Extending the Interaction Area for View-Invariant 3D Gesture Recognition

Maurizio Caon1,2, Julien Tscherrig2, Yong Yue1, Omar Abou Khaled2 and Elena Mugellini2
1 Faculty of Creative Arts, Technologies & Science, University of Bedfordshire, Luton, United Kingdom
    e-mail: maurizio.caon@beds.ac.uk, yong.yue@beds.ac.uk
2 Department of Informatics, University of Applied Sciences of Western Switzerland, Fribourg, Switzerland
    e-mail: omar.aboukhaled@hefr.ch, elena.mugellini@hefr.ch

Abstract—This paper presents a non-intrusive approach for view-invariant hand gesture recognition. In fact, the representation of gestures changes dynamically depending on camera viewpoints. Therefore, the different positions of the user between the training phase and the evaluation phase can severely compromise the recognition process. The proposed approach involves the calibration of two Microsoft Kinect depth cameras to allow the 3D modeling of the dynamic hands movements. The gestures are modeled as 3D trajectories and the classification is based on Hidden Markov Models. The approach is trained on data from one viewpoint and tested on data from other very different viewpoints with an angular variation of 180°. The average recognition rate is always higher than 94%. Since it is similar to the recognition rate when training and testing on gestures from the same viewpoint, hence the approach is indeed view-invariant. Comparing these results with those deriving from the test of a one depth camera approach demonstrates that the adoption of two calibrated cameras is crucial.

Keywords—Image processing application, 3D gesture recognition, view-invariant, depth cameras calibration, Kinect, HMM.

I. INTRODUCTION

The current trend of Human-Computer Interaction (HCI) research is adopting the human-human communication modalities for human-machine interfaces. In particular, the gestural interaction is becoming very popular in the HCI research community [1]. The importance of gestural interaction grew rapidly in the last decade especially because of the interest shown by the industries. In particular, this is evident in the entertainment domain (e.g., Nintendo Wii, Sony PlayStation Move, Asus Xtion Pro, et cetera). Vision-based technologies play an important role in this domain. In particular, 3D cameras are positioned to become one of the most important technologies for gesture recognition. In fact, 3D cameras combine the principal advantage of non-intrusive sensing systems to some other important features, such as the illumination changing resilience and the facilitated users’ body segmentation from cluttered background. Moreover, the depth information provided by these cameras is very important also to track the user in the environment, adding his/her relative position with reference to the sensor. In this spirit, Microsoft presented the Kinect: a motion sensing device that integrates both a 3D camera and a standard RGB camera. The Kinect is a very cheap off-the-shelf device which can provide quite accurate depth information at a good frame-rate (30 Hz) [2]. Its sensing features added to the very low price in comparison with other depth cameras technologies make the Microsoft Kinect a very appealing device do develop natural interaction applications for everyday activities. Thanks to the aforementioned advantages, the Kinect found its natural application in gesture recognition for HCI, e.g., [4]. The research community has already manifested strong interest in this device, also for applications that go far beyond simple video-gaming, e.g., [3].

Designing a gestural interface for interactive environments, such as smart homes, introduces the need for recognizing the gestures performed by the user in a wide area and independently of the user’s orientation. In fact, we can imagine a smart living room that allows the user to interact with several devices distributed in the room (e.g., lamps, TV, hi-fi, et cetera) through gestures (Fig. 1). The user can vary his/her position and orientation, but the system should grant continuous 3D gesture recognition for a seamless interaction experience. This example introduces the issue related to the vision-based gesture recognition that should be independent of the cameras viewpoints. The Microsoft Kinect technology allowed us to design and implement a non-intrusive approach for view-invariant gesture recognition. In this paper, we describe a calibration procedure to exploit the 3D information coming from two Kinect devices. Using two Kinects positioned in an opportunistic configuration permits avoiding the occlusion of user’s limbs [5]. The continuous user’s body tracking allows modeling the dynamic hands gestures and recognizing them through a Hidden Markov Models (HMM) classifier.

The rest of the paper is organized as following. Section 2 presents the related work. The system architecture is described in Section 3. The calibration procedure to use two Kinects is explained in detail in Section 4; our gesture recognition approach is described in Section 5. Section 6 presents the tests that have been made in order to evaluate the system. Section 7 is dedicated to the conclusion reporting also the future work.

II. RELATED WORK

Gestural interaction has been adopted in many different domains. Interacting with a system without touching physical devices is crucial in an operating room [6][7], is intuitive to communicate with a robot [8][9] and can be very convenient in a smart home [10] or a virtual environment [11].

Fig. 1. Gestural interaction in a smart living room.
Most of the vision-based gesture recognition systems are influenced by the position of the users with reference to the sensing cameras. The different position of the users can severely compromise the recognition process. As many applications of gesture recognition are in smart environments or in multi-display environments or for immersive virtual reality, the possibility of performing gestures in wide interactive areas in every posture and position in the space becomes very important. Therefore, some works focused on developing vision-based systems for view-invariant gesture recognition. Using a 2D camera imposes obvious limitations in 3D gesture recognition and usually it makes the system strongly dependent on the specific view-point. A solution is using multiple RGB cameras as in [12]. In this paper, the authors extracted voxel data from the different silhouettes coming from several RGB cameras to have a 3D reconstruction of the user’s body. After a phase of post-processing of these data, they used multilinear analysis for the posture recognition and the HMM for the gesture recognition. The authors of [13] integrated only a stereo-camera in their gesture recognition system. Using the disparity map provided by the cameras, this particular appearance-based method extracted the volume motion template, estimated the optimal virtual view-point and classified the data using the least square method and k-nearest-neighbor algorithm. Unfortunately, RGB cameras are not resilient to illumination changing and they limit the application of these systems in controlled environments. Yuan et al. avoided the lighting changing and the hand-tracking problems using two infrared (IR) cameras and a retro-reflective marker [14]. This system recognized only 3D trajectories of non-directional gestures that approximately lie on a plane. Moreover, a strong disadvantage of this approach was the intrusiveness. Holte et al. proposed an interesting approach based on the 3D motion detection primitives determined from the intensity and range images provided by a Time-of-Flight (ToF) camera [15]. The test results demonstrated that this system recognized four arm gestures with good precision; but the user could vary his angulation only between -45° and +45° with reference to the optical axis of the ToF camera.

In this paper, we introduce our system that integrates two calibrated Microsoft Kinects to recognize 3D hand trajectories. Our system recognizes in real time the 3D user’s hand gestures performed from a -90° and +90° view-point with respect to the center between the two depth cameras. Moreover, this system can be trained to recognize the recorded gestures even if they have been performed in a singular position. The view-invariant gesture recognition provided by this approach can find many applications in the interactive smart environments or for the interaction in a virtual reality cave.

### III. SYSTEM OVERVIEW

Figure 2 shows the system architecture. Two modules based on the OpenNI libraries [16] are dedicated to track the user. Each module constructs a skeleton model of the tracked user that will be represented in a specific XML structure. These data are sent to the “Kinect Data Merger” module that applies the coordinates transformation to represent the skeletons provided by the acquisition modules in the same space reference frame. These modules communicate with XML messages using the UDP protocol. Every XML message contains the information about the coordinates of every user’s joint and the ID number of the Kinect camera that sent these data. This module executes all the calibration function. The “Kinect Data Merger” sends the transformed coordinates to the “Gesture Recognition” block. This functional block is composed of three different modules. The first one reconstructs the 3D model of the tracked user. The user’s skeleton is represented as a half 3D stickman composed of the following joints: head, neck, right and left shoulders, right and left elbows, right and left hands, torso, right and left hips (Fig. 3). When the gesture segmentation is triggered, the second module of this block receives the 3D coordinates of the user’s hands and transforms them to a space reference frame that has origin between the user’s shoulders. When the gesture segmentation is finished, the transformed 3D hands coordinates are sent as a unique sequence that we call 3D trajectory; the receiver of this 3D trajectory is the HMM module that recognizes the performed gesture.

This architecture has been designed to be modular. Indeed, it allows configuring the system to work with only one Kinect as sensing device. The possibility of changing the system configuration between one Kinect and two Kinetics is due to the XML protocol for the description of the joints. Therefore, to use the configuration with one Kinect we have only to set the UDP ports in order that the active “Kinect Acquisition Module” sends the user’s joints coordinates.
directly to the “3D User Modeling” module. In this way, we can compare the gesture recognition performances between a solution that integrates only one depth camera and one with two calibrated Kinecs, as explained in Section 6.

IV. CALIBRATION PROCEDURE

The system presented in this paper uses two Microsoft Kinect cameras and this involves their calibration. The calibration procedure is composed of two consecutive steps. The first one consists of the calibration between the IR and the RGB cameras in each Kinect. The second step is the calibration between the two different Kinects.

A. Calibration between IR and RGB cameras

The acquisition of the common points in the 3D space by the Kinects is obtained using a simple checkerboard as shown in Fig. 5. When the checkerboard is seen by both the Kinects, its center is calculated processing the RGB cameras images with OpenCV [17]. Afterwards, the extracted points for every synchronized image are associated to the depth value extracted by the depth cameras. In order to obtain the depth value of each pixel captured by the RGB cameras, we need to execute the calibration of the RGB camera and IR camera for every Kinect. For this procedure we extend the work made by Van den Bergh and Van Gool [18] for the calibration of a RGB camera and a ToF camera. In order to obtain a good corner detection, the IR laser projector of the Kinect must be blocked and the checkerboard must be suitably illuminated by an IR light source (e.g., a halogen lamp), Fig. 4. We calculated the intrinsic and extrinsic parameters of the RGB and IR cameras using the Matlab camera calibration toolbox [19]. Indeed, we calculate the pixel \( \mathbf{p}_{\text{IR}} \) (expressed as a 2 \times 1 matrix) in the RGB image coordinates as (given that \( Z' \) is not zero)

\[
\begin{bmatrix}
X' \\
Y' \\
Z'
\end{bmatrix}
= \begin{bmatrix}
x_{\text{IR}}^{-1} & 0 & c_{x,\text{IR}} \cdot f_{x,\text{IR}}^{-1} \\
0 & f_{y,\text{IR}} & c_{y,\text{IR}} \cdot f_{y,\text{IR}}^{-1} \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_{\text{IR}} \\
y_{\text{IR}} \\
1
\end{bmatrix} + T
\]

(2)

The pixel \( \mathbf{p}_{\text{IR}} \) is represented with the \( x_{\text{IR}} \) and \( y_{\text{IR}} \) coordinates in the IR image and with the associated depth value \( (z_{\text{IR}}) \). \( R \) is the rotation matrix and \( T \) is the translation matrix. The \( f \) and \( c \) coefficients represent the intrinsic parameters. Finally, the depth value corresponding to the location \( \mathbf{p}_{\text{RGB}} \) in RGB image is \( Z' \).

B. Calibration between the two Kinects

Every Kinect camera measures the distance to each visible point on an object to create a collection of distances called depth map. The 3D points captured by a single Kinect are expressed referring to its own reference frame that has the origin in the Kinect’s depth camera with the \( z \)-axis pointing out of the camera (Fig. 5). For the calibration between two Kinects we need two 3D point sets that represent the same point in the 3D space viewed by the two Kinects. We call \( p_i \) the first set of points with coordinates in the reference frame associated to the first Kinect; \( p_i' \) is instead the set of the same points in the 3D space captured by the second Kinect. Here, \( p_i \) and \( p_i' \) are considered as 3 \times 1 column matrices. Therefore, we can represent the transformation of the points expressed in the coordinates of the first Kinect’s reference frame to the second Kinect’s reference frame as

\[
p_i' = R \cdot p_i + T + N_i
\]

(3)

\[
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix}
= \begin{bmatrix}
X', Y', Z' + N_i
\end{bmatrix}
\]

(1)

Where \( X', Y' \) and \( Z' \) are the 3D coordinates with respect to the RGB camera.

Fig. 4. Calibration between the IR and RGB cameras of the Kinect; a) the IR image with an IR source; b) RGB image of the same scene; c) depth image of the same scene; d) IR image with the unblocked IR laser projector.

Fig. 5. Calibration between two Kinects using a checkerboard; a) disposition of the Kinects and the checkerboard; b) the RGB image captured by the Kinect on the left and elaborated with OpenCV; c) the RGB image captured by the Kinect on the right and elaborated with OpenCV; d) depth image captured by the Kinect on the left; e) depth image captured by the Kinect on the right.
Where $R$ is a $3 \times 3$ rotation matrix, $T$ is a translation vector ($3 \times 1$ column matrix), and $N$, a noise vector. We want to find the $R$ and $T$ matrices that minimize

$$
E^2 = \sum_{i=1}^{N} \| p'_{i} - (R \cdot p_{i} + T) \|^2
$$

We chose to use a non-iterative algorithm which involves the Singular Value Decomposition (SVD) of $R$. In [20], it has been demonstrated being more efficient in terms of time requirements for the computation of the number of points we are interested in. Following this approach, we calculate the centroids of the 3D point sets

$$
p' \triangleq \frac{1}{N} \sum_{i=1}^{N} p'_{i}
$$

$$
p \triangleq \frac{1}{N} \sum_{i=1}^{N} p_{i}
$$

And

$$
q_{i} \triangleq p_{i} - p
$$

$$
q'_{i} \triangleq p'_{i} - p'
$$

Then we calculate the $3 \times 3$ matrix and we find the SVD of it

$$
H = \sum_{i=1}^{N} q_{i}q'^{T}_{i} = U \cdot A \cdot V^{T}
$$

Finally, we calculate

$$
R = V \cdot U^{T}
$$

And

$$
T = p' - R \cdot p
$$

Equations (10) and (11) provide the $R$ and $T$ matrices that minimize $E^2$ and that we use to calculate the coordinates transformation.

\section*{V.Gesture Recognition}

The “Kinect Data Merger” (see Fig. 2) module collects the user’s joints data coming from the two Kinects, applies the transformation matrix to represent both the 3D skeletons in the same spatial reference frame. Once the two 3D skeletons are calculated, this module makes the fusion of the data concerning the tracked user to create a unique 3D skeleton. The joints that are considered during the fusion process must have the maximum value of the associated reliability factor that is provided by the OpenNI libraries. The “3D User Modeling” module receives the merged user’s skeleton coordinates and represents them as shown in Fig. 3. When the segmentation starts, the system tracks the hands joints and records the 3D trajectories. The “3D Trajectory Modeling” module makes the transformation of the trajectories coordinates from the spatial reference frame associated to the Kinects to the reference frame that has origin between the shoulders of the 3D skeleton. After this coordinates transformation, the 3D trajectories are sent to the HMM module. In fact, the classification is based on the ergodic HMMs with four hidden states. We use the Baum-Welch algorithm to find the unknown parameters as implemented in the Accord.NET Framework [21].
The three gestures that were chosen for this evaluation are “left”, “right” and “circle” as shown in Fig. 8. “Left” gesture started when the user put his/her hands together in front of his/her chest, then he/she extended the left arm horizontally to the side. “Right” gesture started when the user put his/her hands together in front of his/her chest, then he/she extended the right arm horizontally to the side. “Circle” gesture started when the user put his/her hands together in front of his/her chest, then he/she raised the arms up and performed a complete rotation of both arms. We chose these gestures because the “left” and “right” are directional gestures, “circle” is a non-directional gestures. Hence, we evaluated the system with both types of dynamic gestures.

During the whole system evaluation, the 10 users performed a total amount of 1860 gestures, 930 gestures for each configuration. Therefore, each test procedure involved 930 gestures of which 330 were for the training phase and 600 for the second phase. The confusion matrix of the evaluation of the configuration with one Kinect is reported in Table 1. Table 2 reports the confusion matrix of the evaluation of the system configuration with two calibrated Kinects. The overall gesture recognition rate divided by the five user’s positions for both the system configurations is reported in Fig. 9. In particular, during the evaluation of the system configuration with one Kinect we obtained the following recognition rates: 100% in position 1, 92.5% in position 2 and 3, 44.2% in position 4 and 48.3% in position 5. The tests conducted with the proposed system configuration that integrates two calibrated Kinects gave as results: 100% of gesture recognition rate in the position 1, 98.3% in position 2, 96.7% in position 3 and 4, and 94.2% in position 5. These data confirmed that our non-intrusive 3D trajectory approach with the classification based on HMM provided excellent results for both types of dynamic gestures. Moreover, the system configuration with two calibrated Kinects obtained recognition rates higher than 90% also in the positions 4 and 5, when the configuration with one Kinect gave rates lower than 50%. Therefore, using two calibrated Kinects allowed extending the interaction area to an angular orientation of 180°, when the configuration with one Kinect granted good results only for an angular orientation comprised between -45° and +45° as for the approach proposed by Holte et al. in [15].

View-invariant gesture recognition provides a more natural environment for an advanced HCI. Although it is difficult to recognize the gestures independently of the camera viewpoint, the depth camera technology allowed us to develop a simple, cheap and non-intrusive approach for view-invariant gesture recognition. The proposed approach finds many applications such as interaction in virtual reality environments and smart homes. In this paper, we proposed a calibration procedure for two Kinects in order to model the user’s hands movements as 3D trajectories. Since HMM has been widely used in modeling spatio-temporal applications, we implemented a HMM classifier to recognize 3D gestures trajectories. 10 users tested the system performing three chosen gestures in five specific different positions; these positions have been thought in order to assess the recognition process in multiple angles in reference to the Kinects’ viewpoints for a total angular variation of 180°. The recognition rates resulted from this evaluation demonstrated that our approach is valid to represent and recognize all the dynamic hand gestures independently of the depth cameras viewpoints. The same gesture recognition process has been tested using the information captured by only one Kinect. The test results demonstrated that to achieve the view-invariant gesture recognition on 180° it is necessary to use two calibrated Kinects.

As next step we are planning to extend our system to the use of three calibrated Kinects in order to allow the user to interact with an environment that can recognize his/her gestures independently of a full 360° angular variation. In fact, the modular architecture of the proposed system has been thought to grant an easy integration of more than two Kinects. Preliminary tests with three Kinects demonstrate that the calibration procedure can be applied to multiple Kinects, but the management of the different user’s skeletons highly increases the complexity of the “Kinect Data Merger” module. Using the OpenNI libraries, the skeletons initialization limits the interactive area. For this reason this system has been adapted to work also with the Kinect for Windows SDK [22], which provides the automatic skeleton initialization. On the other hand, the skeleton tracking provided by Microsoft SDK is not very stable and imposes the development of a more complex fusion algorithm that we are currently implementing.
Table 1. Confusion matrix for the evaluation of the system configuration with one Kinect; the number of recognized gestures are reported for every gesture and every position, and the relative percentage is between parentheses.

<table>
<thead>
<tr>
<th>1 Kinect configuration</th>
<th>Recognized gesture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left</td>
</tr>
<tr>
<td>Position 1</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>40 (100%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 2</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>36 (90%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 3</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>37 (92.5%)</td>
</tr>
<tr>
<td>Right</td>
<td>1 (2.5%)</td>
</tr>
<tr>
<td>Circle</td>
<td>1 (2.5%)</td>
</tr>
<tr>
<td>Position 4</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>7 (17.5%)</td>
</tr>
<tr>
<td>Right</td>
<td>2 (5%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 5</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>24 (60%)</td>
</tr>
<tr>
<td>Right</td>
<td>9 (22.5%)</td>
</tr>
<tr>
<td>Circle</td>
<td>6 (15%)</td>
</tr>
</tbody>
</table>

Table 2. Confusion matrix for the evaluation of the system configuration with two Kinets; the number of recognized gestures are reported for every gesture and every position, and the relative percentage is between parentheses.

<table>
<thead>
<tr>
<th>2 Kinets configuration</th>
<th>Recognized gesture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Left</td>
</tr>
<tr>
<td>Position 1</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>40 (100%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 2</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>39 (97.5%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 3</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>39 (97.5%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 4</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>37 (92.5%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Position 5</td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>40 (100%)</td>
</tr>
<tr>
<td>Right</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Circle</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>

REFERENCES