in a **for** loop for the detection of multiple hands.

B. Hand Palm Center Localization

Similar to the approach of [6], the hand palm center is calculated using the contour pixels as well as the interior pixels of the hand. To identify the bounding box of the hand, the minimum and maximum point on the *X* and *Y* axis are computed. The minimal distance (d_{min}) of interior pixels is then calculated for each block of 5×5 pixels with respect to each contour pixel. One contour pixel in 5 was proved experimentally to be sufficient for the final estimation of the hand center. The interior pixel that has the maximal value for d_{min} is considered to be the center of the palm.

In simple terms, the center of the palm is the center of the largest circle that can be circumscribed in the palm, as illustrated in Fig. 4c for the case of the OK sign. The proposed approach is summarized in pseudo-code as it follows:

Algorithm 2 : Palm center localization

Input:
C = List of contour point from the hand (obtained using Algorithm 1)
I = List of points inside the contour
Output:
Index of palm center coordinates in list **I**

1. For each element *i* of **I**:
   For each element *c* of **C**:
   Calculate the Euclidian distance d_i between *i* and *c*
   If $d_i > d_{i\,min}$ then
   $d_{i\,min} = d_i$
   If $d_{i\,min} < d_{i\,max}$, then
   // *i* cannot be the center of the palm;
   Return to step 1
   end
   Return the minimum Euclidean distance value obtained for *i*, $d_{i\,min}$
   end
2. Return the center of palm = element *i* of **I** with the maximum Euclidean distance value obtained, $d_{i\,max} = max\,(d_{i\,min})$.
   Display a green circle at the coordinates of the palm center.

C. Localization of Fingertips

The *k*-curvature algorithm is employed to identify the fingertips over the hand contour. The choice of this algorithm is justified by the fact that recent research showed that when compared with other solutions for static hand recognition, it provided the best results in terms of overall success rate, supported the highest range of hand rotations within which is capable to perform reliably and had the lowest complexity in terms of specific steps [31]. In simple terms, and as illustrated in Fig. 4d, the algorithm uses two equal length measuring tapes, V_A and V_B, of 20 pixels each (e.g. step size, *k*=20 pixels) and measures the angle θ between them (e.g. between *P(i-k)->P(i)* and *P(i)->P(i+k)*) over the contour of the hand. If the angle has a value between 25° and 55°, a fingertip is considered identified at that position. These angle limiting values are identified experimentally by trial and error. The pseudo-code of the algorithm is summarized in the Algorithm 3 below and the detected fingertips for the OK sign are shown in magenta in Fig. 4d and 4e.

Algorithm 3 : Fingertip localization

Input:
C = List of hand contour points
$\mathbf{P_1}$ = first point of the contour **C**
c_A = Start point of vector V_A
c_B = End point of vector V_B
Output:
Series of points P_f representing the detected fingertips

k = 20; $c_A = \mathbf{P_1} - k$; $c_B = \mathbf{P_1} + k$
For each element c of **C**:
if c < *k* then
c_A = **C**.length - *k* + c
else
c_A = c – *k*
end
if c > **C**.length then
c_B = k - **C**.length + c
else
c_B = c + *k*
end
create vector V_A(**C**[c_A], **C**[c])
create vector V_B(**C**[c], **C**[c_B])
calculate $\theta(c) = \arccos\left|\frac{V_A \cdot V_B}{V_A.\text{length} * V_B.\text{length}}\right|$ (in radians)
transform θ (c) in degrees
If θ (c) <= 55 and θ (c) >= 25
// fingertip detected
display a magenta dot of coordinates c representing the fingertip
end

While sometimes this algorithm can lead to false fingertip detection, particularly when the fingers of the user are thin and/or when they are very close to each other, the situation was not encountered frequently enough to be addressed. A possible solution to this problem would be to use an extra constraint on the distance between the center of the palm and the fingertips as well as between the fingertips. Additionally, a solution could be added for the prediction of a missing finger by computing the mean of movement in previous consecutive images, as in [14]. All these would nevertheless have a negative impact on the hand tracking time, while not necessarily bringing significant detection improvements.

D. Gesture Recognition Using Dynamic Time Warping

While static recognition can be accomplished by standard pattern recognition techniques (e.g. template matching),

dynamic gesture recognition requires the incorporation of a time component in the process. Among the most often employed techniques are: time-compressing templates, dynamic time warping, hidden Markov models and time-delayed neural-networks [4]. In the context of this work, the dynamic time warping algorithm is used to compare the similarity between a reference and a captured gesture, whether the gesture is static or dynamic. The proposed approach allows the user to record a sequence of reference gestures, as further explained in section III, that are saved in a FIFO list. In order to ensure real-time behavior of the system, this list is limited to 40 images. When the recording is finished, these 40 processed images are saved in an xml file. Once a sequence of new images representing a gesture being executed is made available by the Kinect, the recognition module is activated to recognize the corresponding gesture based on the features identified in section B and C.

The gesture recognition module executes two steps: (1) the search of candidate gestures based on the similarity between the observed gesture and each reference gesture and (2) the validation of the similarity between each observed version and each identified candidate reference gesture. For the first step, [14] compares the last image of an observed gesture with the last image of each reference gesture in the list of reference gestures by calculating the Euclidian distance. If the reference gesture associated with the lowest distance is within a threshold, the gesture is considered to be a possible candidate. While simple and fast, this solution fails to distinguish between two dynamic gestures whose last image is identical, because it only compares the latter in order to recognize a gesture. To cope with this problem, we propose the use of DTW algorithm [39, 40] not only for validation (in step (2)), but also as a mean to identify possible candidates (in step (1)). The impact of this choice is further illustrated in section III.

The DTW algorithm calculates the disparity (inverse of similarity) between two data series acquired at different times. A matrix containing the Euclidian distances at aligned points over the two sequences is used to calculate the minimal cost (shortest path) between the two series. The choice of direction for the shortest path is associated with certain rules. In particular, the movement is restricted to horizontal, vertical and diagonal directions. A weight is associated with each of these directions and the shortest path has to be inferior to a threshold in order for the two sequences to be considered similar. In the context of this work, the two series represent the sequences of hand positions in an observed and a reference gesture, respectively. To allow for the recognition of gestures independent of their position on the screen, in step (1), we use the position of fingertips with respect to the center of the palm(s) and the relative position of the palm(s) center with respect to the palm center of the first discovered contour, as illustrated in Fig. 5a. The hand disparity value is calculated as a sum of Euclidean distances between each fingertip and the palm center. For a limited number of gestures in which the fingertips are not visible, the contour points (each 10^{th} point) are used as a basis to calculate the distance. Since this distance gets higher for gestures involving multiple hands and fingers, the resulting value is normalized by the number of hands and fingers detected in an image in order not to bias the recognition. The disparity value for an entire image is therefore obtained by dividing the total hand disparity over all detected hands by the number of hands. If this value is lower than a threshold (e.g. the maximum Euclidian distance accepted for a candidate gesture in Table I), the gestures are considered similar (Fig. 5c) or else they are rejected as possible candidates (Fig. 5b).

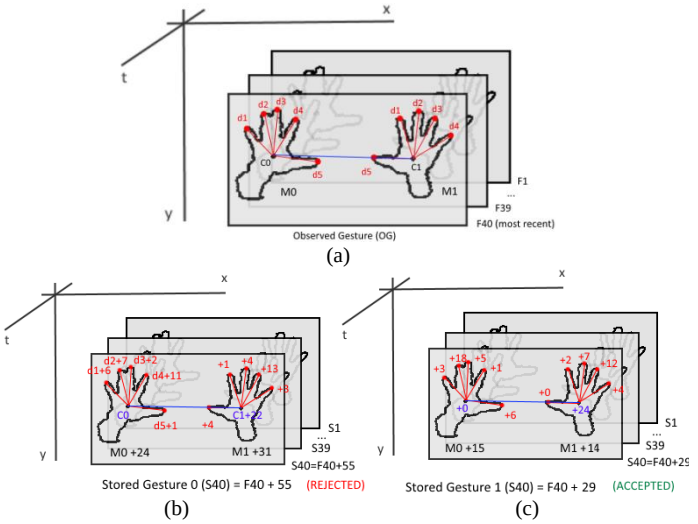


Fig. 5. Gesture disparity calculation between the (a) the observed and (b, c) stored gestures for the selection of gesture candidates

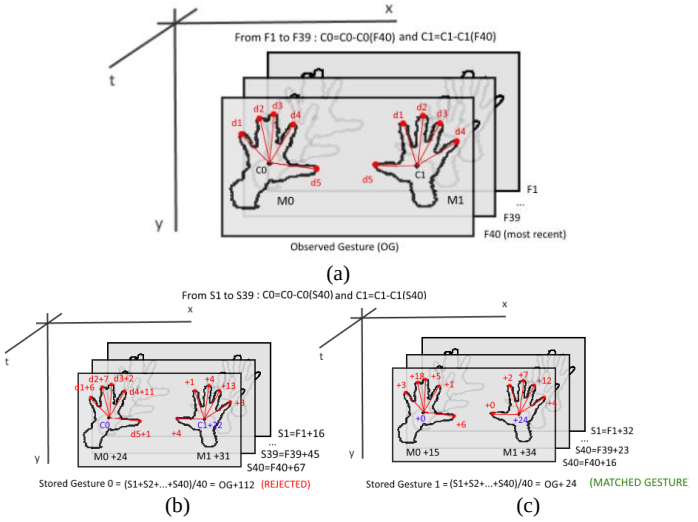


Fig. 6. Gesture disparity calculation between the (a) the observed and (b, c) stored gestures for the validation of the recognized gesture

In step (2), we use the relative position of the palm(s) center with respect to the corresponding palm center of the last recorded image, as illustrated in Fig. 6, to validate the similarity between each observed version and each identified candidate reference gesture. The gesture disparity is calculated as the sum of each image disparity divided by the number of images. Table I summarizes the empirical choice of parameters used during experimentation for the DTW. The value for the maximum Euclidian distance, in pixels, accepted for a candidate gesture is selected as 30. It represents a good compromise between the number of candidate gestures to be

processed and the processing time, as in this case the 40 last images in the list are compared to make a decision. The maximum cost accepted for a path in the DTW matrix per image is set empirically to 30.

TABLE I
PARAMETERS FOR THE DYNAMIC TIME WARPING ALGORITHM

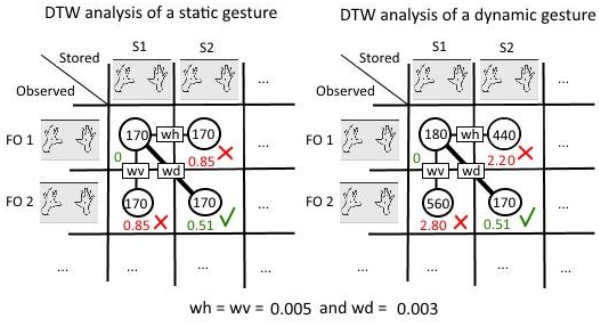
Parameter	Value
The maximum Euclidian distance, in pixels, accepted for a candidate gesture	30
The number of vertical or horizontal movements allowed in the shortest path (HorizontalMovementThreshold, VerticalMovementThreshold)	10
Maximum cost accepted for a path in the DTW matrix per image (PathCostThreshold)	30
Weight used for the difference between the relative palm position in the last image of the reference gesture and the one of the observed during candidate search	7
Weight used for the difference of palm position relative to the last image between each image of the candidate reference gesture and observed gesture at the construction of the DTW matrix	7
Weight used for the difference between the finger position with respect to the palm center in the last image of the reference gesture and the one of the observed during candidate search	1
Weight used for the difference of finger position with respect to the palm center between each image of the reference gesture and observed gesture at the construction of the DTW matrix	1
Weight used for direction choices (horizontal, vertical, diagonal) during the search of the shortest path	(0.005, 0.005, 0.003)

The pseudo-code summarizing the approach is shown below:

Algorithm 4 : Gesture recognition using DTW

```
Input:
    g_obs = Last observed gestures (40 images)
    g_cand = List of candidate gestures (40 images)
Output:
    g_recogn = Name of recognized gesture
      (or none : Observed gesture not recognized )
For each gesture in g_cand:
  // Get the 2D DTW matrix containing the disparity between each
  // image of both gestures.
  dtwMatrix = FindDtwMatrix (g_obs, gesture)
  // Use a Greedy algorithm to find the minimum path cost in disparities
  // in the DTW matrix; return -1 if the cost between two images is ∞
  // or the horizontalMovementCounter > HorizontalMovementThreshold
  // or the verticalMovementCounter > VerticalMovementThreshold
  pathCost = FindLowestCostPath(dtwMatrix)
  // Calculate the average Pathcost per image
  minPathcostPerFrame = pathCost / gesture.Frames.Count
  // Maximum cost accepted per frame = 30
  If (minPathcostPerFrame < 30 and pathCost != -1):
     return g_recogn  //name to be displayed on screen
  else
    return none;
  end
end
```

The weights associated with the directions of movement are experimentally set to 0.005 for vertical and horizontal direction and 0.003 to the diagonal direction, in order to favor the latter. Because we perform a search in the DTW matrix for the shortest path, in the virtual space of frames (observed vs. reference gestures), as illustrated in Fig. 7, in order to include static gestures, for which neighboring values are in general equal (Fig. 7a), the search is biased (diagonal weight *wd* is lower than vertical, *wv*, and horizontal, *wh*, weights) to favor the next image of both reference and last gesture performed, corresponding to a diagonal movement in the DTW matrix.



(a) (b)

Fig. 7. DTW matrix for static and dynamic gestures.

To further improve the performance, a maximum number of 10 horizontal and vertical movements are allowed. This results in a faster elimination of divergent candidate gestures. In order to select the winning gesture among the gesture candidates, the one with the smallest disparity value (and that respects the threshold) is chosen.

E. User Interface

The programmed interface (Fig. 8) proposes various functions and visualization options to the user. The *Color View* and *Depth View* buttons activate the RGB camera and the depth camera of the Kinect, respectively.

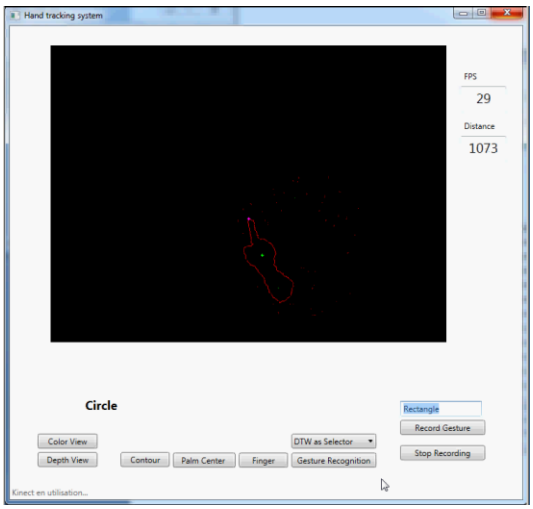


Fig. 8. User interface for the hand tracking and gesture recognition system

The *Contour*, *Palm* and *Finger* buttons allow for the tracking of each interest part of the hand. The *Gesture Recognition* button activates the gesture recognition module. If a gesture is recognized, its name appears in the message zone of the interface, as shown in Fig. 8, where the recognized dynamic gesture is a circle. The field *FPS* on the upper right side of the interface shows in real-time the number of

processed images per second. The field *Distance*, under *FPS*, shows the relative distance of the hand to the camera. A user can record a gesture by first associating a name with that gesture and introducing it in the box on the top of the *Record Gesture* button and then pressing on the latter. The recording starts and can be stopped by the user at any moment by pressing the *Stop Recording* button. If the *Stop Recording* button is not pressed within 5 seconds, the recording is stopped automatically. This functionality is implemented for gestures requiring the two hands of a user. The "Exit" reference gesture is reserved for a gesture that allows shutting down the application. This gesture can be recorded by the user and an example of its use is presented in the experimental section of the paper.

III. Experimental Results

Several experiments were performed in order to test the proposed system for hand tracking and the localization of the palm center and of the fingertips. The approach is then tested and evaluated on a series of 55 static and dynamic gestures for which results are reported in the literature to enable performance comparisons.

A. Hand Detection and Tracking

A few samples for the detection of the contour, palm and fingertips for one hand are shown in Fig. 9a, for two hands in Fig. 9b and for multiple hands in Fig. 9c. It can be observed that, in general, the proposed solution is able to correctly locate the points of interest over the hand surface as well as its contour. A few problematic cases were also encountered with the proposed solution. Some examples are illustrated in Fig. 8d. The palm center, fingers and contour are sporadically lost between two consecutive images and this behavior is related to the Kinect sensor. Fig. 10 shows the position of the palm center, in pixels, during the execution of a circle-like dynamic gesture. To deal with the problem of missing measurements along z axis (the blue points located at origin) and of other outliers, the proposed solution identifies and eliminates them from the data before proceeding with gesture recognition.

Sometimes double fingertips are identified for the same finger. This is mainly due to the use of the *k*-curvature algorithm, which, because of implementation specificities, tends to require larger distances between two fingers in order to make the distinction between them. Fingertips can be missing in particular for gestures involving heavily bending of fingers or when occlusions occur due to the superposition of hands. The latter characterizes all optical motion capture sensors but has less impact on the proposed application, since gestures do not involve occluding hands.

B. Static Gesture Recognition

In order to test the performance of the proposed solution, we have used a set of nine popular gestures, illustrated in Fig. 11, as proposed by [24], the set of sign digits from 1 to 9 shown in Fig. 12 as in [7, 18, 24] and the ASL alphabet [9, 21, 27].

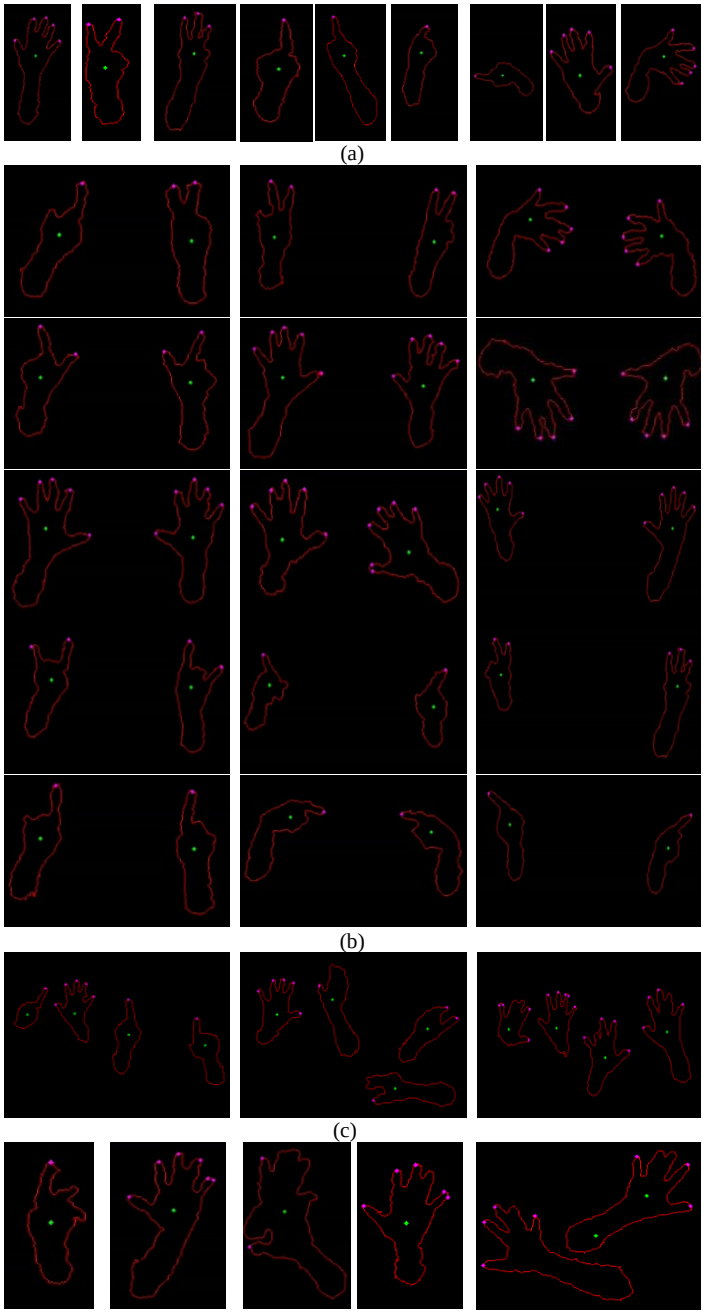


Fig. 9. Examples of contour, palm center and fingertip detection for (a) one hand gestures, (b) two hand gestures, (c) multiple hand gestures, and (d) a few problematic cases

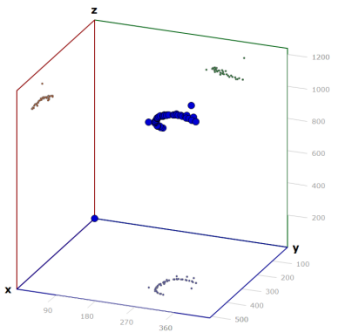


Fig. 10. Relative position of palm center for dynamic circle gesture (in blue) and its projections over x, y, z axis; z represents the depth.

Fig. 11 and Fig. 12 show, for the first two testing sets, the image of each gesture and the identified contour, palm center and fingertips when only the relative position of the fingers with respect to the palm is used for recognition.

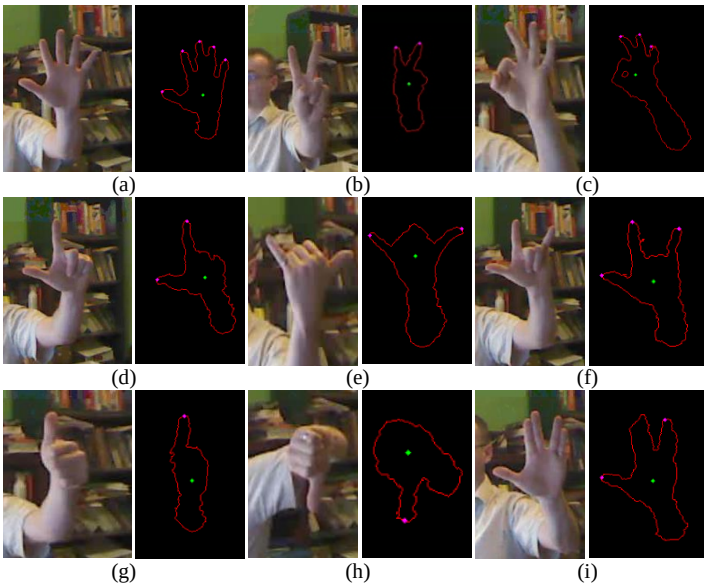


Fig. 11. Popular gestures: (a) complete hand, (b) victory, (c) OK, (d) letter L, (e) letter Y, (f) I love you, (g) thumb-up, (h) thumb-down, and (i) Star Trek.

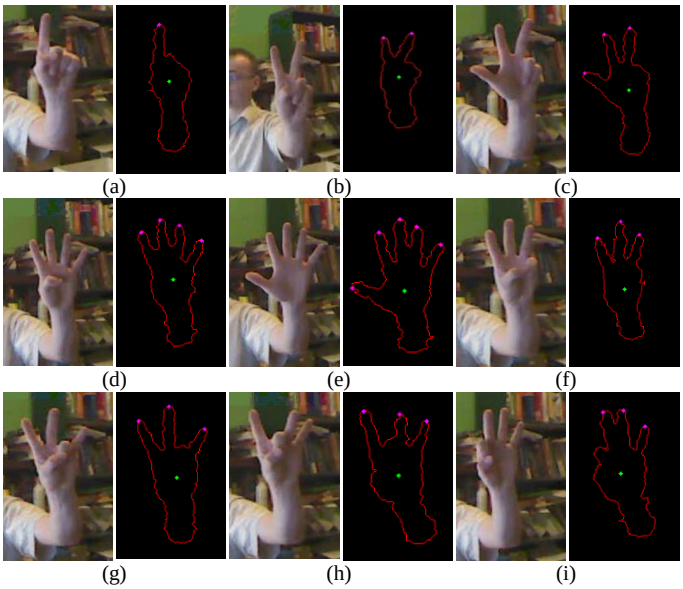


Fig.12. Sign number gestures: (a) one, (b) two, (c) three, (d) four, (e) five, (f) six, (g) seven, (h) eight, and (i) nine.

The tests do not take place in controlled conditions. As one can notice in the figures, the background is heavily cluttered. To perform the experimental testing, for each sign in each test set, reference gestures of the user are registered in double, one version roughly situated at 950mm, the other one at roughly 1050mm. The purpose of this double recording is to compensate for the scaling effect due to the use of different distances of the hand with respect to the camera. Reference gestures are recorded separately for each category of tests. During the second part of the test, the user chooses gestures from the database, in the desired order, and is not informed whether the gesture was recognized or not (double-blind testing). For each type of test, single or double blind, each set of tests is repeated 5 times and the corresponding video is recorded for further analysis. For each gesture executed by a user, the time required by the system to recognize the gesture is calculated. A gesture is considered unrecognizable if the system is unable to identify it within a predefined interval after the beginning of its execution. If the decision fluctuates between two reference gestures, the observed gesture is considered recognized when the system decision stabilizes for one of the candidates. If the system does not stabilize within the interval, it is considered that the corresponding gesture was not identified.

The results of this analysis are shown for the single and double blind tests in Table II and Table III respectively.

TABLE II
AVERAGE RECOGNITION RATES AND TIME ESTIMATES FOR SIGN NUMBERS

Gesture	Single blind		Double blind	
	Rate (%)	Time(s)	Rate (%)	Time(s)
One	100	1.63	100	1.48
Two	100	1.58	100	2.03
Three	80	5.30	90	3.36
Four	100	2.18	95	2.43
Five	80	4.93	80	4.88
Six	90	4.01	75	3.81
Seven	95	2.83	90	3.37
Eight	100	1.85	100	1.68
Nine	100	1.80	100	2.08
Average	**93.9**	**2.9**	**92.2**	**2.8**

TABLE III
AVERAGE RECOGNITION RATES AND TIME ESTIMATES FOR POPULAR GESTURES

Gesture	Single blind		Double blind	
	Rate (%)	Time(s)	Rate (%)	Time(s)
Open hand	80	4.93	70	1.26
Victory/ letter V	95	1.49	100	1.63
OK/ letter F	40	2.69	40	3.81
Letter L	100	1.65	100	2.03
Letter Y	90	1.53	90	3.16
I love you	100	2.80	100	3.45
Thumbs up	100	2.03	100	2.13
Thumbs down	100	2.83	90	4.21
Star Trek	60	4.3	40	1.81
Average	**85**	**2.7**	**81.1**	**2.6**

One can notice that the average performance during the double blinds testing is slightly lower than for the one obtained for the single blind tests. The slight decrease is justified by the lack of feedback to the user, which won't normally be the case of any natural gesture interface. The tests are nevertheless performed to show that, even in such conditions, without any user adaptation, the performance of the proposed solution does not degrade significantly (overall less than 4% single blind versus double blind variation). In terms of recognition time, there is no significant difference between the two types of tests (roughly 0.1s). It is important to state that the time values reported in Table II and III also include the visual presentation of the tracking of the contour, palm and fingertips and not only the computation time required by the recognition module. Tables IV and V compare the single blind tests with related solutions in the literature.

TABLE IV
AVERAGE RECOGNITION RATES (%) FOR SIGN NUMBERS

Gesture	State-of-the-art solutions				Proposed solution***
	[24]	[7]*	[7]**	[18]	
One	91.75	92	96	84	100
Two	89.25	82	86	85	100
Three	90	91	94	75	80
Four	100	86	88	71	100
Five	100	85	92	82	80
Six	77	93	96	84	90
Seven	84.25	91	96	82	95
Eight	76.25	89	93	80	100
Nine	74	97	98	80	100
Average	**86.9**	**89.6**	**93.2**	**80.3**	**93.9**

*threshold decomposition solution, ** near-convex decomposition solution

This comparison gives only a general idea of the performance, because overall, very few details are provided in the literature on the specific testing scenarios employed. Judging by the details provided, our approach seems closer in testing procedure to the one of [24], with three significant differences: (1) the distance between the camera and the hand is larger in our tests (950-1050 mm vs. 500-800 mm), (2) in [24] the results are calculated for both the left and the right hand and (3) tests in [24] seem to be related to a single blind style of testing. It can be noticed in Table IV that for the sign number recognition, our method outperforms most of the existing solutions, including the one of [24]. For the popular gestures, the proposed solution is in line with the existing literature, as it can be observed in Table V.

TABLE V
RECOGNITION RATES (%) FOR POPULAR GESTURES

Gesture	State-of-the-art solutions					Proposed solution	
	[24]	[26]	[16]	[21]*	[27]*	**	***
Open hand	99	-	-	-	-	80	80
Victory	89.2	87	83	90	91	95	100
OK	84.8	35	94	97	94	40	100
Letter L	91	87	95	96	94	100	100
Letter Y	90.5	77	94	96	97	90	90
I love you	85.2	-	-	-	-	100	100
Thumbs up	89.5	-	-	-	-	100	100
Thumbs down	88.2	-	-	-	-	100	100
Star Trek	84	-	-	-	-	60	60
Average	**89.1**	**71.5**	**91.5**	**94.8**	**94**	**85**	**92.2**

- no data available, * best solution, ** fingertips, *** fingertips and contour

The lowest performance was obtained for the more difficult to execute gestures OK and Star Trek, especially in the case when only the relative position of fingertips is used. For the Star Trek sign, the lower rate obtained by our solution might be explained by the fact that in the case of the [24], only two of four people were able to execute the gesture and results are reported for only these two, while in our case the users were asked to imitate the gesture to their best possible and all the users results were included in the evaluation even if the users were not able to execute the gesture correctly. Overall the performance improves when the contour is used as well in the decision making process, even for gestures including fingertips (e.g. OK). This can be explained by the fact that, according to the execution of a gesture by different users, the fingertips can be more or less visible or even lost during tracking, therefore engaging the use of the contour information. The addition of the contour comes however at a slightly higher computation cost (section E). Fig. 13 compares the performance of the static gesture recognition for the ASL alphabet executed with one hand and using the approach that includes the contour, with solutions proposed in the literature.

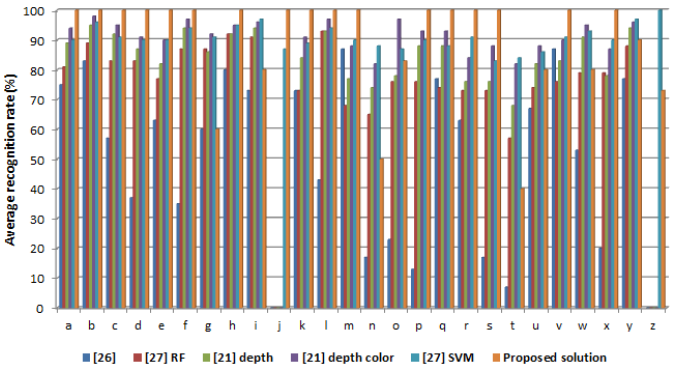


Fig. 13. Comparison for the gestures in the ASL alphabet.

The missing results for the "j" and "z" signs are due to the fact that these two letters are dynamic and several approaches work only on static data. It can be observed that the proposed solution (last column) is similar in performance to [27] for their SVM approach (penultimate column) with 89.85% average recognition rate over the alphabet letters for our solution and 90.85% in [27]. Table VI compares the performance with the solution of Zhu and Yuan [17] for the set of two hand gestures reported in their work. The performance of the proposed solution is in this case better.

TABLE VI
RECOGNITION RATES (%) FOR TWO HAND GESTURES

Gesture	[17]	Proposed solution
Both palms	99	100
Both fists	97	100
Left fist, right palm	98	100
Left palm, right fist	98	100
Left hand above right hand	99	100
Right hand above left hand	99	100
Average	**98.3**	**100**

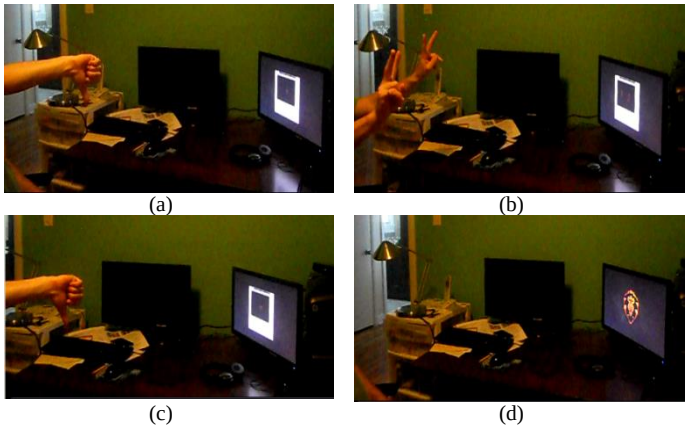


Fig. 14. Natural gesture software application control: (a) training the system with the thumbs-down gesture to close the application, (b) a series of other gestures is performed, (c) the thumbs-down gesture is executed, and (d) the application is closed 5 seconds after the gesture is recognized.

A final test for static gestures was performed for the context of natural control of a software application, as shown in Fig. 14. In particular, the thumb-down gesture was recorded as a reference gesture to close the interface of our system.

C. Dynamic Gesture Recognition

Additional tests were performed to recognize different dynamic gestures. A few test gestures are shown in Fig. 15 and the average recognition rates and a comparison of dynamic and static gestures are shown in Table VII. The results are presented for the proposed approach using single blind testing.

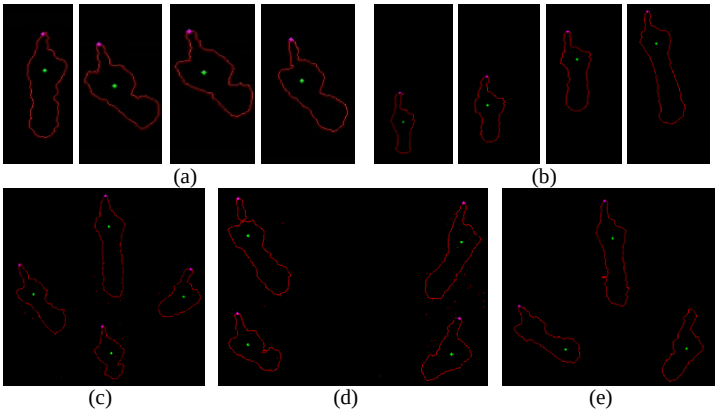


Fig. 15. Dynamic gestures: (a) click, (b) going up, (c) circle, (d) rectangle, and (e) triangle

TABLE VII
RECOGNITION RATES (%) FOR ONE HAND DYNAMIC GESTURES

Gesture	State-of-the-art solutions					Proposed solution
	[24]	[26]	[29]	[17]	[3]	
Circle	100	-	97.8	88	-	100
Bye	94.4	90	-	-	96.3	90
Rectangle	94	-	-	-	-	100
Triangle	99.4	80	-	-	-	95
Going up	-	-	100	94	89.2	100
Going down	-	-	100	93	91.9	100
Going left	-	-	97.8	95	96	100
Going right	-	-	100	95	95.5	100
Average	**96.95**	**85**	**99.12**	**93**	**93.72**	**96.25**

- no data available

One can notice that overall the system obtains good recognition rates for dynamic gestures as well. It is important to state the unlike the system proposed by [24], the one proposed in this paper (the case of relative position of fingers is reported) is performing the recognition in real-time.

D. Distinction between Static and Dynamic Gestures

Because the same solution is used to deal with both dynamic and static gestures, additional testing was performed to test the capability of the system to distinguish between the two as well as to compare with the case when the DTW is used only for validation, as in [14]. As stated in the section IID, the dynamic time warping has in this work a double role: to identify candidate gestures and to validate the similarity. Table VIII compares the proposed solution with the case when the DTW is employed for validation only. In spite of the fact that for our solution some difficulties are encountered with fast movements (e.g. fast consecutive clicks in the first line of Table VIII) and sometimes also in distinguishing between similar static and dynamic gestures, the results are overall better than when using DTW for validation only. The latter is affected by gestures that have the initial and final frame similar.

TABLE VIII
AVERAGE RECOGNITION RATES (%) FOR DTW FOR VALIDATION AND PROPOSED SOLUTION

Gesture description	[14]	Proposed solution
Distinction between click and immobile finger	80	100
Distinction between two hands with one immobile finger each and two simultaneous clicks executed with the finger of each hand	100	80
Distinction between immobile and moving hand	20	80
Average	**66.6**	**86.6**

E. Average Execution Time

Table IX shows the performance, in terms of the average execution time per frame, achieved by the proposed solution with respect to other state-of-the-art approaches. In spite of the fact that a direct comparison on the same computer is not possible, it can be noticed that the proposed solution is faster when compared with most of the approaches from the literature on an almost similar computing platform.

TABLE IX
AVERAGE TIME ESTIMATES PER FRAME (MS)

Solution	Computer configuration	No. of gestures	Time per frame
[9]	1.83Ghz processor	26	62.2ms
[29]	3GHz processor, 1GB RAM	7	60ms
[19]	2.66GHz processor, 3.3GB RAM	10	70ms*
[35]	3GHz processor, 3GB RAM	10	87ms
Proposed solution	2.3GHz processor, 3.3GB RAM	55	33ms** /40ms***

*threshold decomposition, ** fingertips, *** fingertips and contour

IV. CONCLUSION

The paper presented the development of a novel system for gesture recognition using depth images collected by a Kinect sensor. Starting from the closest pixel in the scene, the initial pixel of the hand contour is found using an improved block search scheme and a directional search is performed for the identification of the entire hand contour. The *K*-curvature algorithm is then employed to locate the fingertips over the contour. Based on hand position, the dynamic time warping algorithm is used to select a candidate gesture as well as to recognize gestures by comparing them with a series of pre-recorded reference gestures. The system achieves an average performance of 92.4% for one or multiple hand static and dynamic gestures, with which it is able to deal simultaneously. As only the depth sensor of Kinect is employed, the background, the lighting conditions, the clothing and the skin color have little impact on the resulted obtained by the proposed solution. Nevertheless, during testing some problems were encountered when one of the users wore a bracelet.

As future improvements, an additional module will be used to allow for a more accurate adaptation of the size of a user

hand instead of the double training at different depths that is currently implemented in the system. This would improve the performance at the level of recognition stage. In the same direction, a different version of DTW [41] will be evaluated for further speed up the gesture recognition.

The proposed solution has multiple applications within the realm of natural user interfaces, from which two were presented: one for the real-time interpretation of sign digits and popular gestures, the other one in the context of natural control of a software application.

REFERENCES

[1] A. R. Várkonyi-Kóczy, and B. Tusor, "Human-Computer Interaction for Smart Environment Applications Using Fuzzy Hand Posture and Gesture Models", *IEEE Trans. Instrumentation and Measurement*, vol. 60, no. 5, pp. 1505-1514, 2011.

[2] Q. Chen, N. D. Georganas, and E.M. Petriu, "Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar", *IEEE Trans. Instrum. and Meas.*, vol. 57, no. 8, pp. 1562-1571, 2008.

[3] C.-C. Hsieh, and D.-H. Liou, "Novel Haar features for real-time hand gesture recognition", *J. Real Time Image Processing*, vol. 10, pp. 357-370, 2015.

[4] S. Mitra, and T. Acharya, "Gesture Recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 311–324, 2007.

[5] J. Suarez, and R.R. Murphy, "Hand Gesture Recognition with Depth Images: A Review", *IEEE Int. Symp. Robot and Human Interactive Communication*, pp. 411-417, Paris, France, 2012.

[6] F. Trapero Cerezo, *3D Hand and Finger Recognition using Kinect*, Universidad de Granada, Spain, 2013.

[7] Z. Ren, J, Yuan, and Z. Zhang, "Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distance with a Commodity Depth Camera", *Proc. ACM Int. Conf. Multimedia*, pp. 1093-1096, 2011.

[8] R. P. Mihail, N. Jacobs, and J. Goldsmith, "Static Hand Gesture with 2 Kinect Sensors", *Int. Conf. Image Processing, Computer Vision, and Pattern Recognition*, vol. 2, pp. 911-917, 2012.

[9] D. Uebersax, J. Gall, M. Van den Bergh, and L. Van Gool, "Real-time Sign Language Letter and Word Recognition from Depth Data", *IEEE Int. Conf. Computer Vision*, pp. 383-390, Barcelona, 2011.

[10] K. Kollorz, J. Penne, and J. Hornegger, "Gesture Recognition with a Time-f-Flight Camera", *Int. Journal of Intelligent Systems Technologies and Applications*, vol. 5, pp. 334-343, 2008.

[11] K. Rimkus, A. Bukis, A. Lipnickas and S. Sinkevicius, "3D Human Hand Motion Recognition System", *IEEE Int. Conf. Human System Interaction*, pp. 180-183, Sopot, Poland, 2013.

[12] M. Correa, J. Ruiz-del-Solar, R. Verschae, J. Lee-Ferng, and N. Castillo, "Real time Hand Gesture Recognition using a Range Camera", J. Baltes et al. (Eds.), *RoboCup 2009*, LNAI5949, pp. 46-57, 2010.

[13] M. Van den Bergh, and L. Van Gool, "Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction", IEEE Workshop on Applications of Computer Vision, pp. 66-72, Kona, 2011.

[14] D.J. Ryan, *Finger and gesture recognition with Microsoft Kinect*, Master Thesis, University of Stavanger, Norway, 2012.

[15] M. Tang, "Recognizing Hand Gestures using Microsoft's Kinect", Univesity of Stanford, 2011.

[16] I. Oikonomidis, N. Kyriazis and A.A. Argyros, "Efficient Model-Based 3D Tracking of Hand Articulations Using Kinect", in *Proc. British Machine Vision Conference*, University of Dundee, UK, 2011.

[17] Y. Zhu, and B. Yuan, "Real-Time Hand Gesture Recognition with Kinect for Playing Racing Video Games", *Int. Joint Conf. Neural Networks*, pp. 3240-3246, Beijing, China, 2014.

[18] Z. Ren, J. Meng, and J.Yuan, "Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction", *IEEE Conf. Information, Comm. and Signal Proc.*, pp. 1-5, 2011.

[19] Z. Ren, J. Yuan, Ji. Meng, and Z. Zhang, "Robust Part-based Hand Gesture Recognition Using Kinect Sensor", *IEEE Trans. Multimedia*, vol. 15, no. 5, 2013.

[20] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool and M. Buss, "Real-time 3D Hand Gesture Interaction with a Robot for Understanding Directions from Humans", *IEEE Int. Symp.Robot and Human Interactive Communication*, pp. 357-362, Atlanta, USA, 2011.

[21] K. Otiniano-Rodriguez, and G. Camara-Chavez, "Finger Spelling Recognition from RGB-D Information using Kernel Descriptor", *IEEE Conf. Graphics, Patterns and Images*, Arequipa, pp. 1-7, 2013.

[22] M. Schroder, C. Elbrechter, J. Maycock, R. Haschke, M. Botsch, and H. Ritter, "Real-Time Hand Tracking with a Color Glove for the Actuation of Anthropomorphic Robot Hands", *IEEE-RAS Int. Conf. Humanoid Robots*, pp. 262-269, Osaka, Japan, 2012.

[23] H. Takimoto, S. Yoshimori, Y. Mitsukura, and M. Fukumi, "Classification of Hand Postures Based on 3D Vision Model for Human-Robot Interaction", *IEEE Int. Symp. Robot and Human Interactive Communication*, pp. 292-297, Italy, 2010.

[24] Y. Li, "Multi-Scenario Gesture Recognition Using Kinect", *IEEE Int. Conf. Computer Games*, pp. 126-130, 2012.

[25] X. Liu and K. Fujimura, "Hand Gesture Recognition Using Depth Data", *IEEE Conf. Automatic Face and Gesture Recogn.*, pp. 529-534, 2004.

[26] N. Pugeault, and R. Bowden, "Spelling it out: Real-time ASL fingerspelling recognition". *Conf. Consumer Depth Cameras for Computer Vision*, Barcelona, Spain, 2011.

[27] F. Pedersoli, S. Benini, N. Adami, and R. Leonardi, "XKin: An Open Source Framework for Hand Pose and Gesture Recognition", *Visual Computer*, vol. 30, pp.1107-1122, 2014.

[28] M. R. Abid, F. Shi and E. M. Petriu, "Dynamic Hand Gesture Recognition from Bag-of-Features and Local Part Model", *IEEE Conf. Computational Intelligence and Virtual Environments for Measurement Systems and Applications*, Ottawa, ON, Canada, pp. 12-17, 2014.

[29] X. Wu, C. Yang, Yé Wang, H. Li, and S. Xu, "An Intelligent Interactive System Based on Hand Gesture Recognition Algorithm and Kinect", *IEEE Int. Symp. Computational Intelligence and Design*, pp. 294-298, Huangzhou, China, 2012.

[30] N.H. Dardas and N. D. Georganas, "Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques", *IEEE Trans. Instrumentation and Measurement*, vol. 60, no. 11, pp. 3592-3607, 2011.

[31] M. Vanko, I. Minarik, and G. Rozinaj, "Evaluation of Static Hand Gesture Algorithms", *Int. Conf. Systems, Signals, and Image Processing*, Dubrovnik, Croatia, pp. 83-86, 2014.

[32] M.R. Abid, E.M. Petriu, and E. Amjadian, "Dynamic Sign Language Recognition for Smart Home Interactive Application Using Stochastic Linear Formal Grammar", *IEEE Trans. Instrumentation and Measurement*, vol. 64, no. 3, pp. 596-605, 2015.

[33] J. M. Palacios, C. Sagues, E. Montijano 2 and S. Llorente, "Human-Computer Interaction Based on Hand Gestures Using RGB-D Sensors", *MDPI Sensors Journal*, vol. 13, pp. 11842-11860, 2013.

[34] Z. Li, and R. Jarvis, "Real time Hand Gesture Recognition using a Range Camera", *Australasian Conference on Robotics and Automation*, Sydney, Australia, pp. 529-534, 2009.

[35] Y. Yao, and Y. Fu, "Contour Model based Hand-Gesture Recognition Using Kinect Sensor", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 24, no.11, pp. 1935-1944, 2014.

[36] F. Kobayashi, and F. Kojima, "Handover Motion Based on Human Hand Posture with Hand/Arm Robot", *IEEE Int. Symp. Micro-Nano Mechatronics & Human Sciences*, pp. 1-6, Nagoya, Japan, 2013.

[37] Y. Yao, and Y. Fu, "Contour Model-Based Hand-Gesture Recognition Using the Kinect Sensor", *IEEE Trans. Circuits and Systems for Video Technology*, vol. 24, no.11, pp. 1935-1944, 2014.

[38] P. Payeur, G. M. Nascimento, J. Beacon, G. Comeau, A.-M. Cretu, V. D'Aoust, and M.-A. Charpentier, "Human Gesture Quantification: An Evaluation Tool for Somatic Training and Piano Performance", *IEEE Symp. Haptic Audio-Visual Environments and Games*, pp. 100-105, Dallas, Texas, US, 10-11 Oct. 2014.

[39] E. Keogh, and C. A. Ratanamahatana, "Exact Indexing of Dynamic Time Warping", *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358-386, Springer, 2005.

[40] M. Muller, "Dynamic Time Warping", *Information Retrieval for Music and Motion*, pp.69-82, Springer, 2007.

[41] G. Al-Naymat, S. Chawla and J. Taheri, "SparseDTW: A Novel Approach to Speed up Dynamic Time Warping", *Proc. Australasian Data Mining Conference,* vol. 101, pp. 117-127, 2012.

G. Plouffe obtained his B.Sc. degree from Université du Québec en Outaouais and he is currently a Master student at the University of Ottawa. His research interests include human-machine interaction and gesture recognition.

A.-M. Cretu (S'04-M'10) obtained her M.A.Sc. and Ph.D. degrees from the School of Electrical Engineering and Computer Science at University of Ottawa, Canada. She is currently associate professor in the Department of Computer Science and Engineering at Université du Québec en Outaouais. Her research interests include computational intelligence, soft computing, biologically-inspired computational models, tactile and vision sensing, natural human-computer interaction and 3D object sensing, modeling and manipulation. She is the author of more than 60 technical papers. She serves as Technical Committee Member for several international conferences, as conference organizer and as a reviewer for several Journals and Transactions. She is member of the editorial board for the Springer Journal of Soft Computing. Dr. Cretu is a Member of the IEEE Instrumentation and Measurement Society, of the IEEE Computational Intelligence Society, of the IEEE Systems, Man and Cybernetics Society.