

AUGMENTED VIRTUAL STUDIO

Matei Mancas¹, Michel Bagein², Nicolas Guichard¹, Sullivan Hidot², Caroline Machy⁴, Sidi Mahmoudi², Xavier Siebert³

¹ Laboratoire de Théorie des Circuits et Traitement du Signal (TCTS), Faculté polytechnique de Mons (FPMs), Belgique

² Service Informatique (INFO), Faculté Polytechnique de Mons (FPMs), Belgique

³ Service Mathématiques et Recherche Opérationnelle (MATHRO), Faculté Polytechnique de Mons (FPMs), Belgique

⁴ Multitel (ASBL), Belgique

ABSTRACT

The Augmented Virtual Studio (AVS) project aims at acquiring the tools of video analysis and visualization needed to achieve advanced interfaces or interaction with virtual avatars and virtual worlds. Those techniques consist in data visualization, object segmentation, tracking and identification of blobs but also sketch recognition and more generally faces and objects recognition. All these tools were used to build three practical applications.

KEYWORDS

Video tracking, augmented reality, interfaces, object recognition, visualization

1. INTRODUCTION

Gestural real-time interaction with multimedia data and virtual worlds is of a great importance for live digital arts performances. These topics are part of the COMEDIA research axis within the Numediart project. There are many existing tools for real time video analysis and visualization and we tested here some of them. We also implemented tracking, blob identification and object recognition algorithms in order to build three pilot applications which will be used as test here and which will be enhanced in further projects. In the next section the blob tracking and identification is described for simple (2 blobs) and complex (6 blobs) situations, then a section is dedicated to object recognition. In a third step, several softwares which were tested here are described, while the last section deals with the three pilot applications. Finally a discussion about the results can be found.

2. TRACKING, BLOB IDENTIFICATION AND RELATIVE POSITIONS

2.1. Segmentation and tracking in EyesWeb

The first step for the three applications done in this session is the detection and tracking of IR blocs in real time. We have used Eyesweb to perform this step. EyesWeb is a programming tool using blocks that can work on several kinds of signals, such as video. It is designed to facilitate the interpretation of these signal streams [7]. We can divide our system of tracking on three major steps:

1. *Blob Detection (segmentation)*
2. *Blob Tracking*
3. *Blob Trajectory*

2.1.1. Blob detection

The present system's analysis starts from foreground segmentation based on the analysis of the infra red video streams [5]. This analysis provides a binary mask of the spatial extension of the region of interest through time (blob detection). The signal coming from the IR camera is not affected by illumination variations but might be affected by light reflections or some static infra red sources. We processed the video stream with a background subtraction to eliminate static elements and focus only on the moving objects. Then, we binarized the signal with an empirically-tested threshold value to extract the moving regions of interest (blobs or segments). Figure 1 (a) shows the IR lights located on the two hands used for the first application. Figure 1 (b) shows the IR lights located on the six region of a human (head, two hands, navel, two feet) for the avatar application.



Figure 1: Background subtraction with two blobs (above) and six blobs (below).

2.1.2. Blob Tracking

Resulting from the blob detection (preprocessing) step, we obtained a binary image where white represents the foreground objects (Figures 1 and 2). The main goal of this step is to assign a

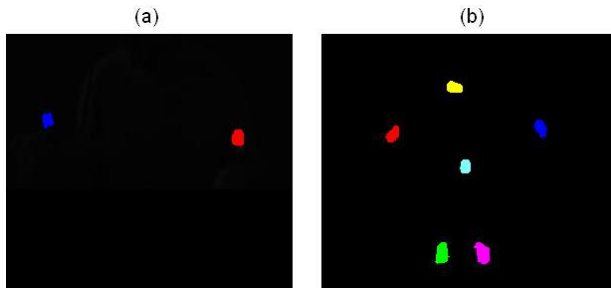


Figure 2: (a) Blob Tracking with two blobs (a) and six blobs (b).

label that identifies the different white blobs, and to track them. In result we have every blob identified by a specific color. The parameters taken into account in our tracking are:

- **Multi-blob tracking**
We used the multi-blob tracking because we have more than one blob to track in our applications.
- **Distance**
It represents the distance between the blobs detected, to have a good performance of tracking we used the smallest value possible of distance '1', if the distance between blobs is less than this value we'll have only one blob detected.
- **Number of blobs**
We define as input for the tracking bloc the number of blobs to be tracked. For the first application this number was 2 (two hands) whereas for the avatar application this number increases to 6 because we have six objects to track. The tracking result can be seen on Figure 2.

From the multi-blob tracking, we obtained some important values and indicators:

- (a) **Confidence Indicator:**
This indicator takes the value '1' if the blob is detected and '0' otherwise. The number of confidence indicator is the same as the number of blobs.
- (b) **Blob2D coordinates:**
These coordinates represent the coordinates (x, y) of each blob detected, these values changes in real time during the blobs displacement.

2.1.3. Blob Trajectory

After tracking we can generate for each blob a trajectory consisting of the temporal sequence of the points provided as input. Figure 3 shows the trajectory of two tracked blobs.

2.2. Blob identification and relative positions

After releasing the tracking of different IR blobs, we have to identify those blobs. This identification differs from an application to

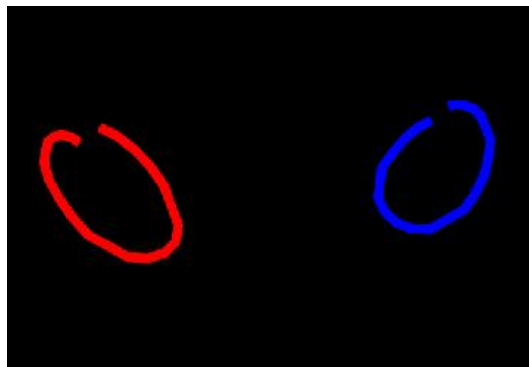


Figure 3: Blob Trajectory with two blobs.

another since each of them uses the results of tracking IR for a specific purpose.

2.2.1. Blob identification in a simple context

In this application we have only 2 IR blobs tracked representing the two hands. The first detected blob represents the left hand and it is used for navigating within a multimedia wall (images, videos, texts) if the second IR light is not present (confidence indicator =0). When the second IR light is present and the confidence indicator changes to 1, the first blob is used to move images, texts or videos in the multimedia wall.

2.2.2. Indicators of zoom and rotation

Using the coordinates of the two blobs detected we have computed two indicators used for zoom and rotation.

- **Zoom Indicator**
The indicator of zoom is calculated by comparing the distance between the two blobs and a threshold, this threshold represents the initial distance (Th) between the two blobs (at the first detection of the second IR light). If this difference is positive there is a request for a zoom in and if it is negative, for a zoom out. Figure 4 shows the relation between the distance and the zoom request. The formula used for distance calculation is:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

With (x_i, y_i) are respectively the coordinates of the first and second blob. If we want to make a request of zoom (in or out) we have only to change the distance between the IR lights which are located on the user's hands¹.

- **Rotation Indicator**
The Rotation Indicator is computed by the same method that the zoom but it uses a different distance in order to avoid conflicts. The distance for rotation is calculated using just the coordinate y and not x by this formula (the threshold represents the initial distance with horizontal coordinates y):

$$D_y = y_1 - y_2 \quad (2)$$

¹To make this request of zoom the 2 IR lights must be visible

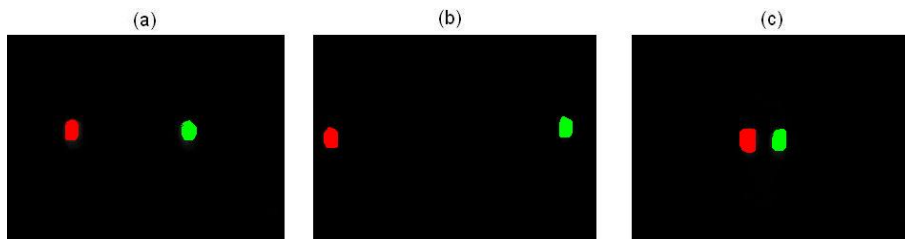


Figure 4: (a) Initial position of blobs (no zoom) - (b) Zoom In ($D > \text{Threshold}$) - (c) Zoom out ($D < \text{Threshold}$).

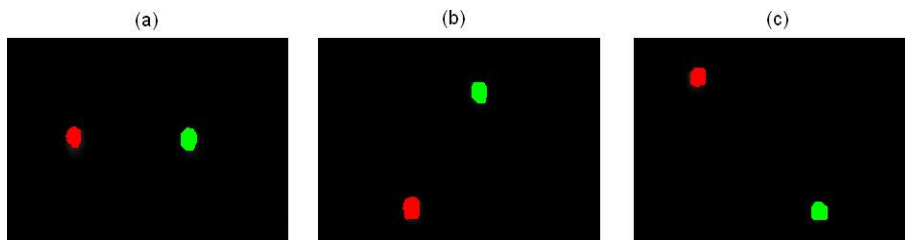


Figure 5: a) Initial position of blobs (no rotation) - (b) Left Rotation ($D_y > 0$) - (c) Right Rotation ($D_y < 0$).

If we have a positive value of D_y , there is a request of left rotation else we have a request of a right rotation. Figure 5 displays the relationship between D_y and the rotation request.

2.2.3. Blob identification in a complex context

In this situation we have six blobs to detect. These blobs represent the IR lights located on the different regions of human (head, two hands, two feet and navel). Our blob detection provides us with the coordinates of each blob and its confidence indicator, but we have to associate each blob to the corresponding human region. In order to achieve this blob identification we have followed these steps:

1. Detection of the 6 blobs

Before starting the tracking we have to detect all the six blobs (the IR lights have to be lit).

2. Blob Identification

After detecting the six blobs we can start our identification using the horizontal and vertical coordinates (x and y), this identification is done for each blob as follows. To identify the head we have simply taken the blob which has the biggest value of y , this blob will be marked, now we have five blobs to identify. To identify the left hand we have taken the blob which has biggest value of x . this blob will be marked, now we have four blobs to identify. To identify the right hand we have taken the blob which has the smallest value of x . this blob will be marked, now we have three blobs to identify. To identify the first feet we take the blob which has the less value of y and to know if it is the left or the right one we have to identify the second feet which represent now the blob that has the less value of y between the two remaining blobs. After the identification of the two feet we know that the left one is simply the one which has the biggest value of x and the right feet will be automatically the second feet. The two blobs will be marked too. Now we

have only one unmarked blob which is obviously the navel. Figure 6 shows the result of blob identification by replacing each blob by its corresponding human region.

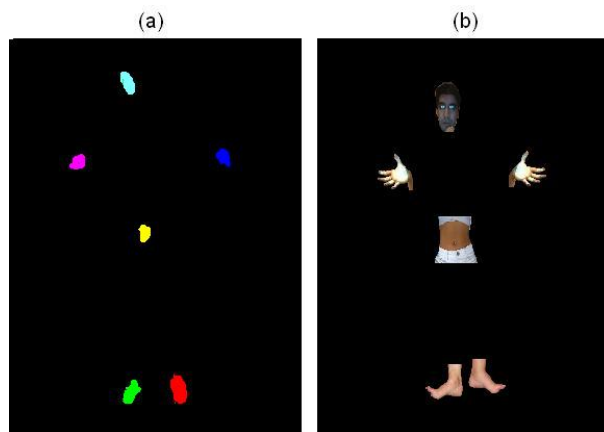


Figure 6: (a) Blob tracking (6 blobs) - (b).Blob Identification (6 blobs).

3. Blob Tracking

After identification of all the blobs we can start our tracking for the six different human body areas.

3. OBJECT DETECTION AND RECOGNITION

3.1. Face detection

We implemented the Viola and Jones [15] face detection algorithm from OpenCV as an EyesWeb XMI block. This algorithm uses files where data concerning a training step on face detection was

already achieved. This training data is used in a Haar-like classifier which will be able to detect faces. It is possible to train the algorithm on other training sets of objects (not necessary faces) to use this block in a more general object detection. The results of this block are satisfying. There are very few false detections, especially if the EyesWeb tracking is followed by a blob-tracking technique which will eliminate the false alarms which only appear in some particular frames and which have a low time persistence. The faces are well detected when seen from the front, but the algorithm performances highly decrease when the face is seen from the side. Face detection is achieved in real-time only if the larger size of the frames is less than 500 pixels, that is why it is interesting to decrease the image size to make face detection and then get the initial size to replace the face mask within the real image (Figure 7).



Figure 7: Face detection block implemented in EyesWeb.

3.2. Sketch recognition

3.2.1. Overview

The aim of this section is to detect a hand-made sketch by an actor and to replace the sketch with a corresponding 3D virtual object with which the actor can interact. More specifically, the tasks to be performed are as follows:

1. detect when the actor is making a 2D sketch and record it.
2. detect which object has been drawn, by comparison with a database of known sketches.
3. display a 3D image corresponding to the object drawn, on top of the video recording of the actor.
4. allow the actor to manipulate interactively the 3D virtual object.

To achieve these tasks, the actor is equipped with 2 IR LEDs (one in each hand) and filmed with two cameras: one conventional digital camera and one IR camera, so that (the numbering below corresponds to the above mentioned tasks):

1. the first LED is used by the actor to signal when (s)he is starting or stopping to draw.
2. when the start signal is triggered, the movement of the second LED is tracked by the IR camera, forming a 2D trajectory.
3. when the stop signal is triggered, the 2D trajectory is extracted and normalized to a 128x128 pixels image (the 'sketch'). This sketch is then compared with a database of 2D sketches using a simple recognition system described below.

4. each sketch in the database has been previously assigned to a 3D object. The database sketch that most closely resembles the hand-made one will be selected, and the corresponding 3D object will replace the hand-made sketch in the video, at the same position and with the same size.
5. turning the second LED off and on again triggers a new mode of interaction, to grab and to move the 3D object.

It is important to point out that IR tracking was preferred over color-tracking, because it is more reliable and because the LEDs can easily be placed in the hands of the actor without hampering his/her motion.

For all the above mentioned operations, we relied on the EyesWeb software [4], which contains efficient algorithms for video-based tracking. Using EyesWeb's Software Development Kit (SDK), we added some modules (also known as 'blocks') tailored to our needs. For example, we implemented 2D Fourier-based distances between images. Other functions, such as the image moments (see below), can also easily be incorporated into EyesWeb, for example using known libraries such as OpenCV [3].

3.2.2. Sketch recognition system

Existing sketch recognition systems fall into two main categories: gestural (e.g., [1]) or free-sketch system. Gestural systems require each sketch to be drawn in a particular style (making it a gesture, rather than a shape). In this application, no particular constraints were imposed on the actor's gesture, so that our recognition system would fall in the 'free-sketch' category. We should emphasize that sketch recognition is a wide field of research, encompassing among others feature-based, vision-based, geometry-based, and timing-based recognition algorithms. For the 'Augmented Sketch' application, simple global properties were tested:

- pixel-per-pixel differences.
- Fourier-based distances
- width/height ratio,
- ratio of filled/unfilled pixels
- higher order image moments (e.g., Hu moments [8])

From our simple tests, it appears that the width/height ratio and the ratio of filled/unfilled pixels are not efficient in practice, because the actor can make drawings of any height, width and size. These could become useful if some conventions were set, for example houses are wider than tall, trees are taller than wide, ... However, simple pixel-per-pixel differences (or, equivalently, between their Fourier transforms, see Eq.3) gives reasonably good results. Hu moments give some additional information but were not necessary for our limited database (see Fig. 8).

The quadratic misfit between two images is given by:

$$Q = \frac{\sum_i \sum_j |F_{ij} - G_{ij}|^2}{\sum_i \sum_j |F_{ij}|^2} \quad (3)$$

Because this expression is quadratic, it can be evaluated in real- or Fourier-space (Parseval's theorem). Evaluating it in reciprocal space has the advantage that the sums in Eq.3 can be calculated on a subset of the Fourier coefficients (e.g. the low-resolution ones) without extra computation. Low-pass filtering in real space would require more computations on each frequency is changed.

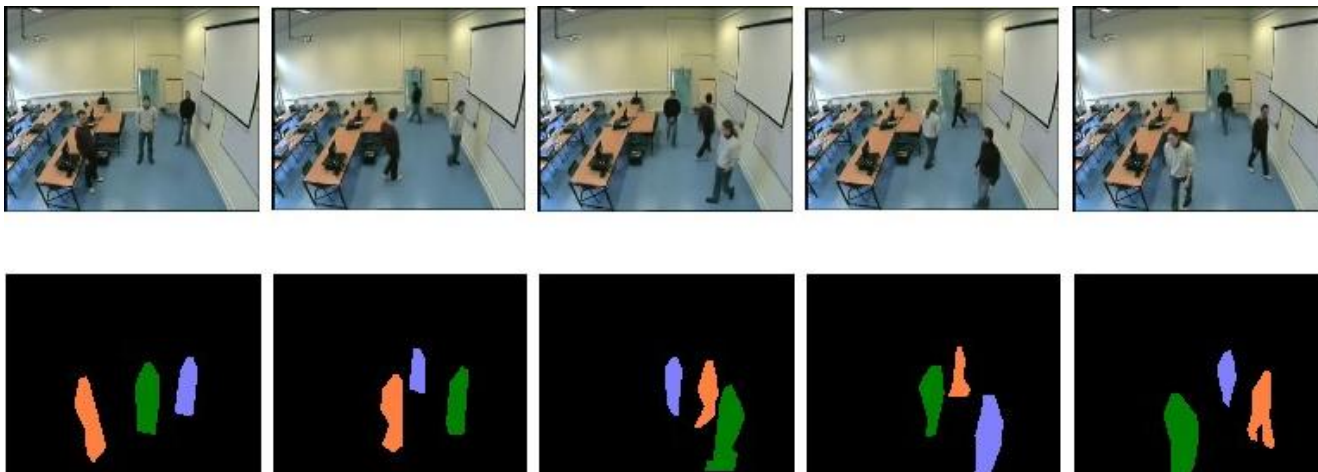


Figure 9: Object recognition and tracking using Gaussian Mixture Model.

[11] that SIFT outperforms a lot of descriptors-based algorithms by comparing for different types of interest regions. SIFT is done

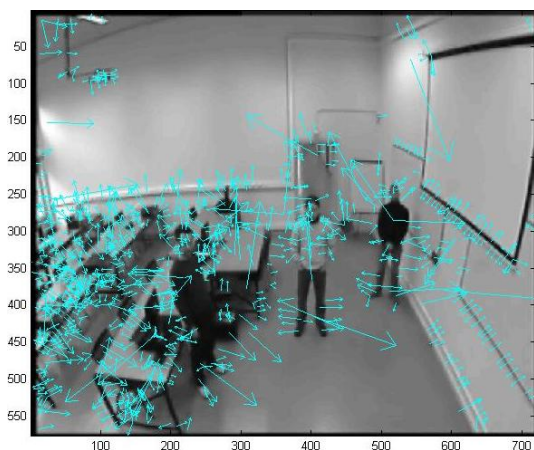


Figure 10: Application of SIFT to keypoints selection.

in four successive steps:

1. *Scale-space extrema detection*
2. *Keypoint localization*
3. *Orientation assignment*
4. *Keypoint descriptor*

Figure 10 shows the keypoints selection of an image. The arrows indicate the keypoint vector with their localization, scale and orientation. Because SIFT detects the main local features of an image, it can be applied to object recognition. Indeed, once the keypoints have been localized and computed, a simple Euclidean distance between the descriptors in their vector form can be achieved

to identify persons in successive frames even if the face information is not available. However, and despite of its robustness to recognize objects among clutter and occlusion, SIFT is not really adapted for our own database. Figure 11 shows the poorly results using this approach. In (a), SIFT can detect the local feature of a person when embedding in a larger image. In (b), because of the keypoints changing over the frames, SIFT is naturally not able to track an object subjected to various changes in positions. To correct this problem, one way to explore is to match between SIFT and the Log-polar transformation. Log-polar technique [16] can be viewed as an analogy with the structure of the retina and provides an increase of the size range of the object to be detected and tracked. Our idea is then to cross the performances of SIFT with Log-polar sampled image in a top view. The problem would then sum up to objects moving around regarding only the scale and orientation.

4. VISUALIZATION

4.1. Processing

Processing [13] is an open source project initiated by Casey Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. It is a programming language and integrated development environment (IDE) built for the electronic arts and visual design communities', which aims to teach the basics of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks. The IDE is licensed under the GNU General Public License. One of the stated aims of Processing is to act as a tool to get non-programmers started with programming, through the instant gratification of visual feedback. The language builds on the graphical capabilities of the Java programming language, simplifying features and creating a few new ones.

4.2. Blender

With the blob identification, it is easy to imagine that relative position of several points located on performer's body could be used to control a virtual character or avatar. When blobs are located close to the performer's joints, we can control a virtual 3D skeleton, or

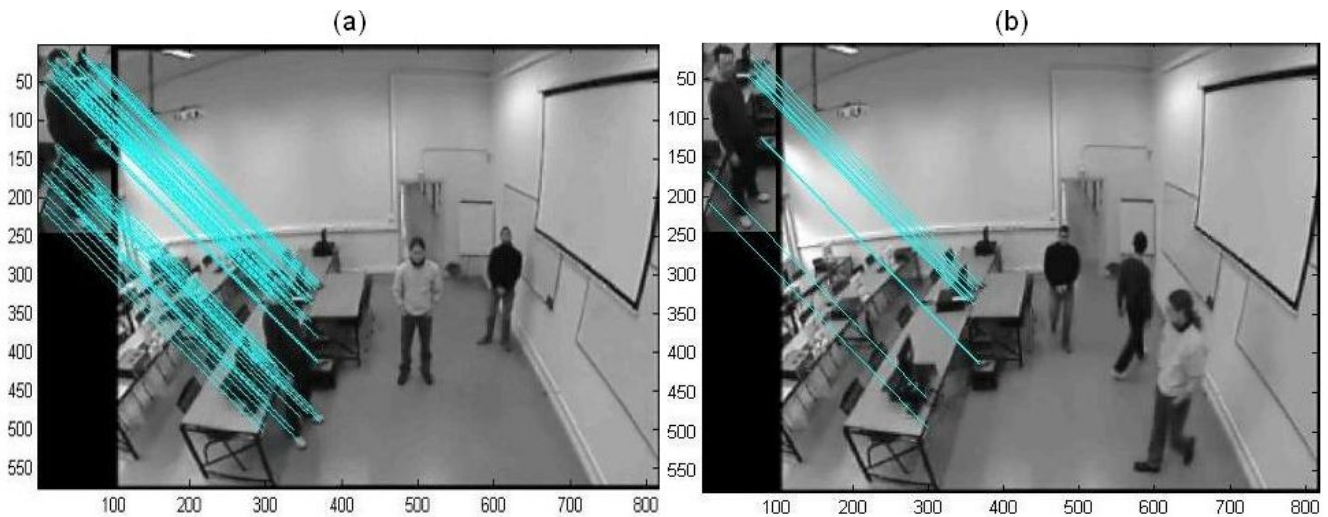


Figure 11: *SIFT applied to our own database. (a) Image local features well detected and matched in a larger image. (b) SIFT is not adapted to various changes in positions.*

more, a full virtual body with flesh. There are several powerful tools on the shelf to create 3D avatar. Blender [2] is one of those and its advantage is to be free and open with a large and very active user community. The 'mancandy' character was designed in this environment with large number of joints (wrist, elbow, shoulder, neck ...) which their realistic degree of freedom (DoF). To make a full control of this avatar, each joint must be 'linked' to the performer's joint but the set of control is quite large. In order to reduce the size of control set, inverse kinematics (IK) can be applied to retrieve intermediate location of uncontrolled joints. This approach, widely used in robotics engineering (and implicitly by puppeteers) attempt to determine optimal location of each uncontrolled joint (elbow, knee, etc.) in a bone chain. Usually, the inverse kinematics provides an infinity of acceptable solutions. To limit the number of solutions, several physiological constraints can be added (joint with few DoF, coarse limits, forces, stresses, etc.). In broadcasting or live performance, the interactivity brings a major constraint: the kinematics of full movement need to as realistic as possible but without any a priori knowledge on the performer's gestures. Without any knowledge of the instantaneous gesture, it is hard to define realistic joint paths. For example, when one lift a glass to the mouth, the elbow can be close to the thorax but when one lift a hairbrush to the front, the elbow is also lifted to initiate the brushing gesture from head front side to back side. Inverse kinematics engine featured in Blender is dedicated to generate intermediate frames in motion sequences, based on the knowledge of initial and the final pose. This engine is activated at edit time (preview): to help the user to locate intermediate joint in pose, by control the location of one selected bone (hand). When several poses are defined, IK is also used to generate each intermediate frame between poses in order to build gesture sequences (one forward step). Gesture sequences can be used to animate an avatar (like in video games) in a movie but this is not realistic enough: sequences can not support fine interaction with its context: walking avatar seems to 'skate', walk speed is constant, etc.

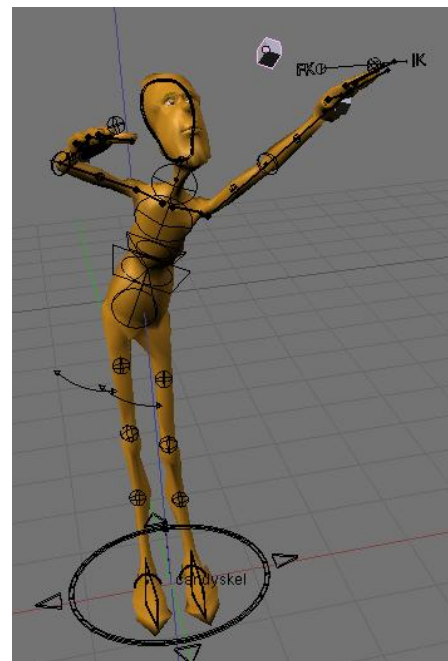


Figure 12: *Avatar visualized on Blender.*

4.3. EyesWeb

The EyesWeb XMI (www.eyesweb.org) is a free software platform [7]. It consists of two main components: a kernel and a graphical user interface (GUI). The GUI manages interaction with the user and provides all features needed to design applications (patches). It allows hand fast development of custom interfaces for use in artistic performances and interactive multimedia installations. The kernel manages real-time data processing and synchronization of multimodal signals. It supports the integration of user-developed plugins: an SDK (Software Development Kit) is provided to sim-

plify the creation of such plugins by means of Microsoft Visual C++. The user-developed plugins, together with the ones provided with EyesWeb are the building blocks that the end user can interconnect to create a patch. The main use of EyesWeb is as a rapid prototyping platform for real time signal analysis. The video side is very well developed.

5. RESULTS

In practice three main applications were developed during this project in order to test the different techniques of analysis and visualization we explored here.

5.1. Gesture-based interface

An intuitive interaction between user's hands and a multimedia wall was achieved by using EyesWeb to analyse the hand motion as described in section 2.1. The hand's position and mode (zoom, rotation, manipulation) were transmitted to the visualization software (Processing) through the OSC (Open Sound Control) network protocol. Data from a database (pictures, movies, texts and sounds) is displayed on a kind of picture board, each picture corresponding to one of the data (see Figure 13).

This pictures can be translated or rotated, user can zoom on one picture, read a video (Figure 15). Many options allow the simple modification of the multimedia wall. For example, it is possible to have one wall by kind of data (see see Figure 14).

Here the videos are in front, behind are the pictures and texts are on the back. In this case, the user's Z position (depth) is sent from EyesWeb to Processing in order to let the user the access to the wall behind he is located in 3D. Figure 15 shows the user in the bottom-left image who manipulates a piece of text in the multimedia wall.

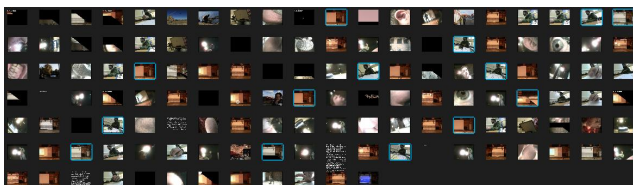


Figure 13: Picture's wall database.

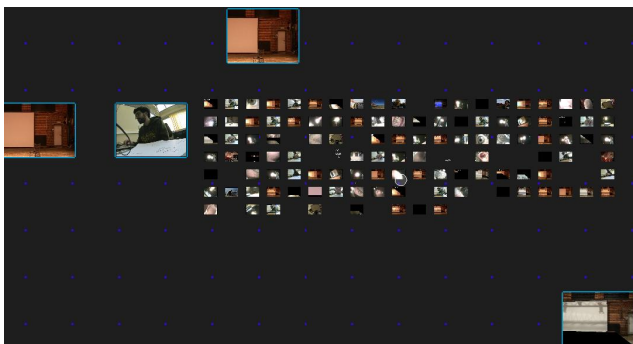


Figure 14: Wall by kind of data.

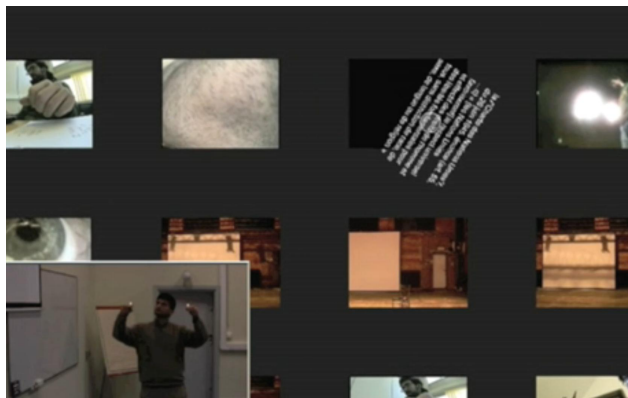


Figure 15: The user manipulates a piece of text in the multimedia wall.

5.2. 2D Sketch recognition

This application provides an augmented reality display. The user draws a sketch by moving his hands. This sketch is automatically recognized among the sketches from the database (section 5.2) and a given synthetic image replaces the sketch at its particular position and size. The user is then able to manipulate the object and change its position through the scene. Figure 17 shows the results of this application. For this application EyesWeb was used both for analysis but also for the visualization.



Figure 16: The user draws a new sketch which be recognized as a house.

5.3. Avatar

The third application detects 6 IR markers located on the user and label them with different body parts (section 2.2.3). Moreover, the user's face is detected and segmented as described in section 3.1 and its face replaces the user head on the final avatar. Two steps are needed in this application: in a first step the calibration of the different body parts is achieved and then real time tracking is performed and an "avatar" is displayed as in Figure 17.



Figure 17: The user moves his body in real time on the screen.

6. CONCLUSIONS

All the techniques we developed and tested in this project led to three highly interactive applications. We measured the difficulty of multi-blob tracking and identification even in simplified video acquisition configurations using IR LEDs. Face tracking works well but mainly when people look to the camera. The SIFT descriptors can provide good results but only on contrasted objects which have a lot of edges and if they are viewed from different angles. GMM-based methods seem to work well in blob recognition by using color information. A problem is in the real time reaction of this algorithm. Hand-made sketches are simple to recognize if they are quite different globally and locally. Correlation of filtered images of the sketches seem to be the best simple matching method for sketch recognition. Blender is an excellent 3D object creation free platform which also contains advanced avatars. If the edit mode offers inverse kinematic mouse-based avatar manipulation, the real time mode does not provide the possibility to dynamically change avatar's position and body parts configuration and thus it is not possible to use it in real time manipulations. Processing is a simple to use (simplified Java) and a practical tool for visualization. The use of OSC protocol let us use another software to achieve image analysis. Finally, some of the three applications achieved here will be used in the next project session (MATRIX project) in order to be enhanced by adding more 3D information and intelligence.

7. ACKNOWLEDGMENTS

numediart is a long-term research program centered on Digital Media Arts, funded by Région Wallonne, Belgium (grant N°716631). Special thanks to Johan Decristophoris for his help in the IR LED setup.

8. REFERENCES

8.1. Scientific references

- [1] Jr. A. Chris Long et al. "Visual similarity of pen gestures". In: *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*. The Hague, The Nether-

lands: ACM, 2000. Pp. 360–367. ISBN: 1-58113-216-6. P.: 150.

- [3] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA: O'Reilly, 2008. P.: 150.
- [4] A. Camurri, M. Ricchetti, and R. Trocca. "EyesWeb - Toward Gesture and Affect Recognition in Dance/Music Interactive Systems". In: *Multimedia Computing and Systems, International Conference on* 1 (1999). P. 9643. ISSN: 1530-2032. P.: 150.
- [5] A. Camurri et al. "Developing multimodal interactive systems with EyesWeb XMI". In: *Proceedings of the 2007 conference on new interfaces for musical expression (NIME07)*. New York, USA. P.: 147.
- [6] A. Dempster, N. Laird, and D. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society B* 39.1 (1977). Pp. 1–38. P.: 151.
- [8] M.-K. Hu. "Visual pattern recognition by moment invariants". In: *Information Theory, IEEE Transactions on* 8.2 (1962). Pp. 179–187. P.: 150.
- [9] D.G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (2004). Pp. 91–110. P.: 151.
- [10] G. McLachlan and D. Peel. *Finite Mixture Models*. Ed. by Wiley Series in Probability and Statistics. 2000. P.: 151.
- [11] K. Mikolajczyk and C. Schmid. "A performance evaluation of local descriptors". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27 (10 2005). Pp. 1615–1630. P.: 152.
- [12] R. Morros et al. "Event recognition for meaningful human-computer interaction in a smart environment". In: *Proceedings of the eNTERFACE'07 Workshop on Multimodal Interfaces*. Istanbul, Turkey 2007. P.: 151.
- [14] T. M. Sezgin and R. Davis. "Sketch recognition in interspersed drawings using time-based graphical models". In: *Computers Graphics* 32.5 (2008). Pp. 500–510. ISSN: 0097-8493. P.: 151.
- [15] P.A. Viola and M.J. Jones. "Robust Real-Time Face Detection". In: *International Journal of Computer Vision* 57 (2 2004). Pp. 137–154. P.: 149.
- [16] C.F.R. Weiman and G. Chaikin. "Logarithmic spiral grids for image processing and display". In: *Computer Graphics and Image Processing* 11 (1979). Pp. 197–226. P.: 152.

8.2. Software and technologies

- [2] "Blender". URL: <http://www.blender.org/>. P.: 153.
- [7] "EyesWeb XMI platform". URL: <http://www.eyesweb.org>. Pp.: 147, 153.
- [13] "Processing". URL: <http://www.processing.org/>. P.: 152.