Telecommun Syst (2006) 32:165–180
DOI 10.1007/s11235-006-9136-4

Video object transmission by means of virtual reality based object extraction

J. P. Bandera · C. Urdiales · B. Segura · F. Sandoval

Abstract This paper presents a new technique to extract objects from a real complex background so that a video sequence can be decomposed into a set of objects as required for object oriented video compression techniques. The proposed method is based on a background subtraction technique. However, instead of using a fixed background, the system relies on predicting one from a previously constructed virtual model of the environment. Thus, camera movements are allowed. These movements are estimated by means of a tracker device. We also present the virtual model construction technique for indoor environments. The method has been successfully tested for several different video sequences including capture errors, partially mapped virtual environments and camera positioning errors. Further work will focus on extending the virtual models not only to environment, but also to objects, and integrating the method in a MPEG4 standard compression system.

Keywords Background subtraction · Virtual models · Video transmission · Video compression

1. Introduction

The capture, storage and transmission of digital video has received a growing attention in the last decades. Particularly, compression of video sequences has currently become a capital question, due to the huge size of high-resolution images and the demand for videoconference systems, in which these sequences must be transmitted in real time using as little bandwidth as possible. The *Moving Pictures Expert Group* (MPEG) publishes the best known and most extended compression standards. MPEG standards basically rely on reducing the redundancy of a sequence by using space and time downsampling mechanisms and brightness and color quantification methods. The last of these standards use object oriented compression techniques, and are called 'second generation standards'. The most popular among them is MPEG4 [8].

J. P. Bandera (✉) · C. Urdiales · B. Segura · F. Sandoval
Departamento de Tecnología Electrónica, Universidad de Málaga
E.T.S. Ingeniería de Telecomunicación, Campus Universitario de Teatinos, s/n
29071 Málaga, España
e-mail: jpbandera@uma.es

The main advantage of second generation standards is that they are qualitative, not quantitative. Thus, audio and video streams are no longer considered as linear streams. Instead, they are decomposed into different objects. Those objects can be separately compressed and adapted to available bandwidth. It is even possible to eliminate some objects or change their attributes (size, position, colors, . . .). MPEG4, though, presents two challenges that contributions in the area need to address. First, the video compression process is computationally expensive, so algorithms must work *off-line* or be efficient enough as to deal with this complexity in real time. Second, it is not specified in the standard how objects can be extracted from a scene [8]. This allows researchers to explore new extraction techniques and incorporate them with MPEG4.

Video object extraction is currently under intense research because it is a hard problem, particularly in real environments where separation between a complex background and a set of objects of interest is ill defined. This paper presents a new technique for object extraction in video-conference applications, where environments are well defined and do not suffer uncontrolled variations. The proposed technique is based on creating a virtual model of the background. It is used to predict what that background might look like at any given time instant. This prediction is used to extract the objects of interest by means of a fast and simple background subtraction technique. The system is briefly described in Section 2. Section 3 presents a short review of background subtraction techniques plus the proposed object extraction algorithm. Section 4 proposes a simple method to build a virtual model of an indoor environment. It is used to estimate the background required by the algorithm proposed in Section 3. Section 5 explains how real objects and virtual background are composed in reception. Section 6 presents several tests and examples. Finally, Section 7 presents conclusions and future work. It is important to note that the proposed application is valid only for video compression in controlled environments that can be roughly modelled by simple image processing techniques before a sequence starts to be captured.

2. System description

Object extraction in video sequences consists of separating the regions of interest from a background, which is defined by a set of homogeneous features. However, in real environments there are no immediate distinctive features in the background. Consequently, the majority of object extraction techniques applied to real images are based on previous knowledge about the background, usually gained by capturing the scene without interesting objects. This *empty* frame can be subtracted from every single frame in the sequence, so that resulting images contain only the objects which are different from the background.

Some background subtraction techniques are described in [10]. Although they seem simple, those algorithms must face very important problems: luminosity variations, appearance of shadows or temporary correspondence between objects and background. These problems provoke errors in subtraction, which are traditionally solved by averaging consecutive frames and dynamically modifying the background [1]. Thus, errors due to illumination changes and noise are reduced. However, this technique leads to static objects loss, because they merge with the background. Mobile masking (e.g. [2,15,16]) avoids these effects, as detected objects are masked and not used to update background. Nevertheless, all mentioned subtraction methods are useless when there are camera movements, because backgrounds change abruptly. There are some methods that solve this problem creating an *a priori* panoramic image that corresponds with the real views taken from the camera in different angles (Rowe and Blake (1995, 1996)). The proposed Background Subtraction Algorithm (BSA) extends this solution creating not an image, but a realistic 3D virtual model of the work environment. This model is used to predict the background for

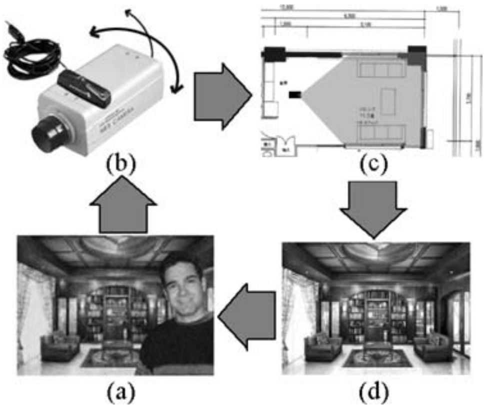

Fig. 1 System description: a) captured frame; b) camera plus tracker device; c) estimated field of view over the environment ground plan; and d) estimated virtual background.

each camera position (Fig. 1), allowing camera translations, distance measurement and addition of complex virtual objects. The method to construct this model is described in section 4. The constructed virtual model is aligned with the real image using a tracker device combined with the camera (Fig. 1.b). When the field of view is correctly determined (Fig. 1.c), a rendered view of the background can be calculated (Fig. 1.d) and subtracted from the real image to get its objects. These objects can be composed in reception with the virtual background to obtain a complete frame (Fig. 1.a).

The proposed algorithm is specially suited for videoconference applications for two reasons: i) these applications are located in controlled and mildly easy to model indoor environments; and ii) the frame rate, for medium-quality videoconferences is low and frames can be processed in real time.

3. Object extraction by background subtraction

BSAs rely on detecting foreground objects in a scene by thresholding the differences between the current frame and either a reference image (background) or the previous image. The background is typically defined by capturing the field of view in absence of objects and it has been traditionally calculated by averaging a large enough number of frames. BSAs are normally improved by updating their statistical description to achieve resistance against changing lighting conditions [10]. When backgrounds are updated, though, it is recommendable to mask the mobile areas in the scene so that they do not have influence on the resulting estimated background [7,15]. After foreground pixels are extracted, further processing is needed to remove noisy pixels and to define foreground objects [6].

3.1. Background estimation

Like most BSAs, the proposed algorithm relies on averaging the frames within a temporal window to obtain an estimation of the background without foreground objects. However, since we use a mobile camera, instead of storing and updating a single background, we operate with several ones which are mapped into a virtual model of the environment. When the camera captures an image, the edges of the walls in that image are extracted from the ground plan of that room and corrected using the vertical edges of the image. These edges are used to separate the image into square regions roughly corresponding to the detected walls. Using this information, the backgrounds

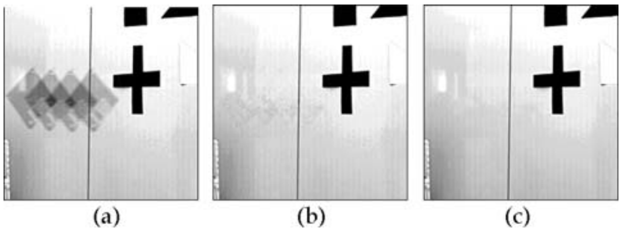
Fig. 2 Background estimation using: a) averages; b) exponential loss; and c) exponential loss plus masked movement.

of all the walls in the captured image can be totally or partially updated. These backgrounds become a bitmap texture for the 3D engine that renders the expected virtual view of the room. In our case, the background of any given wall is calculated as follows:

$$B(x, y, t) = \overline{M(x, y, t)} \cdot [(1-\alpha) \cdot B(x, y, t-1) + \alpha \cdot I(x, y, t)] + M(x, y, t) \cdot B(x, y, t-1) \quad (1)$$

being $B(x, y, t)$ the background estimation at instant t, $B(x, y, t-1)$ the previously estimated background and $I(x, y, t)$ the most recently captured image where that background appears; α is a parameter which is typically used in BSAs to weight the importance of the most recent frame against the stored model. Also, since we do not want mobile objects affecting our background estimation and some parts of the analysed background may not be available in $I(x, y, t)$, we include a binary mask $M(x, y, t)$ defined as follows:

$$M(x, y, t) = 1, \ \textit{if} \ |B(x, y, t-1) - I(x, y, t)| > U \quad (2)$$
$$M(x, y, t) = 0, \ \textit{otherwise}$$

Figure 2 presents an example of the proposed estimation. Figure 2.a presents the simplest option: a background estimated by averaging several frames. In these cases, a mildly fast mobile gets duplicated in the background. A more complex approach is to use an unmasked exponential loss algorithm to estimate the background (Fig. 2.b), so that the mobile does not have so much influence in it. Finally, as Fig. 2.c shows, the best option is clearly to use a mask so that mobiles are not averaged in the background calculation [15].

3.2. Background subtraction

After the expected background is rendered, it is necessary to determine how the input frame and that background are subtracted to return the objects in the scene. The main problem is that objects with the same color as background will be removed from the results. This problem is harder in greyscale sequences, as the system deals with less information in these cases. Therefore, in order to avoid perceptual aliasing as much as possible, it is desirable to work with color video sequences, that presents thrice the information of grey spaces. Consequently, it must be decided which color space to use in the subtraction algorithm. The best known color space is the RGB, where each color is defined by its red, blue and green components. The main drawback of this color space is that it is quite sensitive to illumination changes. Figure 4.a shows an example of background subtraction in the RGB color space for the frame in Fig. 3.a and the background in Fig. 3.b. As depicted in Fig. 4.a, the high sensibility of RGB to illumination changes produces important subtraction errors. These errors affects specially to skin color areas. Other color spaces such as HSI (Hue Saturation Illumination) are also feasible. HSI presents some advantages, the most important one for this application being that illumination is isolated in the field I [9].

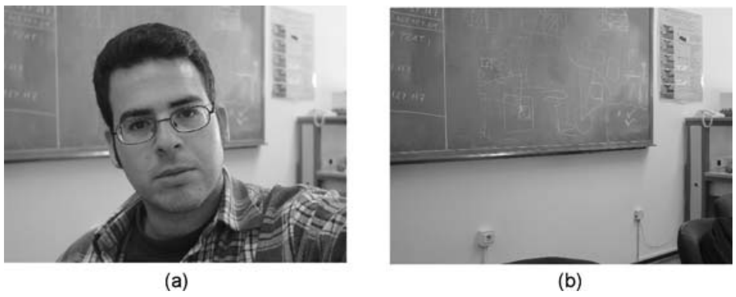

Fig. 3 a) real image; and b) estimated background.

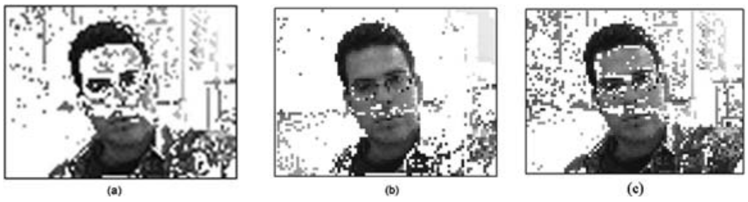

Fig. 4 Background subtraction in: a) RGB; b) HSI; and c) normalized *r*, *g*.

Figure 4.b presents the same experiment in Fig. 4.a but this time working in the HSI color space. It can be noted that the background is better subtracted and less removal noise appears. In this case, skin areas are mostly correctly detected. The main drawback of HSI is that it is strongly non-lineal and that colors are ill defined when they present a low illumination component [9]. These factors make the shirt in the frame too close to the background and, consequently, parts of it are randomly removed. To avoid these problems, a different color space known as normalized *r*, *g* is used. This space is defined by dividing R and G components by R+G+B, that is $r = \mathrm{R}/(\mathrm{R+G+B})$ and $g = \mathrm{G} / (\mathrm{R+G+B})$. The described color space is resistant against illumination changes. Besides, its simplicity makes it suitable for real time processing algorithms. However, colors are ill defined when they are poorly saturated. Thus, we rely on a modified method where poorly saturated pixels are handled in a different way. In order to detect if a given pixel is well saturated or not, its S component is evaluated in the HSI color space. Figure 4.c shows the results using this modified color space. All areas different from the background have been correctly detected. The additional noise introduced with respect to Fig. 4.b, is not relevant as it will be eliminated in the classification process explained in Section 3.3.

Once the correct representation of colors is selected, the system applies the following steps to subtract a virtual background (Fig. 3.b) from a real image (Fig. 3.a):

1. The images are decimated to a fraction of its original size. This increases speed and reduces the effects of capture noise and small variations between real and virtual views. The term 'image' will be later used as 'decimated image' in this section.
2. The two images are compared by computing the color distance pixel by pixel, as described below:
   a. If colors are saturated enough, they are expressed in normalized *r,g* coordinates and the color distance for pixel i in image and background, d_i, is calculated as:

$$d_i = |r_{img}(i) - r_{bg}(i)| + |g_{img}(i) - g_{bg}(i)| \tag{3}$$

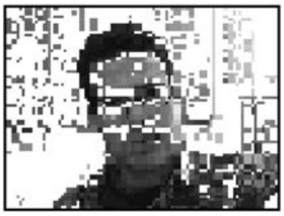

Fig. 5 Background subtraction result.

If d_i gets over a determined threshold, then the two pixels are different.

b. If colors are not saturated enough, three color distances are calculated, one for each RGB component. These distances are simply the difference between components in real and virtual images. If at least one of those distances is over the threshold, the pixels are different.

The saturation level of a color is computed by obtaining its HSI (hue-saturation-intensity) components and considering only the S component [9]. If both real and virtual pixels have a S component over a threshold, the first criterion is chosen. Otherwise, the second criterion is used. Proper values for this saturation threshold are between $(0.035 \cdot S_{max})$ and $(0.045 \cdot S_{max})$, with S component in the range from 0 to S_{max}.

The result of this step can be seen in Fig. 5. The object is extracted, but a lot of noise appears. Further steps remove it and select the interesting area.

3.3. Object extraction

In this stage, pixels remaining after the background subtraction process are grouped into objects. The input instance thus far is simply a set of pixels, where many of them appear because of subtraction noise and errors. The stage has a double goal: i) remaining background pixels are mostly removed; and ii) non background pixels are grouped into one or several objects of interest. The extraction itself is performed in a fast way by means of a simple merge algorithm [12]. Assuming that objects must present at least a given number of connected pixels and that the proposed BSA may provoke "holes" in an object but it does not remove complete parts of it, this algorithm consists of three steps:

1. Pixels remaining after subtraction are grouped in classes. Groups are created by a simple merging algorithm that preserves connectivity [12].
2. The areas in pixels of all classes are computed. In videoconference applications the object of interest, i.e. the speaker, should occupy a great portion of the image. Thus, classes with area under a threshold are discarded. The remaining ones are objects of interest. This step removes noise due to illumination changes, shadows and virtual model defects.
3. Remaining classes are dilated to remove holes caused by subtraction errors.

Fig. 6 shows the steps of the proposed algorithm, applied to image in Fig. 5. Figure 6.a is the difference image obtained in step 2. A main area can be seen, but also other irregular areas. All of them are potential objects, as Fig. 6.b shows. Classes with area under a predefined threshold of 600 pixels are discarded (Fig. 6.c). Thus, there is only one class left. This class is dilated to remove noisy holes. Figure 6.d shows the final object that will be transmitted. The rest of the image has been marked in an homogeneous color and will be filled with a local model in reception, as described in Section 5.

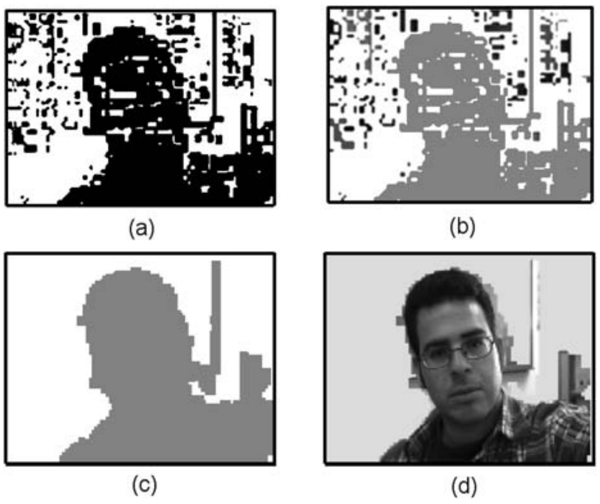

Fig. 6 Object extraction algorithm: a) background subtraction result; b) region segmentation; c) elimination of regions with area under 600 pixels and dilation; and d) masked frame.

Through these steps, the proposed BSA was required to process all decimated image pixels only one time, to compare them with the virtual view. The normalized *r,g* color space allows us to use a very fast distance measurement. After comparison, only marked pixels are classified and dilated. Complexity is proportional to (N/d^2), where N is the number of pixels in the input image, and d the decimation ratio.

4. Creation of virtual background

All background subtraction algorithms (BSAs) rely on a background model which should be equal to the field of view in absence of foreground objects. If the camera moves, a BSA needs an estimated background for each position of the camera. As previously commented, in the proposed scheme such a background is extracted from a virtual model of the environment, where the position of the camera is calculated by means of a tracker device. This section proposes a method to create a rough model of the environment in a simple and fast way. It is important to note that the model does not necessarily need to be precise for background extraction because a post processing technique has been proposed to remove most extraction errors. Also, it is only necessary to provide information about the areas that are expected to be captured by the camera. The simplest possible virtual background can be constructed by means of a ground plan of the room where video is captured plus a set of their textures. Both things can be either available *a priori* or obtained by moving the camera around the room to acquire the model before the sequence starts to be captured. In the first case, the virtual room can be constructed by means of any 3D construction utility and the only problem is to determine the initial position of the camera in the model. In the second case, it is necessary to use image processing techniques to automate the construction process. In addition, the model is constructed with respect to the coordinate system of the camera, so its position is implicitly known. As we usually will not have an *a priori* 3D model of the environment, we consider the second option, in which the environment has to be built. In this paper, we propose a simple method to supervisedly extract all necessary information to do it from a video sequence.

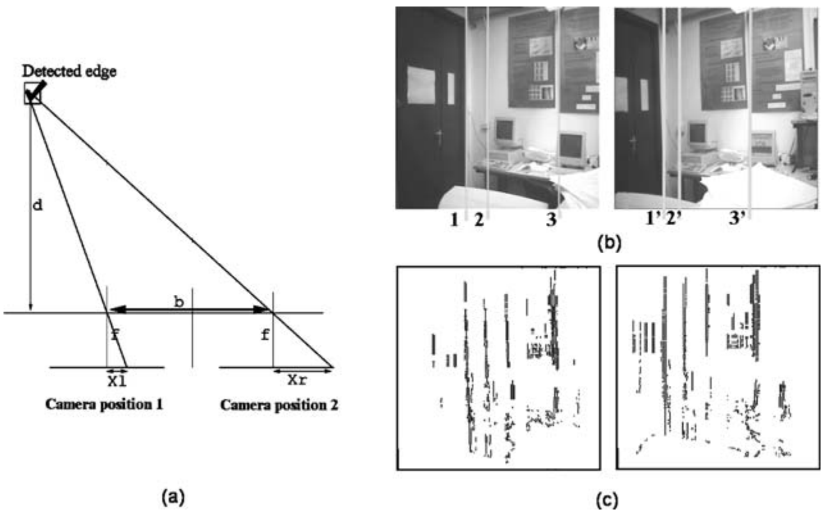

Fig. 7 2D model construction: a) distance estimation for vergency zero; b) wall fit; and c) edge detection and matching.

4.1. Ground plan extraction

A rough 2D model of the environment can be constructed by means of landmark detection, matching and triangulation. Landmarks should be features in the environment that can be easily detected. Obviously, 3D objects can be used as landmarks, but they are difficult to extract from complex scenarios. In this paper landmarks are the longest vertical edges of the scene which, in structured environments, tend to correspond to the intersection of two adjacent walls or to tall pieces of furniture. These edges can be easily extracted from an image in a fast way [12]. In order to estimate distances, at least two images where the same landmark is located are required. Since the proposed system works with a single camera, the user needs to move that camera to capture a stereo pair. If the vergency of that stereo pair is kept equal to zero by keeping the same orientation of the camera when it moves (Fig. 7.a), the distance d to a vertical edge detected by the camera in both images can be simply calculated as:

$$d = b\frac{f}{X_r - X_l} \tag{4}$$

being b the distance between the two camera positions, f the focal distance of the camera and X_r and X_l the position where the edge is detected in the frames corresponding to the right and left image of the stereo pair respectively [3]. To use equation 4, edges detected in both images of the pair must be matched. This task is not hard as long as only the longest vertical edges detected are evaluated, and there are only a few of them in an image. In order to test if two edges captured in different images are the same, each edge is assigned a vector whose component i is equal to the color of edge pixel i, starting at the top of the image. Consequently, the length of the vector is equal to the height of the image. Each edge in the left image is compared to the edges in the right image from left to right by means of least squares to obtain the best match. If a match is not good enough, the edge is discarded. Two or more consecutive aligned edges define a wall (Fig. 7.b)

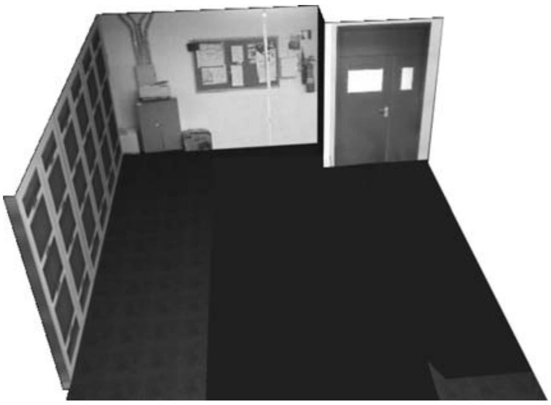
Fig. 8 Wall based virtual model with mapped textures.

Fig. 7.c shows a simple example of vertical edge detection for a pair of images of 256×256 pixels captured with a single camera with focal distance equal to 7 mm, where the two camera positions were separated 10 cms. All three edges are correctly matched and detected as aligned. Thus, they define a wall (Fig 7.b). It can be noted that, in fact, edge 1 does not belong to the detected wall. However, the system does not provide enough resolution to detect structures so small. It is also necessary to note that, in this particular case, matches were relatively simple. This is not always the case, though. Wrong matches provoke wrong distance estimations. If a measured distance is not coherent with the rest, it is discarded. Usually, not all edges required to build the walls of a room can be captured by a single pair of images. Therefore, the user must turn the camera around, trying to keep it approximately at the initial position so that the coordinate system does not significantly change, until all required edges are captured.

4.2. Texture estimation

Once a ground plan is available, our 3D model of the environment is simply built by rising up the walls and mapping them. To achieve a fine 3D model of the environment, it would be necessary to extract the distances not only to walls but to all relevant objects of the scene to create a textureless 3D range model of the environment. However, this process would be complex and computationally expensive. Therefore, the proposed 3D virtual model simply presents walls and all 3D objects in the room are projected in these walls (Fig. 8). It must be noted that this model construction technique is very simple and, consequently, prone to errors, but it will be shown in the results section that the obtained model is adequate for background estimation purposes.

Walls need to be textured in order to achieve a perceptually realistic virtual model of the environment. Our system uses a graphic engine to represent this textured virtual environment. We have chosen Genesis3D [4] engine because of its flexibility, efficiency, open code and free distribution. Once the tool to create the 3D model is available, the textures are mapped in the virtual environment as detailed below:

1. The floor and ceiling textures are captured by pointing the camera up and down. Then, the user pans the camera to capture the different textures of the environment. The following steps are required for each capture.
2. Image projections of the theoretically visible edges of the virtual environment are obtained (Fig. 9). This operation is made by using the camera orientation and parameters, and simple and fast trigonometric relations. Once obtained, these image columns in which edges are projected

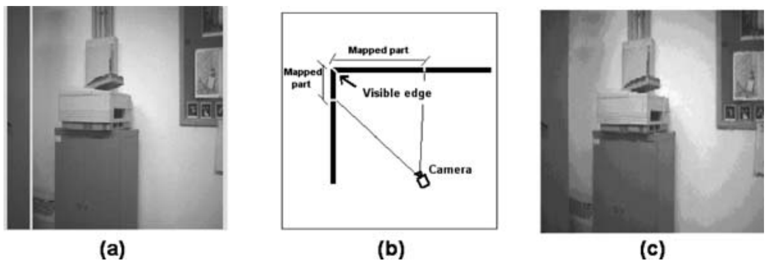

Fig. 9 a) projections of virtual edges over the real image; b) ground plan of a) showing camera, partially mapped walls and visible edges; and c) obtained virtual model.

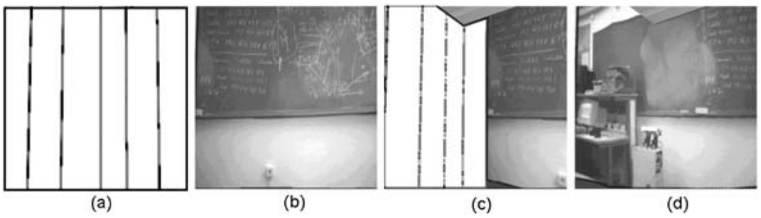

Fig. 10 Texture mapping: a) textureless virtual model view; b) real textures applied to a); c) partially mapped model view; and d) full mapped model view.

(Fig. 9.a) are used in the real image to delimite the textures which have to be mapped in each wall.

3. Using the ground plan (Fig. 9.b), the distances from each visible edge to the camera are easily obtained and stored. These distances are used to correct perspective deformations.
4. The fragments of the real image that must be mapped in each visible wall are cut by using edges obtained in step 2. Using the depth information, perspective deformation is corrected by a fast Affine Texture Mapping algorithm proposed in [5]. This method uses bilinear interpolation to render by calculating the map points at either end of a scan line and linearly stretching the texture map texels between them. This technique is very fast and it works correctly when there is a small angle between the view direction and the surface normal of the rendered object. However, when the angle increases, textures start to look distorted. This is particularly noticeable when the object is close to the viewpoint. Also, errors accumulate along the scan line during rendering. Therefore, mapping accuracy is largely dependent on the length of each scan-line. To avoid this problem, a scan line subdivision technique is used. Scan-line subdivision compensates these errors by splitting each scan-line into smaller segments and eliminating the error as it moves along.
5. Fragments obtained in previous step are finally mapped on the corresponding walls (Fig. 9.c). If a wall is only partially visible, then only its visible part is mapped.

Mapping results are shown to user during the process so that the user can keep on panning and mapping until he is satisfied with the virtual environment, as Fig. 10 shows. Therefore, the user supervises the creation of the environment and it is his decision to make it more or less complex. If the camera is not supposed to move very much during the videoconference, then the virtual model can be a single textured wall behind the objects of interest. This approximation, similar to panoramic backgrounds described in Rowe and Blake (1995, 1996), is more exact as distance between background and objects grows. If the camera is going to

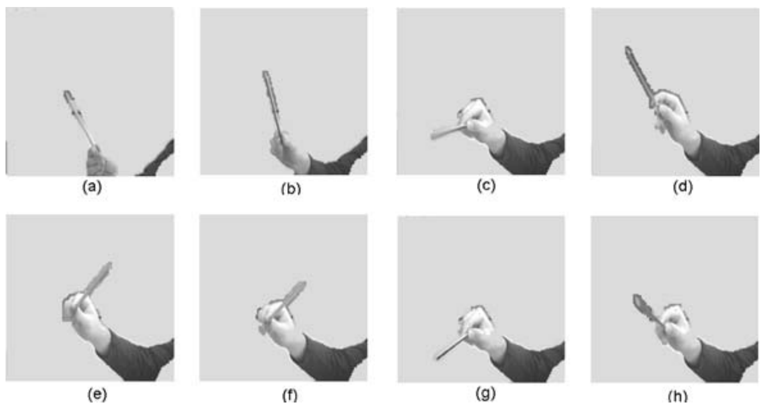

Fig. 11 Results for scenario 1: a–h) objects of interest extracted from different frames.

move very much during the videoconference, or background is closer to interesting objects, a more precise model is required. The user should confirm, in this case, that edges and distances are accurate and textures are mapped correctly in order to avoid transmission of false objects.

Despite user supervision, perspective and barrel distortion, strong illumination changes and distance errors will produce discrepancies between real and virtual backgrounds. It can be observed, in upper part of Fig. 10.d, that there are mapping errors. The effect of these errors is commented in the results section. It is important to note that textures captured *a priori* are not static. So, after initial mapping, textures are updated as described before (Section 3.1) to make the model as resistant as possible to illumination changes.

5. Image composition

The transmitter in a videoconference scenario only sends objects of interest in each frame. To build a complete image in reception, these objects are composed with a background. If the virtual model can be sent before videoconference starts, the transmitter only has to send the camera orientation plus the objects for each frame. The receptor uses the information to predict the virtual background. If no virtual environment is received, objects can be composed with a local virtual background or with a bitmap. Composition takes two steps:

1. The received masked frame is used to construct a special bitmap object of Genesis3D called HUD. All masked pixels of the frame are set to transparent in the HUD.
2. The HUD is overimposed to the rendered view of the virtual environment. The received orientation information allows correspondence between the real background in transmission and the virtual background in reception.

6. Experiments

This section describes some results obtained when the proposed method is used with a sequence captured with a low cost videoconference camera. Two different cameras were used: a CXX-Z11E comercial webcam, and a ALM-2452M camera, that produces more barrel distortion as it

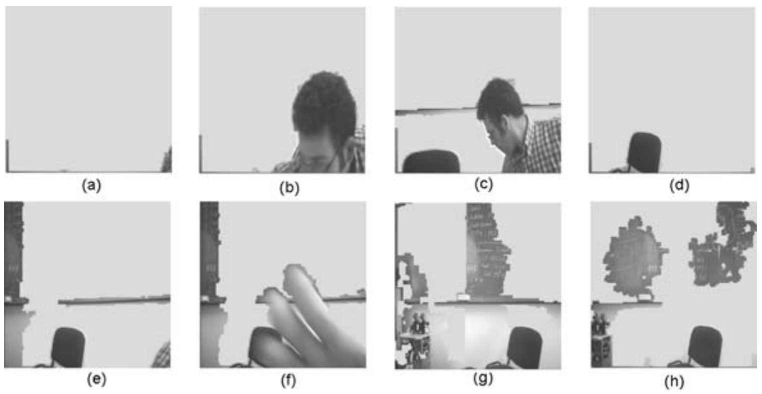

Fig. 12 Results for scenario 2: a–h) objects of interest extracted from different frames.

is smaller. This camera was used as a worst case simulation. The selected camera was combined with a tracker device (InterSense InterTrax^2) that provides 3D rotation angles. The receiver needs these *pan*, *tilt*, *yaw* angles as inputs to align real and virtual environment views. Therefore, information sent for each frame will contain not only extracted objects, but also camera rotation angles. The camera can be freely rotated and it also can be slightly translated, but no major translations are allowed, as the tracker system is unable to detect them. If we worked instead with two cameras to capture stereo pairs, it would be feasible to freely move those cameras because distances could be calculated at any given time but these experiments have been conducted using a single one. The main problem of the used camera was its barrel distortion and its automatic exposure system. The barrel distortion provokes differences between the captured image and the virtual model depending on perspective. The automatic exposure system provokes severe lighting changes depending on the objects of interest in the scene. All those problems are usual in average videoconference systems. The proposed algorithm is capable of adapting to them, as proven in experiments below.

There are many BSAs, but none relies on a virtual background for prediction. The method proposed in [14] tracks curves, and, consequently, object extraction is not guaranteed. Hence, no comparison can be established. In this paper we test the system under different circumstances to prove its validity.

A first set of tests was performed to test the efficiency of the object extraction process. Figure 11 shows a simple example of object extraction from a video sequence in a laboratory modelled as suggested in section 4 (Fig. 8). In this case, all frames are captured in a short time and no severe illumination changes have happened. Also, these frames have been captured when the camera was approximately orthogonal to the back wall and, consequently, perspective distortions were not serious. It can be observed that the hand with the pen is correctly extracted in all cases. However, part of the hand is removed in some frames. This occurs because the background model presents a color similar to the hand at the position occupied by that hand in those frames. As these pixels will be filled with background color in reception, and this color is similar to object color, these segmentation errors will not be appreciated. In fact, they suppose an improvement in the compression rate. Finally, it can be appreciated that in all frames the bottom right corner of the objects is removed. This occurs because in the background there is a chair at that position presenting the same color as the shirt of the user. That chair can be observed in a second experiment in Fig. 12.a.

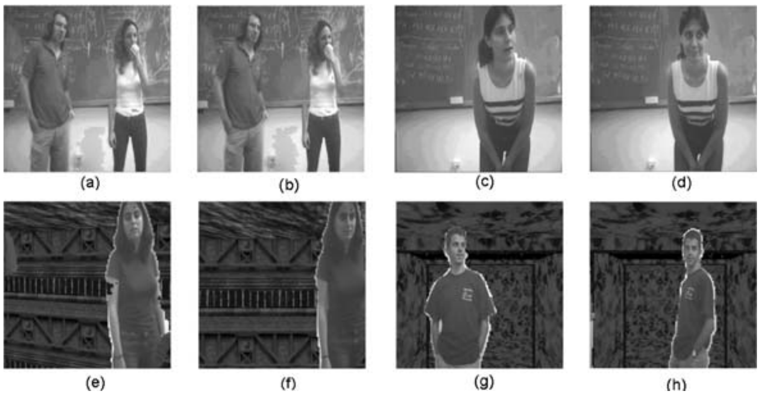

Fig. 13 Composition of extracted objects with: a–d) the virtual model used in BSA; and e–h) a different virtual model.

Figure 12 presents a more complex scenario where the camera moves significantly, the field of view is more affected by perspective errors, lightning changes appear and more than one object of interest appear in the scene. Figure 12.a presents a frame after subtraction where no object of interest are in the scene. As expected, no object appears in the frame as well except for a small part of a chair in the bottom right corner, which was not originally mapped into the virtual model. Figures 12.b and c show a person at different distances from the camera changing the position of the chair in Fig. 12.a. It can be observed in Figs. 12.d to h that the chair becomes an object of interest as long as it is in the field of view. It is also interesting to note that an horizontal line is detected as an object of interest in Fig. 12.c. This occurs because, due to the barrel distortion of the camera, the curvature of that horizontal line is different depending on the relative position of the camera. It can be observed in Fig. 12.d that the point of view of the camera has significantly changed. However, the perspective distortion still does not affect the results. If the camera is turned a bit more, though (Figs. 12.e–f), part of the scene is not correctly subtracted because of perspective errors. This only produces a drop in the efficiency. It is important to note that these errors are not so significant when they are not as close to the camera as in this case. If the camera is turned still a bit more, subtraction errors become larger (Fig. 12.g–h). These new errors are basically due to illumination changes in the room. The background model is not adapted to these changes because the camera was not capturing those areas before. However, it can be observed in Fig. 12.h that the background soon starts to be adapted to the new conditions and errors tend to disappear after a few frames.

A second set of tests was performed to check how the proposed system works in a videoconference application. In this case, only objects of interest are transmitted, but they can be composed in reception either with the virtual environment used in the BSA or, if that model has not been sent to the receptor, with any other available virtual environment. To correctly compose these objects with a virtual environment, it is only necessary to add the camera position to each transmitted frame. The receptor renders the view for the available virtual model regarding the received position information and simply composes the received objects with this view. Figures 13.a to d show an example of composition when the virtual model used in BSA is sent to the receiver at the beginning of the videoconference. It can be appreciated that no major errors can be perceived in the frames. Further examination shows that part of the left arm of the person in the right of the scene is removed (Fig. 13.b). Also, a small part of the white wall is not correctly subtracted and, hence, it is transmitted as an object of interest. Figure 13.e–h show several examples where

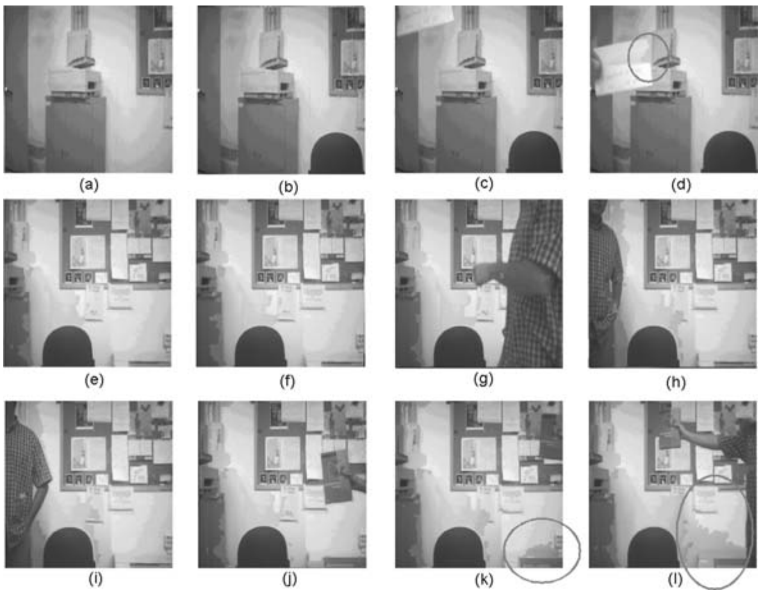

Fig. 14 Sequence composition in the environment in Fig. 8.

objects of interest are composed with an artificial virtual background. It can be observed that the objects have been correctly detected even though the position of the camera has changed, as can be observed in the background different perspectives. In this case, the borders of the objects of interest seems to be worse defined because the contrast with the artificial background is larger. In fact, this effect, produced by dilation, is present in all frames, but it can not be appreciated when objects blend perfectly with the virtual background (Figs. 13.a–d).

Figure 14 presents twelve different frames where objects of interest are composed with the background in Fig. 8. In these frames, the camera moves significantly, different objects of interest may appear at different positions and there are illumination changes. In all cases, objects of interest are correctly transmitted despite their different sizes, positions and distances to the camera. In some cases, though, parts of these objects are removed (Fig. 14.d) or part of the background is incorrectly transmitted because of shadow effects (Figs. 14.k–l). These errors mainly affect the compression rate of the process, because transmitted areas are larger. Effects are barely noticed in reception, though, as long as objects are composed with the virtual background used for BSA.

The efficiency of the method has been tested by compressing some sequences processed by the proposed BSA. The compress ratios have been compared with the ones obtained when the same original sequence is compressed without using the proposed algorithm. The same MPEG4 Codec and options are used in both cases.

Results show that the system provides an average of 50–60% reduction in the size of the resulting sequence. This average reduction is highly dependant on particular conditions of environment and system. Thus, it is not possible to offer more accurate compression ratios. The efficiency of the method depends on the amplitude and velocity of camera movements, feasibility of landmarks to construct the virtual model, and camera specifications. But objects in the scene are the main influence. The proposed BSA can produce a final size of less than 25% with respect to direct compression, if objects are small and appear only for short periods. But in situations in which there are few background pixels (e.g. a user very close to the camera), the use of this method will not improve very much the final compression ratio. In any case, it is still useful to efficiently detect objects in the scene. The system runs at typical videoconference frame rates on a medium PC (Pentium III at 866 MHz, with 512 MBytes of RAM and an AGP-4x 65 MBytes graphic card). The decimation step can be used to change the computational complexity of the method. This makes the system scalable to work at the same rates with larger images or to increase velocity if necessary, although a bigger decimation ratio produces less accurate results.

7. Conclusions and future work

This paper has presented a new object extraction method based on a BSA. Our BSA uses a virtual model of the environment to predict the background, so that camera movements are allowed. The system has been tested for real video sequences captured with low cost video-conference cameras and a commercial tracker device and provides an average of 50–60% reduction in sequences compressed with MPEG4 when compared to non processed compressed sequences.

The most important advantages of the proposed method are its low cost and versatility. In addition, it achieves the typical frame rates of average videoconferences based on a standard PC and a webcam. The main disadvantages of the system are its sensitivity to image deformations in low quality cameras and false extraction produced by automatic gain compensation mechanisms. Future work will address the addition of translation detection, tracking landmarks to compute camera displacements. Also, the system will be improved by incorporating a newly developed segmentation technique [11] to extract objects in real time, at 20–25 frames per second. These objects will be characterized in shape and color so that they can not change too much between consecutive frames. Finally, the proposed system is going to be combined with a view based 3D object recognition algorithm to recognize and virtually model static foreground objects in the environment. This way the environment could be described in terms of background, static objects and mobiles.

Acknowledgements This work has been partially granted by the Spanish Ministerio de Educación y Ciencia (MEC) and FEDER funds Project no. TIN2004-05961.

References

1. Collins, R., Lipton, A. and Kanade, T., A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University (2000).
2. Elgammal, A.M., Harwood, D. and Davis, L.S., Non-parametric model for background subtraction,*Proc. of the 6th European Conference on Computer Vision*, 2 (Ireland, 2000) 751–767.
3. Faugueras, O., Three-dimensional computer vision: A geometric viewpoint. Cambridge: The MIT Press (1993).
4. Genesis3D Open Source Engine., In http://www.genesis3d.com (1998).
5. Heckbert, P.S., Survey of texture mapping, *IEEE Computer Graphics & Applications* 6(11) (1986) 56–67.
6. Haritaoglu, I., Harwood, D. and Davis, L.S., Real-time surveillance of people and their activities, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22(8) (2000) 809–822.
7. Huwer, S. and Niemann, H., Adaptative change detection for real-time surveillance applications, *Proc. of the 3rd IEEE Int. Workshop on Visual Surveillance* (VS), (Dublin, Ireland, 2000) 37–45.
8. Koenen, R., Mpeg-4 - multimedia for our time, *IEEE Spectrum* 36(2) (1999) 26–33.
9. Lin, X. and Chen, S., Color image segmentation using modified hsi system for road following,*Proc. of the IEEE Conf. on Robotics and Automation*, 1998–2003, (Sacramento, California, 1991).

10. McKenna, S.J., Jabri, S., Duric, Z., Rosenfeld, A. and Wechsler, H., Tracking groups of people, Computer Vision and Image Understanding 80 (2000) 42–56.
11. Marfil, R., Rodríguez, J.A., Bandera, A. and Sandoval, F., Bounded irregular pyramid: a new structure for color image segmentation, *Pattern Recognition* 37(3) (2004) 623–626.
12. Pitas, I., Digital Image Processing Algorithms. Prentice Hall (1993).
13. Rowe, S. and Blake, A., Statistical background modelling for tracking with a virtual camera, *Proc. of the 6th British Machine Vision Conference* 2, (Birmingham, 1995) 423–432.
14. Rowe, S. and Blake, A., Statistical mosaics for tracking, *Image and Vision Computing* 14(8) (1996) 549–564.
15. Rodríguez, J.A., Urdiales, C., Camacho, P. and Sandoval, F., Detección jerárquica de móviles sobre geometrías de fóvea adaptativa, Revista Electrónica de Visión por Computador 3, in ISSN:1575-5258 (2002).
16. Stauffer, C. and Grimson, W.E.L., Adaptive background mixture models for real-time tracking, *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Fort Collins, CO, USA) 2 (1999) 246–252.