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

Gesture Control Interface for immersive panoramic displays

M. Alcoverro · X. Suau · J.R. Morros · A. Gil · J. Ruiz-Hidalgo · J.R. Casas

Received: date / Accepted: date

Abstract Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

Keywords First keyword · Second keyword · More

1 Introduction - JR

The need of gesture control interfaces emerges due to an increasing complexity of the functionalities of information systems and the current demand of more natural user interfaces. Commercially available display sets, for instance, offer larger degrees of freedom to manipulate the appearance of the displayed content. Users tend to feel constrained by traditional interfaces, such as remote controls, as the rendering capabilities of the terminal evolve. This compromises the principle of a natural user interface []

A gesture is a non-verbal, natural communication made with a part of the body

2 Related work

User interaction with the TV is typically performed using conventional remote controls with physical buttons or muti-touch devices. However, the need for enabling interaction with new types of data, such as high-quality panorama or 3D video, and the increasing diversity of content the user should be able to choose, has fostered the study of new input modalities to control TV systems. Remote controls have two major drawbacks: they can be misplaced or out of reach of the user, and they require a lot of attention by the user as its control functionalities increase. Current research is focusing on natural user interfaces based on gesture and speech recognition to overcome such drawbacks, following studies from the generic field of Multimodal Human Computer Interaction (MMHCI). MMHCI is a widely studied field and some surveys have been previously published [11][25]. Moreover, MMHCI is at the crossroad of several related areas extensively studied such as face detection and identification [33], facial expression analysis [19], eye tracking [12], gesture recognition [29], human motion analysis [20] or audio-visual automatic speech recognition [21].

The recent commercialization of new game console controllers as Kinect or Wiimote, has been rapidly followed by the release of proprietary or third part drivers and SDKs suitable for implementing new forms of 3D user interfaces based on gestures [7]. On the one side, several authors propose gesture control interfaces based on accelerometers as the Wii controller [24][14]. On the other side, the Kinect depth sensor allows for device-less interfaces. Some solutions as ZigFu [34] or GesturePak [10] use the skeleton tracking SDKs [13][18] as input for gesture recognition based on skeletal poses. Thus, such approaches require a human pose estimation step, which is a complex task with high computational cost, often prone to errors in presence of clutter. Alternatively, several approaches make use of raw Kinect depth data as input for hand pose and gesture recogni-

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n248138. This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación, under project TEC2010-18094

Universitat Politècnica de Catalunya
Department of Signal Theory and Communications
Barcelona, Spain
E-mail: marcel.alcoverro,xavier.suau,albert.gil,ramon.morros, j.ruiz,josep.ramon.casas@upc.edu

tion methods [23][30][22]. In contrast to 2D color video, the use of depth data makes such methods robust to illumination changes and suitable for dark environments. Moreover, depth data provides 3D information valuable to account for scale invariance of human body parts. In general, these features make depth based methods perform better than its color-based counter part.

As introduced by Wachs et. al. [32], gesture-based interfaces for entertainment applications, such as TV control, must address to major issues: *intuitiveness* and *gesture spotting*.

By *intuitiveness*, one means that the types of gesture selected should have a clear cognitive association with the functions they perform. However, a gesture natural to one user may be unnatural to others, due to the strong associations with cultural background and experience. Stern et. al. [27] proposed a method to design and evaluate gesture vocabularies taking into account psycho-physiological measures (e.g. intuitiveness, comfort) together with machine factors (recognition accuracy). Alternatively, Nielsen et. al. [16] propose a procedure to design the gesture vocabulary based on the Wizard-of-Oz paradigm. In a Wizard-of-Oz experiment, user interact with the system, but the response of the system is simulated by having a person respond to the user commands. In this way, the users are the ones who decide which gesture best represent its intentions.

Gesture spotting [32] consists of distinguishing useful gestures from unintentional movement. This problem may be afforded by the recognition technique, by performing a temporal segmentation to determine where the gesture start and ends. However, this is a difficult problem and often the recognition methods assume temporally segmented actions [5]. Also the recognition may require spatial segmentation of the body parts (e.g hands) which may be also a task prone to errors. To overcome this problem, López-Méndez et. al [15] focus on the problem of localization of static gestures on depth data. Their method learns the local appearance of gestures, and neither temporal nor spatial segmentation are required.

3 System Design

The proposed system works in a device-less and marker-less manner allowing users to control the content on their TV sets only using their hands (Figure 1). It is responsible of locating the people that want to interact with the system and of understanding and recognizing their gestures.

The current implementation is focused on controlling the content on a high definition TV screen situated in a home scenario. The content is composed of a panorama image (a high resolution view of the scene) together with several audio tracks and Regions of Interest (ROIs) associated with the panorama.

Fig. 1 Several users interacting with the proposed gestural interface.

Fig. 2 Detail of the displayed content and the user feedback. The icon on the bottom centre (above the bar) is used to give feedback about static gesture commands (the icon shows that the user is lowering the volume). The bar on the bottom shows the 5 available static gestures.

The current setup of the system allows the user to be standing or seating in a chair or coach. Although a single user is able to interact with the system at any given time, the interface is multi-user in the sense that several users can ask for control of the system and interact with it while the others might still be present in the scene (Figure 1).

In order to allow the user to interact with the TV content, the system supports the following functionalities:

Menu selection A menu is overlaid on the current screen and the user can select any button of the menus by pointing at it (Figure 2).

Navigation The user is able to navigate through the panorama scene by panning, tilting and zooming in the content.

ROI Selection The system informs the user of the available ROIs in the current view and the user is able to change between them.

Pause / Resume The user is able to pause / play the content at any time.

Audio volume The user is able to increase or decrease the volume and to mute it completely.

Take control The control of the system can be passed between users.

3.1 User Experience and GUI design

The system design is a compromise between the best user experience we want to create and the best or simplest scenarios from a technical perspective. For instance, users might be willing to perform gestures laying down in the couch, or too similar gestures, but that could produce recognition errors or too much computing time. To ensure that the system is always providing a responsive, convenient and intuitive experience for the user, several requirements and design strategies have been adopted:

- The interface must provide constant feedback to the user. The identified user, the detected gestures, or where the detected hands are located and pointing. Like the common mouse or notification icons in a computer.
- Menus on the screen should contain up to 4 options to make them easy to point and select, and to avoid to overlap too much of the watched scene
- Gesture shortcuts should be easy to learn and use and should provide a fast way to do simple tasks like controlling the volume. They are one of the key functionalities of our proposed system.
- To allow the social aspects of interaction with TV sets, the system should provide an easy mechanism to release, take/switch the control between multiple users; and the user in control should be free to move around, but in the sensor range.
- Panoramic navigation and gestures recognition modes are separated to improve user experience and at the same time reduce false/wrong detections.

3.2 Architecture

To fulfill all the functionalities, the current system implements the architecture depicted in Figure 3. It is divided in three main layers. In the first layer, the **Capture** component is responsible of communicating with a single Kinect camera and feeding the images into the system. The Kinect sensors provide color and depth video with VGA resolution that are feed into all other components of the system.

The second layer is the core of the interface. Here the processing, detection and recognition algorithms take place. It is composed of the following components:

The **Head tracking** (Section 4.1), where the depth image obtained in the capture module is analyzed to detect heads (oval areas of the same depth) [28]. The position of the heads are used in all subsequent components to locate possible users of the system. Other persons in the field of view of the Kinect sensor are not tracked and their gestures do not interfere with the system.

The **Face identification** component (Section 4.4) recognizes users. Faces are detected using a modified Viola-Jones detector [31] on the color images and recognized using a temporal fusion of single image identifications. The recognized users will determine the number of people able to control the system.

Once users are detected, hands are tracked by the **Hand Tracking** component (Section 4.2) using a 3D virtual box in front of the head of the user with control of the system. 3D blobs in the virtual box are segmented and treated as hands [28].

The **Gesture classification** component (Section 4.3) is responsible of detecting and classifying static gestures (gestures performed with still hands). In this case, the classification of the gesture is purely done using the shape and position of the hand, extracted using only the depth data provided by the Capture component. The classification is based on random forests [4], aiming at accurately localizing gesture and object classes in highly unbalanced problems.

Finally the third layer of the architecture contains the **Application Control** component. This module is responsible of acquiring all the detections and tracking information obtained by the components in the middle block (head, hands, user recognized and classified gestures) and mapping them into the functionality listed in the previous section. It communicates with the TV to perform all the needed interactions (select ROIs, change volume, pan-tilt-zoom the panorama, etc.). It controls the user interface and provides all the needed feedback through the GUI overlaid in the TV (Figure 2).

3.3 State Diagram

This section describes the state diagram of the current system and further explains the relation between the different components in the system architecture. Each state in the diagram allows users to perform specific gestures to control the system. The **Application Control**

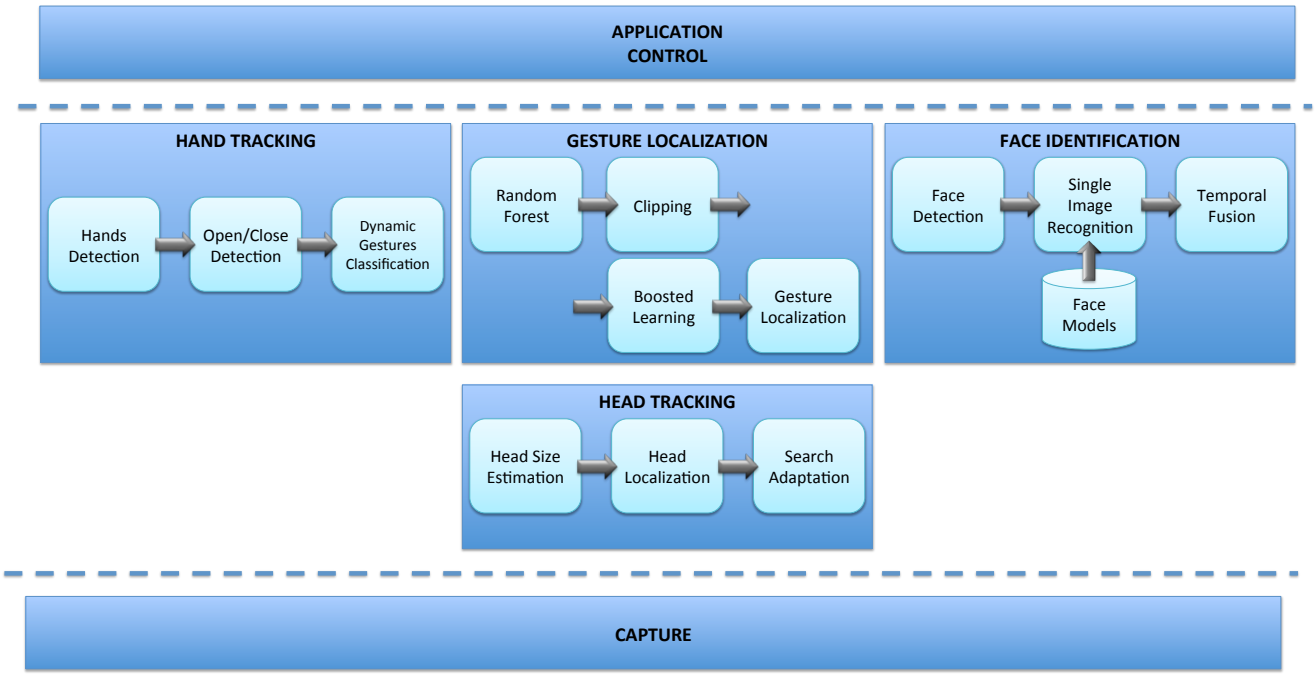


Fig. 3 Architecture of the gesture control interface.

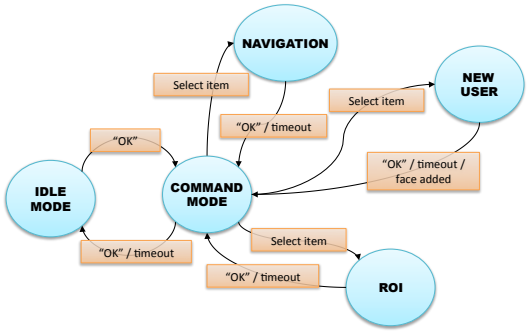


Fig. 4 State diagram for the gestural interface.

module is responsible of controlling the general status of the system and performing the transition between states. The end user does not need to be aware of the general state diagram or the current state as the **Application Control** component informs the user about the available commands at each stage through feedback in the GUI overlay. Figure 4 shows the different states possible. The arrows and yellow boxes indicate the gesture or timeout that will transition the interface from one state to the other one.

– The system always starts in *idle mode*. In this state no menu is shown in the screen and the only gesture available is the *OK* or *take control* gesture (Figure 9). In this state, the Gesture Classification component only recognizes the *OK* gesture and all other components are suspended. If the *OK* gesture is detected by the Gesture Classification component, the user doing the gesture takes control and the interface transitions to the next state. In all following states only the user in control is recognized and other users might be present in the scene without affecting the system.
– The *command mode* is the the principal state where users interact with the system. In this state all the components (Head and Hand Tracking, Gesture Classification and Face Identification) are started and report to the Application Control. The Head Tracking ensures that the user is followed and that all sub-sequent components only focus on the user controlling the system. The Hand Tracking tracks the position of a single hand which is used to access the menus of the GUI overlay. The selection is done by by pointing a single hand to the menu item, wait for a few seconds until an arrow is shown, and then moving the hand in the direction of the arrow (like grabbing the menu item to the centre of the screen). The Face Identification recognize users and allows them to access specific options or interactions with the system.
Finally, the Gesture Classification component recognizes 5 static gestures (Figure 9) and the Application Controls maps them to the following functionalities:
  ◦ The *OK* gesture (the same one the user used to take control of the system) is used to go back to *idle mode*.

- The volume can be increased by locating the finger on top of the mouth and lowered by putting the hand on the ear.
- The volume can be completely muted by doing a cross sign in front of the mouth.
- The video can be paused or resumed by closing the hands together (similar to clapping).

By selecting any of the menus shown in the GUI overlay, the interface can transition to the states of navigation, ROI selection or new user. From all these states the user can go back to command mode by doing the *OK* gesture or by waiting several seconds without doing anything to trigger a timeout.

– The *navigation mode* allows the user to navigate freely through the panorama. In this mode no menus are shown in the screen and only the hands of the user are overlaid in the screen. The user can then pan and tilt the panorama by using one hand and grabbing the scene and zoom in or out by using both hands (such as done in maps applications on tablets). The Hand Tracking component is responsible of tracking this movements and, in this state, no Gesture Classification is performed as experimental results showed and increase of false/positive detections.
– The *ROI mode* allows the user to navigate through the available ROIs by moving your hand left to right or right to left. Also, the Hand Tracking algorithms is responsible to track the hands of the user to detect grabbings and movements in this state with no possibility of doing static gestures.
– Finally, the *new user mode* is used to add new users to the system. This mode creates new user models to be used in the Face Identification component.

4 System Components

This section further describe each of the components (Head and Hand tracking, Gesture Localization and Face Identification) presented in the architecture of the system.

4.1 Head Tracking

The head tracking algorithm is composed of three steps: head size estimation, head localization and a search area resizing.

Head size estimation For every foreground pixel, an elliptical template ($\mathcal{E}$) of the size of a regular adult head (about 17×23 cm) is placed at the pixel's depth level d and projected onto the camera image plane, obtaining an ellipse of the apparent head size (H_x, H_y) in pixel dimensions (see Figure 6). This ellipse is called *template* or $\mathcal{E}$ is then used to find a head location estimate.

We assume that people interacting with the TV will either stand-up or be seated, keeping their head more or less vertical.

Head localization Aiming at finding the image region which better matches ($\mathcal{E}$), a matching score between ($\mathcal{E}$) and the global foreground mask ($\mathcal{F}$) is calculated at every pixel position (m, n) of the image. Matching is calculated within a rectangular search area of size $R_x \times R_y$, by sliding $\mathcal{E}$ across the image. The matching score is calculated according to conditions C^k presented in (1), where $b = background$ and $w = foreground$. Conditions are checked at every pixel position $(u, v) \in \mathcal{E}$ of the template, which is itself centered at (m, n). When a condition C^k is satisfied, $C^k = 1$, otherwise $C^k = 0$. The final matching score for the pixel (m, n) is calculated as the sum of all the scores obtained on the template pixels, as shown in Equation (1).

$$
\begin{aligned}
&C^1_{u,v} : (\mathcal{E}_{u,v} = b) \ \wedge (\mathcal{F}_{u,v} = b) \\
&C^2_{u,v} : (\mathcal{E}_{u,v} = w) \wedge (\mathcal{F}_{u,v} = w) \wedge (|d_{u,v} - d_{H_i}| < d_{max}) \\
&C^3_{u,v} : (\mathcal{E}_{u,v} = b) \ \wedge (\mathcal{F}_{u,v} = w) \wedge (|d_{u,v} - d_{H_i}| > d_{max})
\end{aligned}
$$

$$
\mathcal{M}_{m,n} = \sum_{\forall (u,v) \in \mathcal{E}} (C^1_{u,v} + C^2_{u,v} + C^3_{u,v}) \tag{1}
$$

The pixel (m, n) with a higher score $\mathcal{M}^H$ is selected as the best head position estimation $\hat{p}_H$ in a given search area. Conditions C^2 and C^3 provide robustness against clutter and partial occlusions, incorporating depth information to the matching score.

Search adaptation The position and size of the search area (R_x, R_y) is adapted to the head position variance (σ_x, σ_y), and also to the confidence on the estimation $\bar{\mathcal{M}} = \frac{\mathcal{M}^H}{\mathcal{M}^{max}} \in [0, 1]$, as described in Equation (2).

$$
\begin{aligned}
R_x &= \sigma_x + (1 + \mu) \cdot H_x \\
R_y &= \sigma_y + (1 + \mu) \cdot H_y
\end{aligned}
\qquad with \qquad \mu = e^{\frac{-\bar{\mathcal{M}}}{1-\bar{\mathcal{M}}}} \tag{2}
$$

Such resizing is effective against fast head movements. For example, horizontal movements will enlarge the search area along the horizontal axis. Furthermore, including the matching score in Equation (2) makes the system robust against noisy estimations.

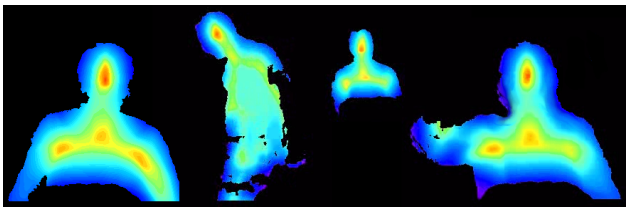


Fig. 5 Head matching score values using a Kinect sensor. Different situations are presented (from left to right): back view, side view (with slight head tilt), far person and long-haired person. In all these cases, the matching score presents a maximum in the head zone.

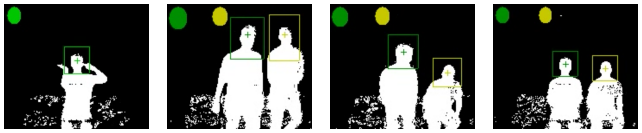


Fig. 6 Head tracking snapshots. Head templates (ellipses), search areas (rectangles) and obtained head locations (crosses).

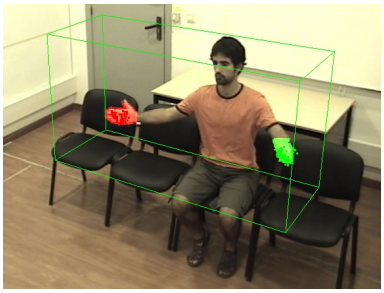


Fig. 7 Caption of the proposed approach, where both hands are detected inside the (green) HandBox.

4.2 Hand Tracking

Hand detection The hand tracking system relies on the robust head estimation presented in Section 4.1. Hands are supposed to be *active* in a space placed in front of the body. We define a HandBox Ω as a virtual 3D box, which is attached to the head position $\hat{p}_H$ so that it follows the user's head at every time instant, as shown in Figure 7.

Dense clusters are searched in Ω by means of a *kd-tree* structure, which allows fast neighbor queries in 3D point clouds [8]. A list of candidate clusters is obtained and filtered according to the following criteria:

Merging Two clusters are merged as a single cluster if the Hausdorff distance between them $< \delta_{min}$

Size filtering The resulting merged clusters are filtered by size, keeping the largest ones ($size > s_{min}$) as hand candidates.

Depth filtering Clusters that fulfill the previous criteria are sorted by depth, keeping those closer to the camera (up to two).

Thresholds δ_{min} and s_{min} should be tuned depending on the type of camera and scene. For example, two hands are being detected in Figure 7.

Open/closed hand detection The Interactivity of the viewer with the rendering node may be increased by determining whether the user opens or closes hands.

We propose to compute the area of the detected hands (Section 4.2) and threshold it to quickly decide if it is closed or not. Such strategy assumes that the observed area is reasonably perpendicular to the camera. Therefore, the depth of each pixel may be replaced by the mean depth of the observed region, simplifying the physical area calculation.

The physical area of an open hand perpendicular to the camera is about $70-90\,cm^2$, whilst that of a closed hand is about $20-30\,cm^2$. It seems reasonable to set a threshold of $50\,cm^2$ to determine the hand status.

Dynamic gestures The pan, tilt and zoom angles of the panoramic viewport are controlled through gesturing. Navigation across the panoramic video is performed by a series of gestures consisting of grabbing (close hand), moving (while hand closed) and releasing, as if the screen was a piece of tissue. The pan and tilt angles are calculated depending on the current position of the hand in the HandBox Ω.

For the zooming case, once grabbed (both hands closed), the distance between hands is calculated. The zoom angle is proportional to that distance.

In addition, a family of dynamic gestures based on trajectory is included. In this setup, for example, horizontal movements with a considerable speed (user-defined) are utilized to shift between ROIs in the ROI mode. Temporal segmentation of these gestures is performed using the entry and exit points in the HandBox, and also assuming a low initial speed if the hand was already in the HandBox.

4.3 Gesture Localization

The gesture localization builds upon the method proposed by [15], which is based on class-specific (one-vs-all) random forests using depth data. Random forest localization in depth data renders an efficient localization algorithm that can operate in real-time under realistic conditions. This implies detecting gestures with high accuracy in the presence of clutter and unintentional motion.

The localization problem is challenging due to two main reasons. Firstly, there is a high unbalance between gesture and non-gesture classes, since non-gesture classes

contains the set of other gestures, as well as clutter and unintentional poses. Secondly, without accurate segmentation, it is difficult to accurately model the appearance of clutter using a limited number of training samples. The gesture localization approach used in our system addresses these issues with two main components:

- Clipping : In order to better capture the local appearance of static gestures, we automatically clip the depth data in a local vicinity of the training samples. In this manner, training is less prone to over-fitting problems caused by learning the specific backgrounds of training data.
- Boosted learning : We use a specific learning approach to deal with the high unbalance of training data [15]. We take advantage of the learning strategy on random forests to efficiently mine the most meaningful samples of the negative class.

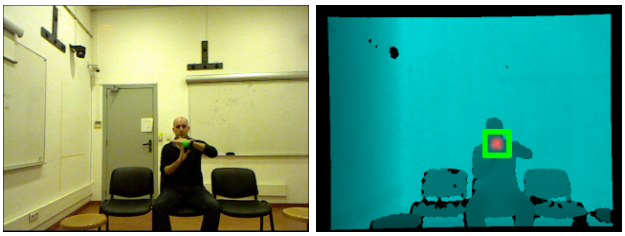
For the sake of completeness, in the following of this section we summarize some details of the approach proposed in [15] (extended version currently submitted to IEEE Trans. MM).

Clipping binary tests Random forests [4] are an ensemble of m randomized trees, which are binary trees. Each tree is trained separately with a small subset of the training data obtained by sampling with replacement. Learning is based on the recursive splitting of training data into 2 subsets, according some binary test f and a threshold θ. The binary test is a function of the feature vector $\mathbf{v}$ obtained from each training example. At each node of the tree, a test f and a threshold θ are randomly generated, and the one that maximizes an information gain criteria is selected.

We define our binary tests based on [26], but we introduce an auxiliary parameter that clips the depth of the available training examples. In such manner, tests are more robust to changes in background, while avoiding a segmentation step. Specifically, the clipping parameter is a value that represents the maximum and minimum relative depth with respect to the depth value of the center pixel. Formally, let κ denote the clipping parameter. Then, for a given pixel $\mathbf{x}$ the test f has the following expression:

$$
\begin{aligned}
&f_\theta(\mathbf{I},\mathbf{x}) = \\
&\max\left(\min\left(d_\mathbf{I}\left(\mathbf{x}+\tfrac{\mathbf{u}}{d_\mathbf{I}(\mathbf{x})}\right), d_\mathbf{I}(\mathbf{x})+\kappa\right), d_\mathbf{I}(\mathbf{x})-\kappa\right) \\
&-\max\left(\min\left(d_\mathbf{I}\left(\mathbf{x}+\tfrac{\mathbf{v}}{d_\mathbf{I}(\mathbf{x})}\right), d_\mathbf{I}(\mathbf{x})+\kappa\right), d_\mathbf{I}(\mathbf{x})-\kappa\right)
\end{aligned}
\tag{3}
$$

where $d_\mathbf{I}$ is the depth map associated to image $\mathbf{I}$ and $\mathbf{u}$ and $\mathbf{v}$ are two randomly generated pixel displacements that fall within a patch size. Pixel displacements are normalized with the depth evaluated at pixel $\mathbf{x}$ in order to make the test features invariant to depth changes.



(a) (b)

Fig. 8 Gesture Localization with Random Forests (best viewed in color). (a) A number of votes (green dots) are casted for the target gesture (b) votes are aggregated to estimate a probability density (overlaid in red on the input depth map) and a localization is estimated (green square).

Boosted learning of random forests In the localization problem posed in this paper, gesture and non-gesture classes are naturally unbalanced. On the one hand, in real applications users are not constantly performing gestures. On the other hand, the actual appearance of a gesture may be represented by a relatively low number of pixels. Summing up, the distribution of gesture classes (*positive*) with respect to non-gesture (*negative*) is biased towards the latter. This unbalance makes that low false positive rates constitute actually a large number of false positive votes. Taking into account this phenomenon is important during the training phase of a random forest since, under such unbalance, it will be difficult to optimize the information gain.

In order to overcome this problem, we adopt a boosted learning scheme, designed in a tree-wise manner as follows:

We train the first tree with a balanced set of samples from each class. Once the tree is trained, we evaluate it against the out-of-bag set [4]. The wrongly classified samples are added to the training set of the second tree (up to a maximum number of training samples). This new training set is completed by sampling with replacement from the full training set until balance is achieved. We train the second tree with this training subset and we repeat the process until the forest is fully trained.

Gesture localization For gesture detection and localization, a set of patches are provided to the detection forest, which casts a vote whenever a *positive* class has more probability than the *negative* class and other *positive* classes. Figure 8 illustrates the casted votes for a positive class in a class-specific learning example. To detect a gesture, we first estimate a probability density

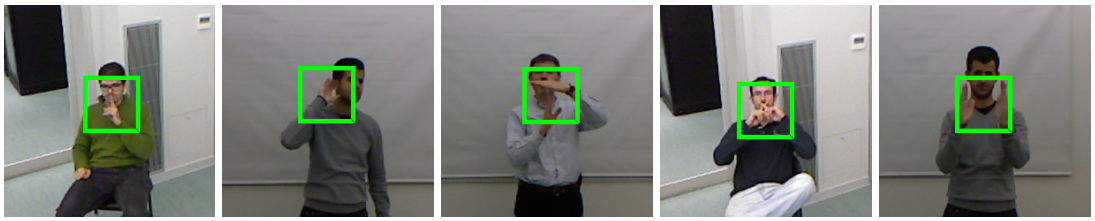


Fig. 9 Examples of successful localization results using our approach ($\kappa = 15$ cm) *Volume Down*, *Volume Up* , *Take Control*, *Mute* and *Pause*.

using the votes within a frame and we take into account temporal consistency by recursively updating this distribution with votes aggregated from past time instants. In order to construct the probability density, we use a Parzen estimator with Gaussian kernel. In order to account for the time component of the approximated density, we sequentially update such density $p(c|\mathbf{I}_t)$ as follows:

$$p'(c|\mathbf{I}_t) = \alpha p(c|\mathbf{I}_t) + (1-\alpha)p'(c|\mathbf{I}_{t-1}) \quad (4)$$

This is a simple yet effective method to keep temporal consistency of the casted votes, as it requires storing a single probability map. An adaptation rate $\alpha = 0.8$ works well in practice, as it prevents several false positives while avoiding a delayed response.

Finally, we compute the pixel location $\mathbf{g}_c$ of a gesture class $c > 0$ as the pixel location with maximum probability. We ensure that such a maximum represents a target gesture by thresholding the probability volume V computed by locally integrating the estimated pseudo-probability measure :

$$V = \sum_{\mathbf{x}\in\mathcal{S}} p'(c|\mathbf{I}_t(\mathbf{x})) \quad (5)$$

where $\mathcal{S}$ is a circular surface element of radius inversely proportional to the depth, and centered at the global maximum. In this way, the localization is depth-invariant.

For efficiency reasons, we approximately segment the scene into relevant and non-relevant pixels by thresholding the depth using two values $z_{\text{near}} = 0.8m$ and $z_{\text{far}} = 3.5m$. Once a user get the control by performing gesture *OK*, we employ the head tracking algorithm result (Section 4.1) to compute a region of interest, where the other 4 gesture class-specific forests are evaluated. In this way, we can run detection forests in real-time. Besides, the region of interest helps in discarding gestures of interest that other users may perform, thus increasing the detection accuracy.

4.4 Face Identification

The purpose of face identification is to allow users to select preferences based on the identity and to enable the system to establish a hierarchy of users that can be used in applications such as parental control.

The continuous monitoring environment offer the possibility of video-based face recognition. This permits improving the performance of face recognition from single image captures.

The workflow of face ID system is the following: faces are detected in first place and forwarded to the face recognition module. The face ID compares each test image with a set of models consisting of images of each person, created off-line with images from previous recordings. For each test face, a set of scores defining the probability it represents the persons in each model is produced. Finally, the fusion module combine the individual results for a temporal group of images of the same person into a final decision.

In the following, we will analyze each module (detection, still image identification and temporal fusion).

Face detection The face detection module is cascaded with the head tracking module, described in Section 4.1. Face detection is only performed inside a reduced region defined by the head tracker. This allows reduced computational complexity and a very low probability of false positives.

We use the OpenCV [3] face detection module that relies on the adaboosted cascade of Haar features, i.e. the Viola-Jones algorithm [31]. In our application, the users generally interact with the system facing the camera. The system is thus restricted to detect frontal faces, allowing for some pose variations.

Single image face recognition For face recognition, a set of meaningful face characteristics are first extracted to form a feature vector. This vector should convey all the relevant information in a face. A feature extraction technique based on Local Binary Pattern (LBP) [17] is used. LBP operator is a non-parametric kernel which represents the local spatial structure of an

image. It provides high discriminative power for texture classification and a good amount of illumination invariance, because it is unaffected by any monotonic gray-scale transformation which preserves the pixel intensity order in a local neighborhood.

At each pixel position, $LBP_{P,R}$ is defined as an ordered set of binary comparisons between the intensity of the center pixel and the intensities between the center pixel and its P surrounding pixels, taken over a circumference of radius R. The decimal form of the resulting LBP code can be expressed as follows:

$$LBP(x_c, y_c) = \sum_{n=1}^{P} s(i_n - i_c)2^n \tag{6}$$

Where i_c corresponds to the grey value of the center pixel (x_c, y_c), and i_n to the grey values at the P surrounding locations. Function s(x) is defined as:

$$s(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{7}$$

Our method is based in [1] For each pixel of the luminance component of the face, its $LBP_{8,2}$ representation is computed, that is, using 8 samples taken over a circumference of radius 2 from the pixel. The resulting transformed image is partitioned into non-overlapping square blocks of 7x7 pixels. The feature vector is formed by concatenation of the histograms of the LBP values of each block. This results in 49 local 64 bins histograms (face image was resized to 49x49). The final feature vector dimensionality is thus 3136.

A k nearest neighbor (kNN) classifier [6] is used to obtain the final decision. In our system kNN is modified so that a vote is penalized by the inverse of the distance between the test vector $\mathbf{t}$ and the selected neighbor, in such a way that the bigger the distance, the more penalization. The k_i inverse distances are normalized by the overall sum of the k nearest neighbor inverse distances of all classes, leading to the a posteriori probability for each class [6].

These probabilities for each class allow the classifier to give an estimation or score on how confident the classification of the test vector is.

The Chi-Square distance (8) is used to compare feature vectors because it is shown that it performs better than the Euclidean one when using histogram based features.

$$\chi^2(\mathbf{v_1}, \mathbf{v_2}) = \sum_i \frac{(v_{1,i} - v_{2,i})^2}{v_{1,i} + v_{2,i}} \tag{8}$$

where $\mathbf{v_1}$ and $\mathbf{v_2}$ are feature vectors.

Models for all individuals in the database should be created off-line, by using sequences of images collected in a different session than the testing recordings. 150 training images per subject are used, extracted automatically from small ad-hoc recordings.

Temporal fusion As face detection is cascaded with head detection and this module performs a temporal tracking of the detected heads, we can ensure that the identity of the person is maintained along the track, so that video based recognition can be used. Temporal fusion combines the information of several images to perform the recognition.

After face detection, each track is composed of T consecutive face images of the same individual. The face identification algorithm is used to compute a vector of scores $\mathbf{s_i}$ for each single frame. A simple score fusion scheme is used, based on averaging the scores along the track. The instantaneous final decision is computed by taking an average of the last N_s score vectors and selecting the class with highest score.

5 Experimental Results and User Evaluation

The evaluation of the proposed system is divided into two different parts. In the first one, an objective evaluation of the different technical components is performed. In this case, all head and hand tracking, gesture localization and face identification are evaluated using objective metrics against other similar methods from the state of the art. In the second part, the entire system is evaluated subjectively by end users with low or no prior knowledge of the system.

For the experimental results, the proposed system has been implemented and a complete gesture control interface for panoramic TV content has been created as a standalone demonstrator. As described, the demonstrator is composed of multiple algorithms that must be connected in different ways; some can run independently in parallel, others need some signaling and synchronization and others must run serially after another one. Therefore, careful signaling must be shared between them. The demonstrator is implemented in the following hardware

- A Microsoft Kinect sensor to capture depth and color video.
- A laptop with 8 CPUs (at least 6 are needed) and 8GB of RAM

It uses the Debian operating system and the following open-source libraries:

- OpenNI as capture drivers for the Kinect sensor
- SmartFlow as a transport and communication middleware

– OpenCV as a support for the face detection algorithm
– PointCloudLibrary (PCL)
– Boost C++ Libraries

All algorithms described in Section 4 (Head and hand tracking, gesture localization, face recognition and application control) are implemented in C++ in an internal library.

5.1 Objective Evaluation of Technical Components

This section evaluates the performance of each individual component presented in the system.

Head Tracking Evaluation We analyze the convenience of conditions C^2 and C^3 in Equation (1), and how robustness is increased by including depth data queues. We compare the proposed scheme with a limited version which only verifies condition C^1. This way, the contribution of C^2 and C^3 is shown. The head location error between a ground-truth (manually marked) head position g^H and the estimated head location is calculated as $\varepsilon = |g^H - \hat{p}^H|$, and presented in Figure 10 for the two versions of the algorithm. Note that, even if C^1 plays the role of 2D method, depth data is used for the initial head size estimation.

Figure 10 shows the frame by frame error of the $C^1 + C^2 + C^3$ compared to C^1. The C^1 version loses target twice, a reset of the algorithm being needed (frames 74 and 398). The labeled arrows in Figure 10 correspond to adverse clutter and occlusion situations. Our proposed algorithm presents an error that does not go above 10 pixels, which is about the head radius.

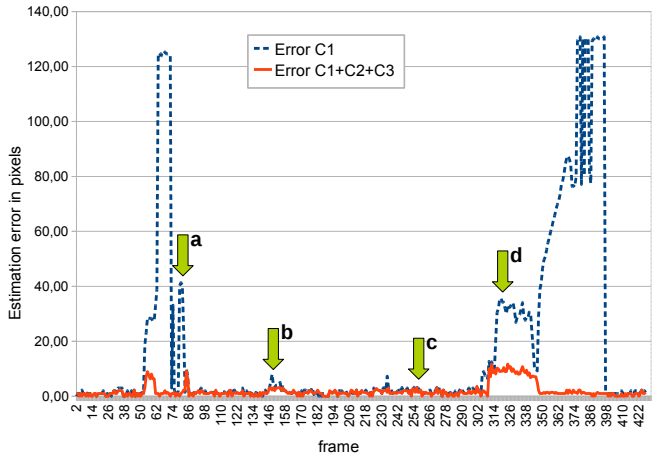


Fig. 10 Error between the obtained head location estimates and the ground-truth.

Table 1 Hand detection 3D accuracy on different gestures

Gesture	# frames	error R hand	error L hand
push	30	2.62 *cm*	-
circle	30	6.61 *cm*	-
replay	35	2.86 *cm*	-
hand up-down	115	5.87 *cm*	-
separate hands	75	2.36 *cm*	3.80 *cm*

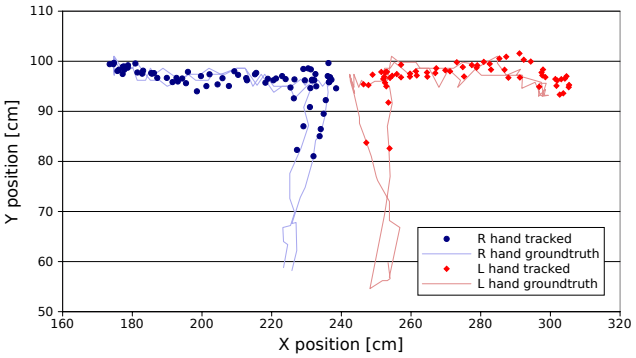


Fig. 11 Ground-truth and estimated trajectories of the (R)ight and (L)eft hands. The estimated hand positions are nicely close to the reference ground-truth positions. Only the XY projection of these 3D trajectories is shown.

Hand Tracking Evaluation Table 1 summarizes the average error for different one-handed and two-handed gestures gestures, computed as the average 3D error (with respect to the ground-truth hand positions) along the duration of the gestures. Ground-truth trajectories have been extracted by hand, selecting a reasonable hand center which may vary in some *cm* along frames. Despite these adverse noisy conditions, the hand estimation error rarely goes above 10 *cm*.

The average error is higher for fast movements, resulting in about 6 *cm* of error. For the other gestures, the error is of about 3 *cm*, which is fairly adequate given the size of a human hand. For the sake of illustration, a sample of the zooming gesture is included in Figure 11, showing the detection error between the ground-truth trajectory and the detected one.

Gesture Localization Evaluation We conduct experiments of the gesture localization method to provide a quantitative performance of the approach and determine optimal clipping parameters. We focus on the gestures defined in Figure 9. We record 5 training sequences where 5 different actors perform a set of gestures and actions, including the 5 target gesture classes. Additionally, we record 6 test sequences with 4 additional actors that were not included in the training set.

The employed detection forests have 15 trees with maximum depth 20, and each tree is trained with approximately 20000 examples per class.

In all cases, we employ squared patches of size 85x85 pixels. We train class-specific forests with the boosted learning method. Additionally, in order to compare the learning method, we implement a boosted approach based on [9].

To measure the accuracy of the proposed methods, we consider a correct localization if the estimated gesture and the actual gesture belong to the same class and if the estimated location is within a radius of 10 pixels. We compute the curves representing 1-precision vs recall to then compute the Area Under the Curve (AUC). Average AUC's per gesture class are shown in Table 2.

The comparison of different training approaches without clipping shows that the proposed boosted learning yields an overall better performance. The proposed learning approach outperforms the boosting scheme proposed in [9].

Face Identification Evaluation Evaluation of face recognition has been done at two levels. In the first place, the single-frame face ID algorithm has been tested. This measures the ability to recognize persons given a unique test frame. Then, we evaluate the performance using the fusion algorithm, where the use of face tracking allows combining consecutive individual results into a more robust decision.

The system is designed to work in small groups of people that share the access to a TV set (for instance, a family). Having this in mind, 12 individuals have been recorded while using the demo in two sessions: the first one is used to create the models while the second one is used for testing. As some gestures involve putting the hands between the face and the camera, there are a good number of partial face occlusions. Pose is mostly frontal but with variations as the users look at the different corners of the screen. There is also a medium degree of expression variability since the users are asked to behave normally while using the system.

Table 3 shows the results either for single frame and for temporal fusion face ID. Due to occlusions and the restrictive settings, there are good number of frames where no faces are detected. This is not a problem in this setup as a decision is only provided at regular intervals (currently, each one or two seconds). It can be seen that even using single frame, the identification results are very good, with only 37 errors in 6605 detected faces. When using the temporal fusion, the identity of the person in each segment is always given correctly.

Table 3 Face ID results

	# frames	detected	error
Single frame	8716 frames	6605 frames	0.56%
Temporal fusion	-	266 segments	0.0%

5.2 Usability Evaluation

In order to validate the design of the interface we conducted a usability study. We selected a set of participants, who were asked to interact with the system and then they answered a questionnaire. This usability study is derived from similar studies that have been recently conducted to evaluate a gesture controlled interface for elderly in [2].

The procedure was the following:

1. First, users watched a short video about how to use the interface [1].
2. Then, they interacted freely with the system to familiarize with its functionalities.
3. They were asked to perform 3 specific conceptual tasks: To change the volume (up, down and mute); To zoom into a defined part of the scene; To select certain ROI.
4. Finally, they answered the questionnaire.

In total, the number of participants were 15. The questions had 5 answer choices according to the degree of agreement with the sentence (*Strongly Disagree*=0,. . ., *Strongly Agree*=4). The sentences and the mean and standard deviation results are shown in Table 4.

In general, users were satisfied with the system and they could perform effectively the assigned tasks in a short amount of time. The most successful aspects of the user experience are related to the use of static gestures to activate several control functions. As reported by participants, the system is simple to use, easy to learn and the GUI is clear. From their response and according to our observation of the participants, we can conclude that the selected gestures are intuitive and easy to perform. Moreover, the use of static gestures as *shortcuts* to perform actions reduce the amount of information required in the GUI, improving its clarity. Navigation through panoramic video was not so successful. The reason seems to be due to the fact that the interaction paradigm chosen for zooming and panning (i.e hand gestures with hands open or closed to simulate touching a virtual tablet in front of the user) results not being comfortable and intuitive by participants. Further research is required to improve the usability of the system and such study offers insights into a better user experience.

[1] The video is available at http://vimeo.com/55432492

Table 2 Average Area Under the Curve (AUC) for different learning approaches.

Method	Volume Down	Volume Up	OK	Mute	Pause	Average
Boosted [9]	0.17	0.10	0.31	0.14	0.02	0.15
Ours	0.17	0.10	0.31	0.20	**0.25**	0.21
Ours + κ = 15cm	0.58	**0.53**	**0.81**	**0.47**	0.11	**0.5**
Ours + κ = 30cm	**0.69**	0.50	0.54	0.27	0.14	0.42
Ours + κ = 50cm	0.59	0.46	0.61	0.24	0.18	0.41

Table 4 Usability Questionnaire. Sentences and Mean and Standard deviation of the responses of 15 participants. (*Strongly Disagree*=1,..., *Strongly Agree*=5).

Questions	Mean	Standard Deviation
1. I liked using the interface	4.13	0.96
2. The interface was pleasant to use	3.81	0.98
3. It was simple to use this system	4.00	0.73
4. It was easy to learn to use the system	4.31	0.79
5. The organization of information presented by the system was clear	4.25	0.68
6. The information provided by the system was easy to understand	4.50	0.52
7. Navigation in panoramic video was pleasant to use	3.47	0.99
8. Navigation through interface menu items was comfortable	3.94	1.00
9. The system responded effectively to my gestures	3.19	0.98
10. Task 1 (Zoom into a specific part of the scene) was easy to perform	3.88	1.31
11. Task 2 (Change audio volume) was easy to perform	4.13	1.36
12. Task 3 (Region of Interest Selection) was easy to perform	4.00	0.89
13. I felt comfortable using the system	3.63	0.89
14. Overall, I am satisfied with this system	3.69	0.87

6 Conclusion and Future Work

This paper have presented a gesture recognition system to control a TV set. It works in a device-less and marker-less manner allowing users to control the content on their TV sets only using their hands. It automatically locates the people that want to interact with the system and understands and recognizes their gestures. The objective evaluation of the technologies integrated in the system show comparable results to similar techniques in the state of the art. The complete integrated system has been evaluated subjectively by 15 users which where satisfied by the simplicity and intuitiveness of the interface.

Our future work include the extension of the gesture localization part to successfully detect and recognize fingers exploring ways to include finger gestures within the pre-defined gesture database. Performing further user evaluation studies, we plan to improve the usability of the navigation through the panoramic video in order to improve the responsiveness of the system and to limit the fatigue caused to the users.

References

1. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**(12), 2037 –2041 (2006). DOI 10.1109/TPAMI.2006.244
2. Bhuiyan, M., Picking, R.: A gesture controlled user interface for inclusive design and evaluative study of its usability. Journal of Software Engineering and Applications **4**(9), 513–521 (2011)
3. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Demirdjian, D., Varri, C.: Recognizing events with temporal random forests. In: Proceedings of the 2009 international conference on Multimodal interfaces, ICMI-MLMI '09, pp. 293–296. ACM, New York, NY, USA (2009). DOI 10.1145/1647314.1647377
6. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2. edn. Wiley, New York (2001)
7. Francese, R., Passero, I., Tortora, G.: Wiimote and kinect: gestural user interfaces add a natural third dimension to hci. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12, pp. 116–123. ACM, New York, NY, USA (2012). DOI 10.1145/2254556.2254580
8. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An Algorithm for Finding Best Matches in Logarithmic Expected Time. Trans. on Mathematic Software **3**(3), 209–226 (1977). DOI 10.1145/355744.355745
9. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. TPAMI **33**(11), 2188 –2202 (2011). DOI 10.1109/TPAMI.2011.70
10. Gesturepak: Gesture recording and recognition toolkit. http://www.franklins.net/gesturepak.aspx. Accessed: 20/02/2013

11. Jaimes, A., Sebe, N.: Multimodal humancomputer interaction: A survey. Computer Vision and Image Understanding **108**(1-2), 116 – 134 (2007). DOI 10.1016/j.cviu.2006.10.019. ¡ce:title¿Special Issue on Vision for Human-Computer Interaction¡/ce:title¿
12. Ji, Q., Wechsler, H., Duchowski, A., Flickner, M.: Editorial: Special issue: eye detection and tracking. Comput. Vis. Image Underst. **98**(1), 1–3 (2005). DOI 10.1016/j.cviu.2004.07.006
13. Kinect for windows sdk. http://www.microsoft.com/en-us/kinectforwindows/develop/. Accessed: 20/02/2013
14. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing **5**(6), 657 – 675 (2009). DOI 10.1016/j.pmcj.2009.07.007. ¡ce:title¿PerCom 2009¡/ce:title¿
15. López-Méndez, A., Casas, J.R.: Can our tv robustly understand human gestures?: real-time gesture localization in range data. In: Proceedings of the 9th European Conference on Visual Media Production, CVMP '12, pp. 18–25. ACM, New York, NY, USA (2012). DOI 10.1145/2414688.2414691
16. Nielsen, M., Strring, M., Moeslund, T., Granum, E.: A procedure for developing intuitive and ergonomic gesture interfaces for hci. In: A. Camurri, G. Volpe (eds.) Gesture-Based Communication in Human-Computer Interaction, *Lecture Notes in Computer Science*, vol. 2915, pp. 409–420. Springer Berlin Heidelberg (2004)
17. Ojala, T., Pietikinen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern Recognition pp. 51–59 (1996)
18. Openni sdk. http://www.openni.org/openni-sdk/. Accessed: 20/02/2013
19. Pantic, M., Rothkrantz, L.J.M.: Automatic analysis of facial expressions: The state of the art. IEEE Trans. Pattern Anal. Mach. Intell. **22**(12), 1424–1445 (2000). DOI 10.1109/34.895976
20. Poppe, R.: Vision-based human motion analysis: An overview. Computer Vision and Image Understanding **108**(12), 4 – 18 (2007). DOI 10.1016/j.cviu.2006.10.016. ¡ce:title¿Special Issue on Vision for Human-Computer Interaction¡/ce:title¿
21. Potamianos, G., Neti, C., Luettin, J., Matthews, I.: Audio-visual automatic speech recognition: An overview. Issues in Visual and Audio-Visual Speech Processing pp. 356–396 (2004)
22. Pugeault, N., Bowden, R.: Spelling It Out: Real-Time ASL Fingerspelling Recognition. In: ICCV-CDC4CV (2011)
23. Ren, Z., Yuan, J., Zhang, Z.: Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In: ACM MM, MM '11, pp. 1093–1096. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/2072298.2071946
24. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a wii controller. In: Proceedings of the 2nd international conference on Tangible and embedded interaction, TEI '08, pp. 11–14. ACM, New York, NY, USA (2008). DOI 10.1145/1347390.1347395
25. Sebe, N.: Multimodal interfaces: Challenges and perspectives. J. Ambient Intell. Smart Environ. **1**(1), 23–30 (2009)
26. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: CVPR, pp. 1297 –1304 (2011). DOI 10.1109/CVPR.2011.5995316
27. STERN, H.I., WACHS, J.P., EDAN, Y.: Designing hand gesture vocabularies for natural interaction by combining psycho-physiological and recognition factors. International Journal of Semantic Computing **02**(01), 137–160 (2008). DOI 10.1142/S1793351X08000385
28. Suau, X., Ruiz-Hidalgo, J., Casas, J.R.: Real-Time Head and Hand Tracking based on 2.5D data. Transactions on Multimedia **1**(99), 1 (2012)
29. Turk, M.: Gesture recognition. Handbook of Virtual Environment Technology (2001)
30. Uebersax, D., Gall, J., Van den Bergh, M., Van Gool, L.: Real-time Sign Language Letter and Word Recognition from Depth Data. In: ICCV-HCI, pp. 1–8 (2011)
31. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision **57**(2), 137–154 (2004)
32. Wachs, J.P., Kölsch, M., Stern, H., Edan, Y.: Vision-based hand-gesture applications. Commun. ACM **54**(2), 60–71 (2011)
33. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Comput. Surv. **35**(4), 399–458 (2003). DOI 10.1145/954339.954342
34. Zigfu. motion controlled web. http://zigfu.com. Accessed: 20/02/2013