A Biological and Real-Time Framework for Hand Gestures and Head Poses

Mario Saleiro, Miguel Farrajota, Kasim Terzić, João M.F. Rodrigues, and J.M. Hans du Buf

Vision Laboratory, LARSyS, University of the Algarve, 8005-139 Faro, Portugal
{masaleiro,mafarrajota,kterzic,jrodrig,dubuf}@ualg.pt
http://w3.ualg.pt/~dubuf/vision.html

Abstract. Human-robot interaction is an interdisciplinary research area that aims at the development of social robots. Since social robots are expected to interact with humans and understand their behavior through gestures and body movements, cognitive psychology and robot technology must be integrated. In this paper we present a biological and real-time framework for detecting and tracking hands and heads. This framework is based on keypoints extracted by means of cortical V1 end-stopped cells. Detected keypoints and the cells' responses are used to classify the junction type. Through the combination of annotated keypoints in a hierarchical, multi-scale tree structure, moving and deformable hands can be segregated and tracked over time. By using hand templates with lines and edges at only a few scales, a hand's gestures can be recognized. Head tracking and pose detection are also implemented, which can be integrated with detection of facial expressions in the future. Through the combinations of head poses and hand gestures a large number of commands can be given to a robot.

Keywords: Hand gestures, Head pose, biological framework.

1 Introduction

With the advent of newer and more complex technologies has come an increasing effort to make them easy to use. Some years ago computers were only used by specialized technicians, but nowadays even young children and elderly can use complex technology with great ease. The way how we use computers, cell phones and other devices has drastically changed because we began to research and implement natural ways of interacting with them. Part of that research effort consists of the analysis of humans and their actions such that machines and software may be designed to match our natural behaviors. One of the areas of interest for such interpretation is the recognition of human gestures, as they are used as a natural, intuitive and convenient way of communication in our daily life. The recognition of hand gestures can be widely applied in human-computer interfaces and interaction, games, human-robot interaction, augmented reality, etc.

Gesture analysis and recognition has been a popular research field for some years and numerous approaches have been developed. Interest in this area has

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2013, Part I, LNCS 8009, pp. 556–565, 2013.

spiked since the advent of low-cost and very reliable depth-based sensors like the Kinect [12,15]. Although many gesture-based interfaces have been developed, to the best of our knowledge none of them is biologically inspired. Most of them are based on traditional methods from computer vision.

A method for hand tracking and motion detection using a sequence of stereo color frames was proposed by Kim et al. [6]. Another approach, which consists of the recognition of gestures by tracking the trajectories of different body parts, was developed by Bandera et al. [1]. In this method, trajectories are described by a set of keypoints and gestures are characterized through global properties of those trajectories. Suk et al. [13] devised a method for recognizing hand gestures in continuous video streams by using a dynamic Bayesian network. Suau et al. [12] presented a method to perform hand and head tracking using the Kinect. One- and two-handed gestures are also recognized by analysing the trajectories of both hands. Also using the Kinect, Li [15] presented a method that is able to recognize nine different gestures and identify fingers with high accuracy.

Although some methods do work fairly well for a specific purpose, they may not be suitable for a more profound analysis of human behavior and gestures because these are very complex. In this paper we complement a biological and real-time framework for detecting and tracking hands [4] with head movements. This framework is based on multi-scale keypoints detected by means of models of cortical end-stopped cells [7,9]. Cell responses around keypoints are used to classify the vertex type, for creating annotated keypoints [3]. The model has been extended by multi-scale line and edge information, also extracted by models of cortical cells. We also developed a model for optical flow based on annotated keypoints [4]. By integrating optical flow and annotated keypoints in a hierarchical, multi-scale tree structure, deformable and moving objects can be segregated and tracked over time.

Hand and gesture recognition is obtained by using a simple line and edge template matching algorithm which relates previously stored templates with the acquired images across two scales. By using only five hand templates with lines and edges obtained at two different scales, a hand's gestures can be recognized. By tracking hands over time, false positives due to complex background patterns can be avoided. We also focus on head movements because they too can be an important part of human-robot interaction. When combined with the recognition of facial expressions (ongoing work) it will provide invaluable information for natural human-computer and human-robot interaction. Our framework addresses the most common movements: leaning left/right and nodding (up/down) (shaking left/right is ongoing work). These can be used to give feedback to a robot, expressing doubts or affirming or criticizing actions, respectively. By combining a few head movements with hand gestures, a large number of instructions can be given.

The developed system does not require any prior calibration. Since the cell models have been optimized for running on a GPU, a speed of about 10 frames per second can be obtained, which is fast enough for real-time applications.

2 Multi-scale Lines, Edges and Keypoints

In cortical area V1 we find simple, complex and end-stopped cells [9], which are thought to play an important role in coding the visual input: to extract multi-scale lines and edges and keypoint information (keypoints are line/edge vertices or junctions, but also blobs).

Responses of even and odd simple cells, corresponding to the real and imaginary parts of a Gabor filter [9], are denoted by $R^E_{s,i}(x,y)$ and $R^O_{s,i}(x,y)$, i being the orientation (we use $N_\theta = 8$). The scale s is given by λ, the wavelength of the Gabor filters, in pixels. We use $4 \leq \lambda \leq 20$ with $\Delta\lambda = 4$. Responses of complex cells are modeled by the modulus $C_{s,i}(x,y) = [\{R^E_{s,i}(x,y)\}^2 + \{R^O_{s,i}(x,y)\}^2]^{1/2}$.

The basic scheme for line and edge detection is based on responses of simple cells: a positive or negative line is detected where R^E shows a local maximum or minimum, respectively, and R^O shows a zero crossing. In the case of edges the even and odd responses are swapped. This gives four possibilities for positive and negative events. An improved scheme [9] consists of combining responses of simple and complex cells, i.e., simple cells serve to detect positions and event types, whereas complex cells are used to increase the confidence. Lateral and cross-orientation inhibition are used to suppress spurious cell responses beyond line and edge terminations, and assemblies of grouping cells serve to improve event continuity in the case of curved events. We denote the line and edge map by $LE_s(x,y)$.

Keypoints are based on cortical end-stopped cells [7]. They provide important information because they code local image complexity. Furthermore, since keypoints are caused by line and edge junctions, detected keypoints can be classified by the underlying vertex structure, such as K, L, T, + etc. This is very useful for most matching problems: object recognition, optical flow and stereo disparity. In this section we briefly describe the multi-scale keypoint detection and annotation processes. The original model has been improved such that multi-scale keypoints can be detected in real time [14].

There are two types of end-stopped cells, single and double. These are applied to $C_{s,i}$ and are combined with tangential and radial inhibition schemes in order to obtain precise keypoint maps $K_s(x,y)$. For a detailed explanation with illustrations see [7] and [14].

In order to classify any detected keypoint, the responses of simple cells $R^E_{s,i}$ and $R^O_{s,i}$ are analyzed, but now using $N_\phi = 2N_\theta$ orientations, with $\phi_k = k\pi/N_\theta$ and $k = [0, N_\phi - 1]$. This means that for each of the 8 simple-cell orientations on $[0,\pi]$ there are two opposite analysis orientations on $[0,2\pi]$, e.g., $\theta_1 = \pi/N_\theta$ results in $\phi_1 = \pi/N_\theta$ and $\phi_9 = 9\pi/N_\theta$. This division into response-analysis orientations is acceptable according to [5], because a typical cell has a maximum response at some orientation and its response decreases on both sides, from 10 to 20 degrees, after which it declines steeply to zero; see also [2].

Classifying keypoints is not a trivial task, mainly because responses of simple and complex cells, which code the underlying lines and edges at vertices, are unreliable due to response interference effects [2]. This implies that responses must be analyzed in a neighborhood around each keypoint, and the size of the neighborhood must be proportional to the scale of the cells. The validation of the line and

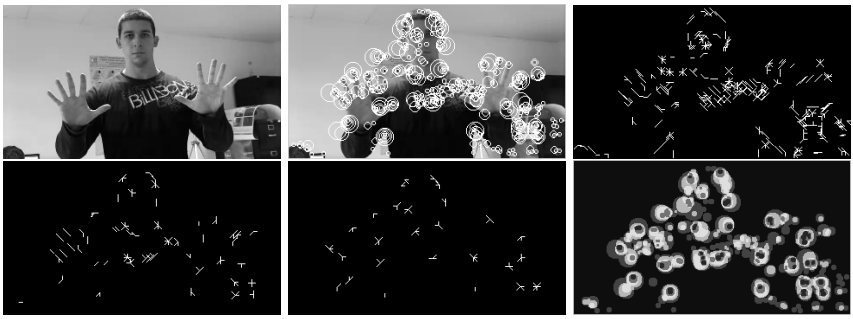

Fig. 1. Left to right and top to bottom: input frame, keypoints detected at all 5 scales, annotated keypoints at scales $\lambda = 4$, 8 and 12, and the frame's saliency map where red indicates higher and blue lower saliency

edge orientations which contribute to the vertex structure is based on an analysis of the responses of complex cells $C_{s,i}(x, y)$. At a distance of λ, and for each direction ϕ_k, responses in that direction and in neighboring orientations ϕ_{k+l}, with $l = \{-2, -1, 0, 1, 2\}$, are summed with different weights equal to $1/2^{|l|}$. After this smoothing and detection of local maxima, each keypoint is then annotated by a descriptor of 16 bits which codes the detected orientations. In the case of keypoints caused by blobs with no underlying line and edge structures, all 16 bits are zero.

This method is an improvement of the previous method [3]. It provides a more detailed descriptor of the underlying line and edge structures, with a significant increase in performance and with a negligible loss of precision. The first five images in Fig. 1 illustrate keypoint detection and annotation at the given scales. For more illustrations see [7].

3 Optical Flow

Keypoint detection may occur in cortical areas V1 and V2, whereas keypoint annotation requires bigger receptive fields and could occur in V4. Optical flow is then processed in areas V5/MT and MST, which are related to object and ego motion for controlling eye and head movements.

Optical flow is determined by matching annotated keypoints in successive camera frames, but only by matching keypoints which may belong to a same object. To this purpose we use regions defined by saliency maps. Such maps are created by summing detected keypoints over all scales s, such that keypoints which are stable over scale intervals yield high peaks. In order to connect the individual peaks and yield larger regions, relaxation areas proportional to the filter scales are applied [7]. Here we simplify the computation of saliency maps by simply summing the responses of end-stopped cells at all scales, which is much faster and yields similar results. Figure 1 (bottom-right) illustrates a saliency map.

We apply a multi-scale tree structure in which at a very coarse scale a root keypoint defines a single object, and at progressively finer scales more keypoints

are found which convey the object's details. For optical flow we use five scales: $\lambda = [4, 20]$ with $\Delta\lambda = 4$. All keypoints at $\lambda = 20$ are supposed to represent individual objects, although we know that it is possible that several of those keypoints may belong to a same object. Each keypoint at a coarse scale is related to one or more keypoints at one finer scale, which can be slightly displaced. This relation is modeled by down-projection using grouping cells with a circular axonic field, the size of which (λ) defines the region of influence, and this process continues until the finest scale is reached; see [3].

As mentioned above, at a very coarse scale each keypoint – or central keypoint CKP – should correspond to an individual object [7]. However, at the coarsest scale applied here, $\lambda = 20$, this may not be the case and an object may cause several keypoints. In order to determine which keypoints could belong to the same object we combine saliency maps with the multi-scale tree structure.

At this point we have, for each frame, the tree structure which links the keypoints over scales, from coarse to fine, with associated regions of influence at the finest scale. We also have the saliency map obtained by summing responses of end-stopped cells over all scales. The latter, after thresholding, yields segregated regions which are intersected with the regions of influence of the tree. Therefore, the intersected regions link keypoints at the finest scale to the segregated regions which are supposed to represent individual objects.

Now, each annotated keypoint of frame i can be compared with all annotated keypoints in frame $i-1$. This is done at all scales, but the comparison is restricted to an area with radius 2λ instead of λ at each scale in order to allow for larger translations and rotations. In addition, (1) at fine scales many keypoints outside the area can be skipped since they are not likely to match over large distances, and (2) at coarse scales there are less keypoints, λ is bigger, and therefore larger distances (motions) are represented there. The matching process, as for building the tree, is now done top-down. Previously it was done bottom-up [3]. Due to the use of a more detailed descriptor for keypoint classification than in [3], matching keypoints at the coarsest scale provides sufficient accuracy to correctly match entire tree structures. An additional gain in performance is due to the reduced number of comparisons at finer scales, because of existing dependencies between keypoints in the branches of the tree structure. Keypoints are matched by combining three similarity criteria with different weight factors:

(a) The distance D serves to emphasize keypoints which are closer to the center of the matching area. For having $D = 1$ at the center and $D = 0$ at radius 2λ, we use $D = (2\lambda - d)/2\lambda$ with d the Euclidean distance (this can be replaced by dynamic feature routing [3,8]).

(b) The orientation error O measures the correlation of the attributed orientations, but with an angular relaxation interval of $\pm 2\pi/N_\theta$ applied to all orientations such that also a rotation of the vertex structure is allowed. Similar to D, the summed differences are combined such that $O = 1$ indicates good correspondence and $O = 0$ a lack of correspondence. Obviously, keypoints marked "blob" do not have orientations and are treated separately.

Fig. 2. The optical flow model applied to a person while performing several hand and head gestures. Hands and head are marked by their bounding boxes. The bottom-right image shows the combined centers of the boxes.

(c) The tree correspondence C measures the number of matched keypoints at finer scales, i.e., at any scale coarser than the finest one. The keypoint candidates to be matched in frame i and in the area with radius 2λ are linked in the tree to localized sets of keypoints at all finer scales. The number of linked keypoints which have been matched is divided by the total number of linked keypoints. This is achieved by sets of grouping cells at all but the finest scale which sum the number of linked keypoints in the tree, both matched and all; for more details see [3].

The three parameters are combined by grouping cells which can establish a link between keypoints in frame $i-1$ and i. Mathematically we use the similarity measure $S = \alpha O + \beta C + \gamma D$, with $\alpha = 0.4$ and $\beta = \gamma = 0.3$. These values were determined empirically. The candidate keypoint with the highest value of S in the area (radius 2λ) is selected and the vector between the keypoint in frame $i-1$ and the matched one in frame i is computed. Remaining candidates in the area can be matched to other keypoints in frame i, provided they are in their local areas. Keypoints which cannot be matched are discarded. Figure 2 shows a sequence with tracked hands by using optical flow.

4 Hand/Head Tracking and Gesture/Pose Recognition

To initialize the tracking and recognition process, it is only required that at the beginning the user stands still, looking straight ahead and showing the palms of both hands to the camera. As the first step in the processes that will be described in this section, we use skin color segmentation to detect both hands and the head as previously applied in [10]. If I is an input frame, we can use the following expression to get a binary skin image, I_s, where skin is marked in black

and all the rest is white: $I_s(x, y) = 0$ if $\varphi[I(x, y)] = 1$, otherwise $I_s(x, y) = 255$, where $\varphi = [(R > 95) \wedge (G > 40) \wedge (B > 20) \wedge ((\max\{R, G, B\} - min\{R, G, B\}) > 15) \wedge (|R - G| > 15) \wedge (R > G) \wedge (R > B)]$, with $(R, G, B) \in [0, 255]$.

After obtaining the skin regions we can obtain three regions: left hand, right hand and head. Then we apply two filters: the first one is an erosion which removes small regions, and the second one is a dilation which makes the remaining regions more homogeneous. After this we apply a fast blob detection algorithm [10] to obtain the coordinates and sizes of the three biggest skin regions. The region with the highest y coordinate will be considered as being the head. The system will use the head blob's dimensions to calculate the reference ratio $R_r = h/w$, with h the height and w the width, as a reference for the neutral pose. The detection of head poses is done like previously in [10]. We use five head poses: face straight forward, head up, head down, head leaning to the left and head leaning to the right. To detect the up and down poses we use a very simple method which consists of comparing the blob's actual ratio R_a, to an upper (U_{thr}) and a lower threshold (L_{thr}). The latter are determined from the reference ratio R_r computed during the initialization: $U_{\text{thr}} = 1.1 \times R_r$ and $L_{\text{thr}} = 0.9 \times R_r$.

To detect the left- and right-leaning poses, two vertical lines, at distances $w/6$ and $5w/6$ inside the blob's box are considered. The average position of black pixels on each of these lines is calculated and the two resulting positions are used to detect the two poses: when the user leans the head to one side, the average positions will go up and down relative to the middle of the box ($h/2$). A minimum vertical distance MVD of $0.2 \times h$ between both positions was determined experimentally, such that small lateral movements can be ignored. The two poses are detected when (a) the vertical distance between the two positions is larger than the MVD, and (b) one of the positions is higher than $h/2$. The latter position determines the side of the movement: left or right.

While the head will normally be at a static location, hands may be constantly moving and therefore they must be tracked. To do that we employ the optical flow as explained in the previous section. The recognition of hand gestures is more complex than the detection of the head's poses. To recognize hand gestures, we need to use a single template, at a few different scales, of each gesture. The templates are previously prepared so that they are available for online matching when the system is working. To prepare the templates, we apply the previously described line- and edge-extraction algorithm at two different scales, and then dilate the resulting maps to make the templates more robust against small differences between them and the real frames containing moving hands. Each template is a binary image which contains white lines against a black background. Example templates are shown in Fig. 3.

To perform the template matching in a fast way, we only compare the templates with the regions tracked by optical flow. This way no processing time is wasted in other image regions. The matching process is done in two steps: (a) direct template matching and (b) template density matching. In step (a) we take a tracked region and apply the same process that was used to prepare

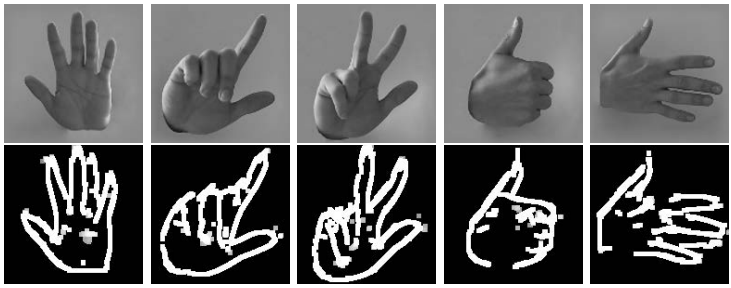

Fig. 3. Top: five hand gestures. Bottom: their dilated templates at scale $\lambda = 8$.

the templates. Then we shift each template over the tracked region and at each shift position we compare them, pixel by pixel, and count the number of white pixels, P_w. Basically this is a 2D correlation process. We divide P_w by the total number of white pixels in the template, P_{wt}, and store the resulting value in a probability map at the center position of the (shifted) template. The result is a 2D histogram or correlation matrix in which higher values indicate a better correspondence between the tracked region and the template.

In step (b), template density matching, we verify whether the test region has the same ratio of white and black pixels as the template. This must be done because if only direct matching was used, some complex textures, for example on a (moving) T-shirt or (static) background can result in false detections of some templates. Again we use the shifting window with the same size of the template and, for each shift position, we calculate the ratio between the number of white pixels and the total number of pixels in the window, $R = W_p/T_p$, where W_p is the number of white pixels and T_p the total number of pixels. Like before, this ratio is stored in a similar probability map.

After these two steps we combine both maps for each template, giving a 70% weight to the first map and 30% to the second one. This yields a single probability map for each template. This process is applied at the two scales used ($\lambda = \{8, 12\}$), and the two probability maps are mixed prior to multi-scale recognition, thereby giving equal weights to the two scales. At this point we have a single but multi-scale probability map for each template. Every time that a value greater than a threshold value occurs ($T = 60$ was experimentally determined), the system considers that the gesture which corresponds to that map has been recognized, and at the peak location. When more than one gesture is recognized, only the one with the greatest probability value will prevail. Figure 4 illustrates the matching process (top) together with the detection of head poses (bottom).

5 Discussion

In this paper we presented a biologically inspired method for hand detection, tracking, and gesture recognition. By using optimized algorithms for the detection of keypoints plus lines and edges, and by selecting only a few scales, the

Fig. 4. The top row illustrates hand gesture recognition: input image, thresholded probability maps of the two detected gestures with their peaks in white, and the final result. The bottom row shows examples of the recognition of four head poses: right, left, up and down. Along the vertical edges of the bounding box, the two green points show the average location of skin-colored pixels.

method can run in real time. Even when using a cheap HD webcam, very good results can be obtained. This we expected due to our previous experience with cortical models: multi-scale keypoints, lines and edges provide very useful information to generate saliency maps for Focus of Attention or to detect faces by grouping facial landmarks defined by keypoints at eyes, nose and mouth [7]. In [11] we were able to use lines and edges to recognize facial expressions with success, and in [9] we have shown that lines and edges are very useful for face and object recognition. The method included here for the detection of head poses is not biological, but in principle we can integrate our methods for face detection and recognition of facial expressions.

Biologically inspired methods involve many filter kernels, here in eight orientations and at several scales. In order to achieve real-time processing, we only use five scales for optical flow and region segregation. For gesture recognition we use lines and edges at only two scales. The system's main limitation is the costly filtering. The optimized GPU implementation allows us to process at least 10 frames/s with a maximum resolution of 600×400 pixels and using at least 6 scales if coarser scales are used. The main bottleneck for using large images and fine scales is the 1 GByte of memory of the GPU, because of the Gaussian pyramid employed in the filtering.

Future work will focus on motion prediction, a process that occurs in cortical area MST. We also intend to increase the precision such that individual fingers can be detected in combination with a larger number of gestures. The ultimate goal is to apply 3D processing in the entire process, with emphasis on body language. This can be done by using cheap off-the-shelf solutions like a Kinect or two webcams with a biological disparity model.

Acknowledgements. This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) project PEst-OE/EEI/LA0009/2011; EU project NeuralDynamics FP7-ICT-2009-6 PN: 270247; FCT project Blavigator RIPD/ADA/109690/2009, and by FCT PhD grants to MS (SFRH/BD/71831/2010) and MF (SFRH/BD/79812/2011).

References

1. Bandera, J.P., Marfil, R., Bandera, A., Rodríguez, J.A., Molina-Tanco, L., Sandoval, F.: Fast gesture recognition based on a two-level representation. Pattern Recogn. Lett. 30(13), 1181–1189 (2009)
2. du Buf, J.M.H.: Responses of simple cells: events, interferences, and ambiguities. Biol. Cybern. 68, 321–333 (1993)
3. Farrajota, M., Rodrigues, J.M.F., du Buf, J.M.H.: Optical flow by multi-scale annotated keypoints: a biological approach. In: Proc. Int. Conf. on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2011), Rome, Italy, pp. 307–315 (2011)
4. Farrajota, M., Saleiro, M., Terzić, K., Rodrigues, J.M.F., du Buf, J.M.H.: Multi-scale cortical keypoints for realtime hand tracking and gesture recognition. In: Proc. 1st Int. Workshop on Cognitive Assistive Systems: Closing the Action-Perception Loop, pp. 9–15 (2012)
5. Hubel, D.H.: Eye, Brain and Vision. Scientific American Library (1995)
6. Kim, H., Kurillo, G., Bajcsy, R.: Hand tracking and motion detection from the sequence of stereo color image frames. In: Proc. IEEE Int. Conf. on Industrial Technology, pp. 1–6 (2008)
7. Rodrigues, J., du Buf, J.M.H.: Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection. BioSystems 2, 75–90 (2006)
8. Rodrigues, J., du Buf, J.M.H.: A cortical framework for invariant object categorization and recognition. Cognitive Processing 10(3), 243–261 (2009)
9. Rodrigues, J., du Buf, J.M.H.: Multi-scale lines and edges in V1 and beyond: brightness, object categorization and recognition, and consciousness. BioSystems 95, 206–226 (2009)
10. Saleiro, M., Rodrigues, J., du Buf, J.M.H.: Automatic hand or head gesture interface for individuals with motor impairments, senior citizens and young children. In: Proc. Int. Conf. Softw. Dev. for Enhancing Accessibility and Fighting Info-Exclusion, pp. 165–171 (2009)
11. de Sousa, R.J.R., Rodrigues, J.M.F., du Buf, J.M.H.: Recognition of facial expressions by cortical multi-scale line and edge coding. In: Campilho, A., Kamel, M. (eds.) ICIAR 2010. LNCS, vol. 6111, pp. 415–424. Springer, Heidelberg (2010)
12. Suau, X., Ruiz-Hidalgo, J., Casas, J.R.: Real-time head and hand tracking based on 2.5d data. IEEE Trans. on Multimedia 14(3), 575–585 (2012)
13. Suk, H., Sin, B., Lee, S.: Hand gesture recognition based on dynamic Bayesian network framework. Pattern Recogn 43(9), 3059–3072 (2010)
14. Terzić, K., du Buf, J.M.H., Rodrigues, J.M.F.: Real-time object recognition based on cortical multi-scale keypoints. In: Accepted for 6th Iberian Conference on Pattern Recognition and Image Analysis, Madeira, Portugal, June 5-7 (2013)
15. Yi, L.: Hand gesture recognition using kinect. In: Proc. IEEE 3rd Int. Conf. on Softw. Engin. and Service Science, pp. 196–199 (2012)