Biologically inspired vision for human-robot interaction

M. Saleiro, M. Farrajota, K. Terzić, S. Krishna, J.M.F. Rodrigues, and J.M.H. du Buf

Vision Laboratory, LARSyS, University of the Algarve, 8005-139 Faro, Portugal,
{masaleiro, mafarrajota, kterzic, jrodrig, dubuf}@ualg.pt,
saikrishnap2003@gmail.com,

Abstract Human-robot interaction is an interdisciplinary research area that is becoming more and more relevant as robots start to enter our homes, workplaces, schools, etc. In order to navigate safely among us, robots must be able to understand human behavior, to communicate, and to interpret instructions from humans, either by recognizing their speech or by understanding their body movements and gestures. We present a biologically inspired vision system for human-robot interaction which integrates several components: visual saliency, stereo vision, face and hand detection and gesture recognition. Visual saliency is computed using color, motion and disparity. Both the stereo vision and gesture recognition components are based on keypoints coded by means of cortical V1 simple, complex and end-stopped cells. Hand and face detection is achieved by using a linear SVM classifier. The system was tested on a child-sized robot.

Keywords: Hand gestures, Human-Robot Interaction, biological framework

1 Introduction

It is expected that, in the future, robots will employ a growing number of roles in society. Currently robots are mainly used in factory automation, but they are also deployed in service applications, medical assistance [1], schools [2] and entertainment, among other application fields. As they start to roam among us, there will be a need to interact with them in easy and natural ways. Human-robot interaction (HRI) research is therefore attracting more and more attention. Researchers are trying to develop new, easy and natural ways of programming robots, either by teaching them by manipulating a robot's hardware manually [3] or by creating programming interfaces so simple that even children can use them at school [2]. However, programming a robot still requires some skill that must be learned. In a world where robots navigate next to us, it will be necessary to be possible to interact with them effortlessly, using voice commands or gestures. In an ideal situation, robots should even be able to perceive some of our intentions by analysing the motions of our body.

The analysis and recognition of static hand gestures for HRI has been an interesting research area for some time and there have been many approaches. Some are intrusive, requiring the user to use specially designed gloves [4], while others are less intrusive, but still require specific hardware like Leap Motion [5] or Microsoft Kinect [6]. Other approaches rely on simple cameras and computer vision methods. A simple solution can be based on skin color segmentation and matching of previously stored gesture templates [7]. More complicated is to extract the skeleton of the hand [8] and use this for matching. For dynamic gestures there are methods that perform tracking and motion detection using sequences of stereo color frames [9], or gestures are characterized by using global properties of trajectories described by a set of keypoints [10]. Although many solutions may work quite well for a specific type of application, they are too simple for more complex gestures. In addition, they are often limited by fixed hardware devices or lighting conditions. In order to use gesture analysis and recognition for human-robot interaction, a system that is able to work under most lighting conditions and almost anywhere is needed.

In our previous work [7] we developed a biological and real-time framework for detecting and tracking both hand and head gestures. In this paper we present an extension of the system and also add new features to improve the system for human-robot interaction. The previously developed framework is based on multi-scale keypoints detected by means of models of cortical end-stopped cells [11,12]. We have improved the previous annotation of keypoints by using a fast binary descriptor that allows for fast and robust matching. We also added a combined disparity and motion saliency process so that the robot can initially focus on the hands of the user and track them. Gesture recognition is performed by matching the descriptors of the detected keypoints with the descriptors of previously stored templates. We also integrated a head and hand detector based on a linear SVM classifier.

The robot uses stereo vision for navigation, which also allows it to detect obstacles. Every time it finds an obstacle in front of it, it looks up and searches for the user's head in order to start the interaction. The robot attempts to center its own camera on the user's face, and then starts performing hand detection and gesture recognition. To detect a face/hand we employ a modified HOG (Histogram of Oriented Gradients) descriptor combined with responses of complex cells and a linear SVM to code the shape. The face and hand detectors were trained and evaluated on the FaceScrub dataset [13] and the Oxford hand dataset [14], respectively. The developed HRI system does not need any prior calibration and has been designed to run in real time.

2 Biologically inspired HRI system

In this section we describe all components of the developed system: (a) keypoint descriptor, (b) stereo vision for navigation and obstacle detection, (c) visual saliency, (d) face and hand detection, and finally (e) gesture recognition.

2.1 Keypoint descriptors for gesture recognition

In cortical area V1 there are simple, complex and end-stopped cells [12], which are thought to be responsible for part of the process of coding the visual input: they extract multi-scale line, edge and keypoint information (keypoints are line/edge vertices or junctions and also blobs). In this section we briefly describe multi-scale keypoint detection and the fast binary descriptor that we designed for matching keypoints. The descriptor is also based on V1 cell responses.

Keypoint detection: Responses of even and odd simple cells, which correspond to the real and imaginary parts of a Gabor filter [12], are denoted by $R^E_{s,i}(x,y)$ and $R^O_{s,i}(x,y)$, i being the orientation (we use $4 \leq N_\theta \leq 12$). The scale s is defined by λ, the wavelength of the Gabor filters, in pixels. We use $4 \leq \lambda \leq 12$ with $\Delta\lambda = 4$. Responses of complex cells are obtained by computing the modulus $C_{s,i}(x,y) = [\{R^E_{s,i}(x,y)\}^2 + \{R^O_{s,i}(x,y)\}^2]^{1/2}$. The process of 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. 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. On the other hand, keypoints are based on cortical end-stopped cells [11] and they provide important information because they code local image complexity. Since keypoints are caused by line and edge junctions, they are usually located at interesting locations of the image. When combined with a proper descriptor, they can be very useful for object categorization [15]. 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 to obtain precise keypoint maps $K_s(x,y)$. For a detailed explanation with illustrations see [11] and [15].

Binary keypoint descriptors: Creating a biological descriptor for keypoints is not a trivial task, mainly because responses of simple and complex cells, which code the underlying lines and edges at vertices, are not reliable due to response interference effects [16]. Therefore, the responses in a neighbourhood around a keypoint must be analyzed, the neighbourhood size being proportional to the scale of the cells. In our approach we developed a 128-bit binary keypoint descriptor based on the responses of simple cells, each bit coding the activation level of a single cell. Also, from a computational point of view, a binary descriptor is much faster to compute and to match than a floating-point one. This method is an improvement of the previous method [17]. We start by applying maximum pooling in a circular region around each keypoint, followed by zero-mean normalization and extraction of the maximum cell responses in 8 filter orientations. Then we combine the extracted values by a weighted sum, using a weight matrix previously learned using LDAHash [18] on the Notredame dataset [19]. Finally, we apply a threshold vector, also previously trained on the Notredame dataset, to set each of the 128 bits to 1 or 0. The resulting descriptor is a huge improvement of the previous one [17]. It is comparable to the SIFT-based LDAHash descriptor in terms of performance when tested on the Yosemite dataset [19].

The developed descriptor significantly outperforms other biologically-inspired descriptors. In terms of processing time, the descriptor is also very fast to compute and to match. Figure 1 (left) shows one example of matching using the present descriptor. The right graph shows a comparison between our descriptor(128 bits), BRISK(512 bits) and BRIEF(128 bits) and LDAHash (128 bits) over 200,000 patches of the Yosemite dataset.

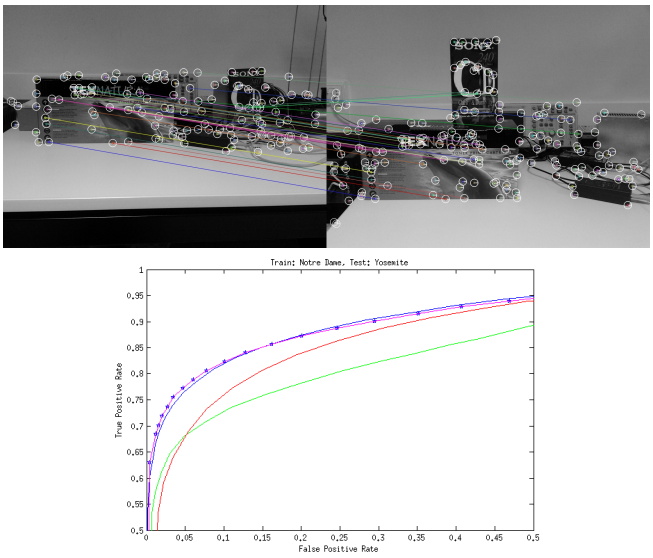

Figure 1. Top: Example of matching. Bottom: comparison between our descriptor(blue), BRISK(green), BRIEF(red) and LDAHash(pink) over 200,000 patches.

2.2 Stereo vision for robot navigation and obstacle detection

Stereo vision is a fundamental process for robot navigation: it allows the robot to detect open spaces, obstacles on its path and estimate the distance to those obstacles. It can also be useful for computing visual saliency. The algorithm we used to generate the disparity maps is the same as previously used in [17]: (a) resize the left and right images to a small size (160×120); (b) extract complex cell responses on circles around each pixel; (c) compare each pixel P in the left image to the next K pixels in the right image on the same line and starting from the same position P as in the left image (we used $K = 35$); (d) use the Hamming distance to find the best-matching pixels; and (e) apply median filtering (5×5 kernel) to reduce noise due to wrong matches. The computed disparity maps are then thresholded to find nearby obstacles ($t_d = 70$). Whenever the robot detects an obstacle by using the thresholded disparity maps, it looks up and evaluates if it is a person by trying to find a human face using a linear SVM classifier, as described in Section 2.4. Figure 2 (top) row shows one example of an image acquired by the robot and the respective disparity map.

2.3 Visual saliency

Visual saliency is also an important component of the real-time vision system, since by using it the robot can select important regions to process instead of processing entire images. The generated saliency maps are also useful to segregate hands from the background, to reduce clutter and to improve gesture recognition rates. The visual saliency algorithm we developed is an improvement of the one described in [17] and combines three different features: color, disparity and motion. The three features are processed separately and then merged into a single saliency map with equal weights. The top row of Fig. 2 shows an image of a person in front of the robot (left and right frames), the resulting disparity map, and the middle row shows the motion (left), color (middle) and disparity(right) saliency maps. The bottom row shows the resulting saliency map the thresholded map and the selected regions. Details are explained below.

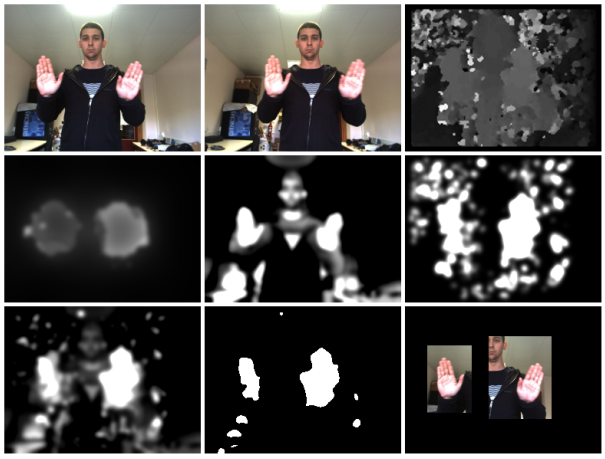

Figure 2. Top row: stereo images acquired by the robot camera and the disparity map. Middle row, from left to right: motion saliency, color saliency and disparity saliency. Bottom row, from left to right: resulting saliency map, thresholded saliency map ($t_f = 200$) and selected regions after blob detection and region growing.

Color Saliency: We build a stack of 6 retinotopic maps representing different channels in CIE L*A*B color-opponent space, which is based on retinal cones and provides a standard way to model color opponency. The first three channels code white, green and blue. The remaining three channels complement the first ones and represent black, red and yellow. After computing the maps, a stack of bandpass Difference-of-Gaussians (DoG) filter kernels with $\sigma_+ \in \{5, 10, 20\}$ and $\sigma_- = 2\sigma_+$ are used for blob detection. Since a saliency map does not need to be detailed, we compute them using subsampled color images for faster processing.

Disparity Saliency: To generate the disparity-based saliency map, we use the disparity and map computed in Section 2.2, and extract a single disparity layer where pixels with disparity $d = 100$ get the maximum value $M = 20$ and pixels bigger or smaller than d get smaller values according to their difference

from d. After this step we apply the same filtering used for the colour-based saliency maps.

Motion Saliency: To compute the motion-based saliency map, we first calculate the optical flow using Farneback's method for every pixel [20]. Then we process the flow's magnitude and direction separately, creating a feature stack of magnitudes and orientations of motion. The feature stack has 3 maps representing the magnitude (speed) of motion and also has 8 maps representing 8 different directions of motion with either 0 or the value itself. In practice, 0 occurs when a pixel is not moving in the preferred direction or with preferred speed. When a pixel is moving in the preferred direction or with preferred speed we consider the value to be the value of the pixel at that location. Objects moving in a certain direction with certain speed will thus cause large coherent regions in one of the maps of the stack.

Final Saliency Map: The final saliency is the equally-weighted sum of the three normalized saliency maps. After computing this map we threshold it ($t_f = 200$) to get only the nearest regions and then apply the fast blob detection algorithm from [21]. After blob detection, only the two biggest blobs are kept. Each blob is converted into a square region that afterwards is grown by 15 pixels in all directions, so that there is enough margin to apply the Gabor filters to extract the simple cell responses used to extract keypoints and build keypoint descriptors.

2.4 Face and hand detection

Detection of faces and hands is achieved by coding responses of cortical complex cells within a region into a feature vector and by using a classifier (linear SVM) to predict whether a face/hand is inside the detection region or not. The process is similar to [22], but in our case the detection process employs a single Gabor filter scale ($\lambda = 6$) and 8 orientations ($N_\theta = 8$) for the complex cells (see Section 2), but also in combination with several scales of the HOG-like features (several sizes) over the entire image, and then a sliding window (6×6 blocks) to scan all blocks inside the sliding window's region. At each layer, the pooling cell size is increased, but the detection window size and the block's size remain the same. To this purpose, we use between 6×6 and 10×10 pixels per cell with a stride of 1 pixel. Finally, non-maximum suppresion is applied to eliminate multiple detections of the same face/hand (see below). We used the FaceScrub dataset [13] and the Oxford Hand dataset [14] to train and evaluate our face and hand detectors. For each detector we train an initial classifier using the positive and a random set of negative examples, then we use it to scan over images not containing faces or hands and collect false positives, and then we do a second round of training by including these hard false positives into the negative training set. The FaceScrub dataset consists of 107,818 face images of 530 celebrities (265 male and 265 female), although only 80,659 faces could be downloaded successfully and a few samples being unusable for training. For negative samples we took 100,000, 42×42 pixel patches from random images of the SUN database [23]. For training our hand classifier we used the Oxford

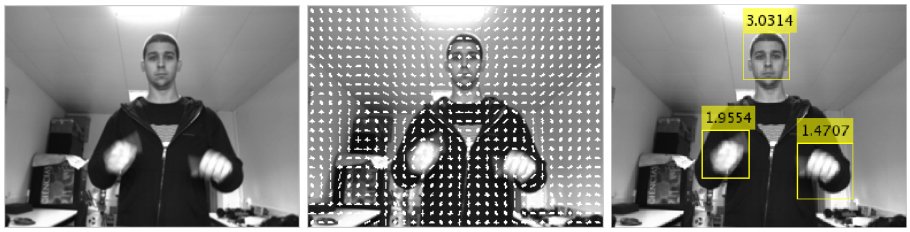

Figure 3. (Left to right) a person in front of the robot in gray scale (left) coded by HOG-like features (middle) with detected face and hands (right).

hand dataset which contains 13,050 annotated hand instances (26,100 mirrored samples). Again, we took 30,000 random 42×42 pixel patches from the SUN database as negative samples.

HOG-like features: A modified version of the Dalal and Triggs HOG features is used here for person detection. In their implementation [22] they use the RGB colour space with no gamma correction, 1D gradient filters, linear gradient voting into 9 orientation bins, 16×16 pixel blocks of four 8×8 pixel cells, a Gaussian spatial window with $\sigma = 8$ pixels, L2-Hys block normalization, a block spacing stride of 8 pixels, and a 64×128 detection window for training the linear SVM classifier for pedestrian detection. Here we use an adapted and slightly modified version of the previous procedure for face and hand detection: (a) we use only grayscale information for speed purposes, (b) complex cell responses are used as gradient information, (c) no Gaussian window is applied, (d) the L2-norm is used instead of L2-Hys and (e) a 42×42 detection window for training. Complex cell responses provide a good alternative for gradient information since they are also robust to noise. In addition, linear gradient voting can be skipped because of the cells' oriented responses. We use 12×12 pixel blocks of four 6×6 pixel cells with 50% block overlap; see below for parameter assessment and evaluation. Our selection ensures optimal performance while also complying also with Dalal and Triggs' recommendations [22] of having many orientation bins in combination with moderately sized, strongly normalized and overlapping descriptor blocks. See Fig. 3 (middle) for the bio-inspired HOG-like features.

Classification: To detect a face or a hand, features in a detection window are classified using a linear SVM. The face and hand classifiers are trained using the FaceScrub dataset and the Oxford hand dataset, respectively, with a soft linear SVM (C=0.01) using LIBLINEAR [24]. As mentioned above, we used a 42×42 pixel detection window for training both classifiers, since the sizes of hands and faces in the image relative to the robot are similar. This results in 6×6 blocks to be used by the classifiers across the image at all scales. We found that a detection window of 42×42 pixels for training constitutes a good trade-off between performance and running speed. Increasing the detection window's size beyond 42×42, although increasing detection performance, has a significantly higher computational cost due to more features being used for classification.

Non-maximum suppression: The detection window is applied to salient image regions using a sliding window approach where multiple detections of the

same head or hands often occur. To remove multiple detections due to the sliding window, a non-maximum suppression technique is applied to discard overlapping windows: when two windows overlap at least 50%, the window with the weakest classification response is discarded. To this purpose, we use the unsigned SVM classification output in order to determine a window's classification response. Fig. 3 (right) shows detected regions of face and hands after non-maximum suppression.

Concerning performance evaluation and optimization, several factors have been evaluated in the classifier training stage, namely Gabor filter scale (λ), number of orientations (N_θ), cell size, block size and overlap. Smaller sets of 2000 positive and 2000 negative random samples for training, and 1000 positive and 1000 negative random samples for testing were used for cross-validation and parameter optimization for both classifiers. We used detection error trade-off (DET) and miss rate (better: 1.0 - Recall) measures to quantify the performance.

First, the scale of the cortical cells was analyzed in order to determine the optimal λ. Fig. 4 (top-left) shows the overall performance of seven different scales $\lambda = [4, 10]$. Smaller scales yield better performance than bigger scales, mainly due to lines and edges being better encoded by smaller filters. Moreover, by choosing a smaller λ the processing time decreases. Here, $\lambda = 4$ performed best. Fig. 4 (top-middle) shows the performance impact of the number of orientations used ($N_\theta \in [4, 12]$) in the HOG-like feature bins. Increasing the number of orientations beyond 8 does not improve performance significantly. Therefore, for the final classifier we chose $N_\theta = 8$ orientations which gave a 1.4% and 8.03% miss rates for face and hand detection, respectively. Three other key factors taken into account were the block size vs. cell size vs. block overlap. Fig 4 (middle and bottom) shows three graphs with 0% (left), 25% (middle) and 50% (right) block overlap, with block sizes ranging from 1×1 to 4×4 and pooling cell sizes from 4×4 to 10×10 for faces (middle) and hands (bottom). From all tested combinations, block sizes of 2×2 with 6×6 pooling cells and 50% overlap, gives the best performance with a 1.1% miss rate for face and 8.6% for hand detection. Fig. 4 (top-right) shows the overall performance of the two detectors.

2.5 Gesture Recognition

In order to be able to recognize gestures, the robot keeps in its memory a small set of templates of different hand gestures. These templates have been prepared prior to robot operation, and by applying exactly the same processing as done during real-time robot operation. At the moment we use a set of 7 different gestures for both hands, with several samples for each gesture with different backgrounds and sizes. After finding the hand regions, the robot then processes those regions for keypoint extraction and their descriptors. Descriptor matching is based on the 128-bit Hamming distance: when the distance between two keypoint descriptors is smaller than 30, a match is accepted. When at least 4 matches between the acquired hand region and a template are found, the robot assumes that the gesture that corresponds to the matched template has been detected. For faster computation time and higher reliability of the system, we extract keypoints at

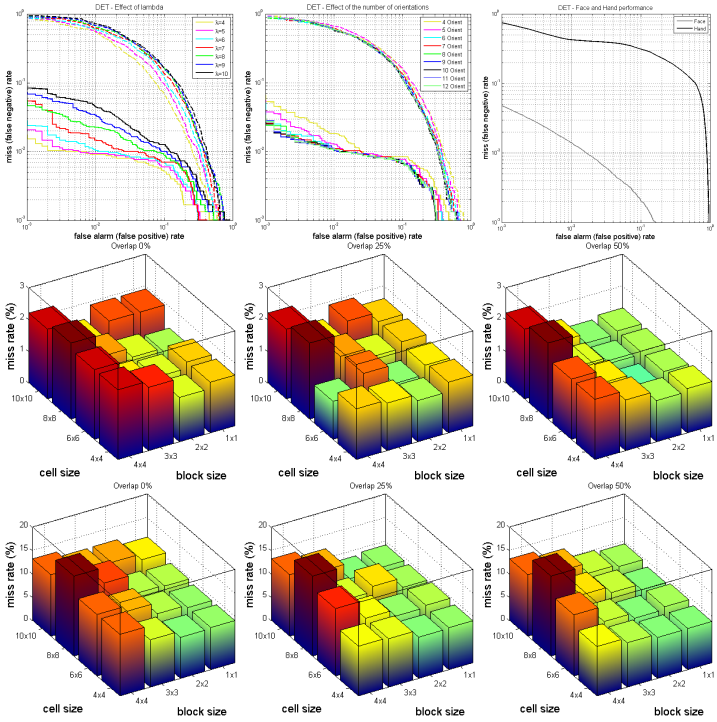

Figure 4. HOG-like feature parameter performance. Top row: effects of λ (left) and the number of orientations (middle-left) in face (full lines) and hand (dot lines) detection. The mid-right and right plot shows the overall performance of our face and hand detectors. Middle and bottom rows, left to right: block vs. cell size combinations with 0%, 25% and 50% block overlap, and with block size from 1×1 to 4×4 and cell size from 4×4 to 10×10 (bluer is better) for face (middle) and hand (bottom) detectors. The best result is given by 2×2 blocks with 6×6 cell size and 50% overlap with 1.1% miss rate for face and 8.6% for hand detection

scale $\lambda = 12$ and use simple cell responses at scale $\lambda = 6$ for the descriptors. By extracting keypoints at a coarser scale, they become more stable. By using scale $\lambda = 6$ on extracted keypoint locations, more detail is available to describe the keypoints.

3 Tests and Results

To test the developed system we used a child-sized Pioneer 3DX robot, equipped with a Bumblebee-2 stereo camera, a PhantomX robot turret for pan and tilt movement of the camera, and ultrasonic and laser rangefinder sensors (see Fig. 5 left). The Bumblebee-2 camera captures images at a resolution of 1024×768 pixels. The range sensors are only used for emergency collision avoidance, not for navigation. A structure has been mounted on the robot in order to make it taller, providing the point of view of a child with a height of $115cm$ and eyes

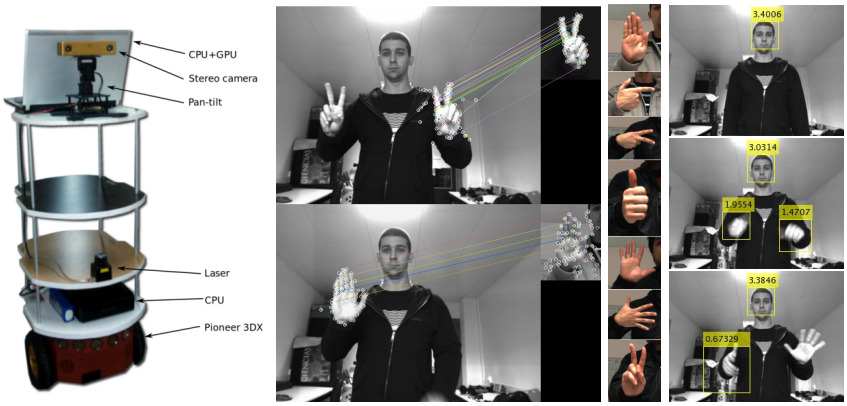

Figure 5. Left: robotic platform used for testing. Middle left: examples of recognition of two different gestures. Middle Right: some of the gesture templates. Right: detection of heads and hands using the SVM classifiers.

at 110cm. The robot has been set up with ROS (Robot Operating System). Although the robot is of mecanoid type, its pan and tilt system combined with the stereo camera convey the idea that it has a neck and a head with two eyes. Since it is programmed to focus on a person's head in the center of the image, it seems like it is looking the person in the eyes. This makes it much more engaging than a robot with static cameras and no neck movements.

Fig. 5 right shows some results: on the left we can see the mobile robot platform used for testing; on the middle left column we show two examples of gesture recognition using the extracted keypoints and their descriptors; on the middle right column we show some of the gesture templates; and on the right we show some examples of head and hand detection using the SVM classifiers.

The system proved to work quite well, being able to recognize the 7 different gestures for both hands in most situations. At start it failed to recognize some of them but as we added more gesture templates with different sizes and backgrounds the gesture recognition was improved.

As usual with most vision systems, we noticed that under bad illumination the system can sometimes fail to detect hands or head and thus may not be able to recognize gestures. Failure in detection of hands and recognizing gestures could also happen during fast hand movements, since in these cases the image of the hands is blurred.

4 Discussion

In this paper we presented a biologically inspired vision system for human-robot interaction. It allows the robot to navigate and to evaluate if an obstacle is, in fact, a person by trying to find a human face whenever it encounters an obstacle. The system also allows the robot to direct its attention to important visual areas using an attention model which is based on color, disparity and motion. The presented methods allow the robot to identify human hands in salient regions

and then recognize the gestures being made by using keypoints and keypoint descriptors based on V1 cell responses. Although biologically inspired methods usually require a high computational power and long computation time, due to the many filter kernels at several orientations and scales, by choosing only a few scales the system is able to run in real time. The SVM classifiers showed to be fast in detecting faces and hands in an image using a sliding window. Faces had the best performance overall with few false negatives, while hands had less performance in detection mainly due to the large intra-class variability. The keypoints and their descriptors proved to be quite robust and work quite well for recognizing the gestures previously stored as templates. Although sometimes few keypoints were matched to wrong keypoints, most of them were matched to the correct ones. These wrong matches can easily be eliminated by using some geometry between groups of keypoints to validate or invalidate matches. Experiments showed that the system can now be programmed to execute different actions according to specific hand gestures.

As future work we plan to add more gestures to the system, increase the precision such that individual fingers can be detected, and integrate facial expression recognition in order to allow the robot to act according to a person's mood. We also plan to use the ability of recognizing gestures and facial expressions as a way to teach the robot simple tasks through reinforcement learning: by using different gestures or facial expressions it will be possible to tell the robot that it is doing the right or the wrong thing. Another part of future work consists of integrating the binary keypoint descriptor into the GPU implementation of the keypoint extractor in order to free the CPU for other future developments. Future work will also address motion prediction, a process that occurs in cortical area MST.

ACKNOWLEDGEMENTS

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) projects UID/EEA/5009/2013 and SparseCoding EXPL/EEI-SII/1982/2013; and by FCT PhD grants to MS (SFRH/BD/71831 /2010) and MF (SFRH/BD/79812/2011).

References

1. Messias, J., Ventura, R., Lima, P., Sequeira, J., Alvito, P., Marques, C., Carrio, P.: A robotic platform for edutainment activities in a pediatric hospital. In Proc. IEEE Int. Conf. Autonomous Robot Systems and Competitions (2014) 193–198
2. Saleiro, M., Carmo, B., Rodrigues, J., du Buf, J.: A low-cost classroom-oriented educational robotics system. Social Robotics SE **8239**(8) (2013) 74–83
3. Kronander, K., Billard, A.: Learning compliant manipulation through kinesthetic and tactile human-robot interaction. IEEE Tr. on Haptics **7**(3) (2014) 367–380
4. Kumar, P., Verma, J., Prasad, S.: Hand data glove: a wearable real-time device for human-computer interaction. Int. J. of Advanced Science and Technology **43** (2012) 15–25
5. Han, J., Gold, N.: Lessons learned in exploring the leap-motion(tm) sensor for gesture-based instrument design. Proc. New Interfaces for Musical Expression (2014) 371–374

6. Qian, K., Niu, J., Yang, H.: Developing a gesture based remote human-robot interaction system using kinect. Int. J. of Smart Home **7**(4) (2013) 203–208
7. Saleiro, M., Farrajota, M., Terzic, K., Rodrigues, J., du Buf, J.: A biological and realtime framework for hand gestures and head poses. Universal Access in Human-Computer Interaction. Design Methods, Tools and Interaction Techniques for eInclusion SE **8009**(60) (2013) 556–565
8. Ionescu, B., Coquin, D., Lambert, P., Buzuloiu, V.: Dynamic hand gesture recognition using the skeleton of the hand. EURASIP J. of Applied Signal Processing (13) (2005) 2101–2109
9. Ghaleb, F., Youness, E., Elmezain, M., Dewdar, F.: Hand gesture spotting and recognition in stereo color image sequences based on generative models. Int. J. of Engineering Science and Innovative Technology **3**(1) (2014) 78–88
10. Suk, H., Sin, B., Lee, S.: Hand gesture recognition based on dynamic Bayesian network framework. Pattern Recogn. **43**(9) (2010) 3059–3072
11. Rodrigues, J., du Buf, J.: Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection. BioSystems **2** (2006) 75–90
12. Rodrigues, J., du Buf, J.: Multi-scale lines and edges in V1 and beyond: brightness, object categorization and recognition, and consciousness. BioSystems **95** (2009) 206–226
13. Ng, H., Winkler, S.: A data-driven approach to cleaning large face datasets. Proc. IEEE International Conference on Image Processing (ICIP) **265**(265) (2014) 530
14. Mittal, A., Zisserman, A., Torr, P.H.S.: Hand detection using multiple proposals. (2011)
15. Terzic, K., Rodrigues, J., Lam, R., du buf, J.: Bimp: A real-time biological model of multi-scale keypoint detection in v1. Neurocomputing (150) (2015) 227–237
16. du Buf, J.: Responses of simple cells: events, interferences, and ambiguities. Biol. Cybern. **68** (1993) 321–333
17. Saleiro, M., Terzic, K., Lobato, D., Rodrigues, J., du Buf, J.: Biologically inspired vision for indoor robot navigation. Image Analysis and Recognition, Part II **8815** (2014) 469–477
18. Strecha, C., Bronstein, A., Bronstein, M., Fua, P.: Ldahash: Improved matching with smaller descriptors. IEEE Tr. on Pattern Analysis and Machine Intelligence **PP**(99) (2012) 66–78
19. Brown, M.: Learning local image descriptors data (2011) [Online; accessed 17-Feb-2015].
20. Farneback, G.: Two-frame motion estimation based on polynomial expansion. Proc. 13th. Scandinavian Conf. on Image Analysis **2749** (2003) 363–370
21. Saleiro, M., Rodrigues, J., du Buf, J.: Automatic hand or head gesture interface for individuals with motor impairments, senior citizens and young children. Proc. Int. Conf. Softw. Dev. for Enhancing Accessibility and Fighting Info-Exclusion (2009) 165–171
22. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition **1** 886–893
23. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. Computer vision and pattern recognition (CVPR), 2010 IEEE conference on (2010) 3485–3492
24. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9** (2008) 1871–1874