Webcam motion detection and tracking interfaces for immersive embodied experiences

Joana Martins
joana.barreto.martins@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2016

Abstract

Video tracking is an important area of Computer Vision research. It has tremendous potential regarding the improvement of human-computer interaction. Thus, it contributes to the most different fields like security and surveillance, video communication and compression, augmented reality, technological control, medical imaging, video editing, among others. In order to track an object in video, it is usual to follow four steps: first detect the moving region, then apply object classification techniques in that area of the frame to recognize what object has moved and, finally, proceed with the prediction of the next movement by that object. In this thesis we present the methods that we consider to be more relevant for each step of this quest. The work is undertaken in the context of a collaborative dance-technology project Senses Places, which fosters mixed reality performances that involve participants from all over the world. Meeting physically and/or virtually, in a common environment, they engage kinaesthetically in new shared embodied mediated experiences towards expanding awareness to cross geographical, cultural, disciplinary, artistic and human boundaries. We contribute to the project with a mixed reality interface that recognizes the human movements in front of a webcam in real time and translates them to an online virtual world where an avatar will respond, according to the choreographic approach. In this dissertation we explain the process followed to achieve the best interface, including the tracking techniques used, and the virtual world scripts and animations developed.

Keywords: Video tracking, Mixed reality webcam interface, Senses Places, Computer vision

1. Introduction

Movement recognition and object tracking research is an area of study in high demand. They grant machines with the sense of vision, which increases their similarities with humans and improve their interactions. The progress of this technology arouses great interest, since there are endless applications [5].

This technology extends to multiple areas like medicine, military, surveillance and entertainment, but in this work we will be focused on the performing arts applications. These software and systems have contributed to bring closer the physical and virtual worlds, allowing the creation of virtual dance partners, which generates new and transcending choreographies. As a consequence it provides the emergence of new styles and forms of art, revolutionizing concepts and aesthetics [16].

1.1. Objectives

The work developed had the fundamental goal of creating an object recognition software to serve the project Senses Places. Senses Places is a dance-technology collaborative project that aims to generate a playful mixed reality performance environment for audience participation [17]. The interface was designed to track the dancer's movements and connect them to the movements of an avatar in the Multi-User Virtual Environment (MUVE), Second Life (SL), providing a deeper connection between the physical and virtual embodiments and realities. In order to be able to develop this interface we studied different techniques of object tracking via video, that are also presented in this dissertation, providing a global view of the main algorithms in use.

The methods presented are for a single common webcam, not a stereo camera, nor any other type of camera. These recognition and tracking methods do not require the use of sensors or special backgrounds. Besides these limitations, we are also subjected to the following typical difficulties given by the complexity of the problem like noise in the images, complex object motion and nonridged or the articulated nature of objects.

With all these constraints in mind, it was clear that there is no single solution that applies to all problems and applications. We approached existing techniques, referring their advantages and disadvantages. This way we could make informed choices, concerning our final goal that is the design of a webcam interface to track dance movements.

2. Motion Detection and Tracking Algorithms

Human tracking is a complex issue because of its body shape, articulation, fast motion and clothing. To further complicate the problem, the lighting and background conditions and the noise may have a great influence in several techniques [13]. To achieve our final goal, these circumstances require a pre-processing of the video before executing any other steps.

After pre-processing, we need to detect and categorize the object, before being able to track it [12]. We have to apply object detection techniques to detect objects of interest in the video sequence and to cluster pixels corresponding to these objects. Then, we have to find ways to represent them in order to allow differentiation from all the others. Objects can be classified as different things, e.g. vehicles, body parts, birds, floating clouds, swaying trees, among others. After that we are able to track our objects more easily.

2.1. Pre-Processing

Once we get the frames from the video, it is required to use pre-processing techniques to facilitate their analysis. The most common techniques involve enhancement, restoration and processing of the colour, compression and noise removal. Pre-processing must be done with caution to avoid loss of information.

2.2. Motion Detection

At the beginning of this process of object tracking, we have to detect the objects in the video frames. Most methods described in this section are motion recognition algorithms, since they are typically the primary source of information and the cheapest, in terms of computational memory. In this section we present the frame difference algorithm, the background subtraction algorithms and the optical flow.

2.2.1 Frame Difference Algorithm

This technique starts from the assumption that the background is static and compares the pixels of the frames captured in very small time intervals (Δ_t). If the absolute difference comes to a value above a pre-defined threshold (Γ), it means that the pixel has changed and therefore there was movement.

This method can be described by the following formula:

$$I_i = \begin{cases} 1, & \text{if } \mid f_t - f_{t-1} \mid \geq \Gamma \\ 0, & \text{if } \mid f_t - f_{t-1} \mid < \Gamma \end{cases}, \qquad (1)$$

where I_i is the binary result of the absolute difference of a pixel between two consecutive frames (f_t and f_{t-1}).

The differences of all the frame pixels give us a collection of binary numbers that, ultimately, will translate the movement [14]. This method allows a good movement recognition, it is easy to implement and does not require a lot of memory space. However, it considers the background static.

2.2.2 Background Subtraction Algorithms

A common approach to identify the moving objects is background subtraction, where each pixel from the video frame is compared against a reference image of the background. When pixels in the current frame differ significantly from the reference image, it is considered that movement occurred. A very plain approach to this method is to detect the foreground objects as the difference between the current frame and an image of the scene' static background, similar to the technique addressed before. This approach is very simplistic and it probably won't give very good results because it does not adapt the background image to any change that may occur.

The median filter technique overcomes that problem by adapting the background image by calculating the average of the previous frames. The preceding frames are saved and the background is calculated as the median of the buffered frames. Then, it will be processed as seen before, the background model is subtracted from the current frame and thresholded to determine the binary image of the foreground pixels. Unfortunately, it requires a large amount of memory to store and process many frames. For that reason, it was developed **the approximate median method**. In this scheme, if a pixel in the current frame has a value higher than the corresponding background pixel, that background pixel is incremented by 1, and if it is lower that pixel from the background model is decremented by 1. This estimate eventually converges to a value where half the input pixel is greater than the background and half is less (approximately the median). The principal problem with this filter is its slow recovery from changes in the background [1].

One of the most used models of background subtraction is the pixel wise mixture of Gaussians (MoG), proposed by Stauffer and Grimson, [15], because it presents a good compromise between robustness versus computation time and memory requirements. In this technique the background is not considered a frame of values. Instead, the background model is parametric and the model parameters can be adaptively updated without keeping a large buffer of frames. MoG keeps a probability density function for each pixel, in a way that each pixel is characterized by a mixture of n weighted Gaussian distributions, with a certain mean and standard deviation. A background pixel has a high weight and a weak variance, since the background is usually more present than the moving objects and it is practically constant. There are normally three to five Gaussian distributions for each pixel, depending on memory limitations. The parameters can be initialized, resorting to median filtering and then updated. After initializing the parameters, we proceed to the foreground detection by ordering the n Gaussians of a pixel by a criterion ratio. The background model is formed by the first Gaussian distributions that exceed a specific threshold. The remain components are considered to be the foreground distributions. Every pixel is compared to these distributions to determine if they belong to the foreground or the background [15]. This comparison is done based on the Mahalanobis distance. At every moment we need to update the components of each pixel. This technique cannot deal with sudden and drastic lighting changes.

2.2.3 Optical Flow

Optical flow is the pattern of apparent motion of objects, surfaces and edges in a visual scene, generated by the relative motion between an observer and the scene [12]. The optical flow method calculates the optical flow field and groups different types of movements, according to the optical flow distribution characteristics of the image. This technique seizes all the movement information and detects the moving object easily, however, it is demanding in terms of computation time and memory requirements. In the definition of the optical flow problem, we consider that in the frame obtained at the instant t_1 we get a certain point (x_1, y_1) that corresponds to the point (x_2, y_2) in the frame of instant t_2. It is assumed that the distance between both points (or pixels) is small. Considering a linear local deformation in a sequence of images, we are able to translate the optical flow problem into the following formula:

$$I\Big(x(t), y(t), t\Big) = I\Big(x(t) + \Delta x, y(t) + \Delta y, t + \Delta t\Big). \tag{2}$$

By introducing in the formula the velocities for both coordinates ($u \equiv \frac{\Delta x}{\Delta t}$ and $v \equiv \frac{\Delta y}{\Delta t}$) and substituting the discrete changes Δx, Δy and Δt with their infinitesimal counterparts dx, dy and dt, we get: $I\Big(x(t)+udt, y(t)+vdt, t+dt\Big) - I\Big(x(t), y(t), t\Big) = 0.$ Considering I a differentiable function, we can expand the left-hand side of this formula with a Taylor series and by denoting the partial derivatives of I as I_x, I_y and I_t, and $\mathrm{v} = (u, v)$ for the flow vector, we obtain the so called image brightness constancy equation:

$$\begin{aligned} & I_x u + I_y v + I_t = 0 \\ & or \\ & \nabla I \cdot \mathrm{v} + I_t = 0. \end{aligned} \tag{3}$$

It conveys that the apparent brightness of moving objects remains constant. As we can notice, this equation has two unknown variables (u and v), which leads to an under constraint problem. Here we describe the two most common techniques used for optical flow computation: Horn and Schunck [7] and Lucas and Kanade [10]. These two methods are the most commonly used because they offer accurate results for almost every type of movement.

The Horn and Schuck's method combines the optical flow constraint equation (3) with a smoothness constraint. It states that the optical flow field must vary smoothly and have few discontinuities. One way to convey this new constraint is by limiting the difference between the flow velocity (v) at a point and the average velocity over a small neighbourhood containing that point, which is equivalent to minimize the sum of the squares of the derivatives in x and y. If the flow velocity is significantly different compared to its neighbours, the gradient along either x or y will be large. The sum of this constraint with the optical flow constraint presents the function to be minimized:

$$\begin{aligned} E^2 = \iint & (\nabla I . \mathrm{v} + I_t)^2 + \\ & w^2 \bigg(\Big(\frac{\partial u}{\partial x}\Big)^2 + \Big(\frac{\partial u}{\partial y}\Big)^2 + \Big(\frac{\partial v}{\partial x}\Big)^2 + \Big(\frac{\partial v}{\partial y}\Big)^2 \bigg) dxdy, \end{aligned} \tag{4}$$

where w^2 symbolizes a constant weighting factor. Larger values of w lead to a smoother flow. To minimize this function, it is proposed to use the multi-dimensional Euler-Lagrange equations, giving us the system:

$$\begin{aligned} (w^2 + I_x^2 + I_y^2)(u - \overline{u}) &= -I_x(I_x\overline{u} + I_y\overline{v} + I_t) \\ (w^2 + I_x^2 + I_y^2)(v - \overline{v}) &= -I_y(I_x\overline{u} + I_y\overline{v} + I_t), \end{aligned} \tag{5}$$

where $\overline{u}$ is the average of the component u and $\overline{v}$ is the average of the component v.To solve this system of equations, Horn and Schunck proposed an iterative solution [7]:

$$\begin{aligned} u^{n+1} &= \overline{u}^n - \frac{I_x(I_x\overline{u}^n + I_y\overline{v}^n + I_t)}{w^2 + I_x^2 + I_y^2} \\ v^{n+1} &= \overline{v}^n - \frac{I_y(I_x\overline{u}^n + I_y\overline{v}^n + I_t)}{w^2 + I_x^2 + I_y^2} \end{aligned}. \tag{6}$$

This approach, often run for 100 iterations, makes the propagation of the optical flow vectors happen in a way that will eventually fill in regions of the field that correspond to homogeneous regions in the image [7].

Lucas and Kanade differential technique is one of the most popular techniques for optical flow computation. This method solves the aperture problem by assuming that the velocity vector will be similar in a small neighbourhood surrounding the pixel (V). This is approached through a weighted least-squares problem, where we have to minimize the error function below:

$$E_{\mathrm{v}} = \sum_{p \in V} W^2(p)\Big(\nabla I(p) \cdot \mathrm{v} + I_t(p)\Big)^2, \quad (7)$$

with p being the neighbour pixel and $W(p)$ the weight associated to that pixel. The farther the neighbour pixel is from the given pixel, the smaller the associated weight should be. The weights work in order to diminish the importance of distant neighbours. The equation (7) simply sums the error of applying the flow velocity vector v to the spatial and temporal gradients of all the surrounding neighbours using the optical flow constraint equation (3). We should be able to find a v consistent with the neighbouring spatial and temporal gradients. If it is inconsistent with some neighbours, it would have a larger error. Applying it to a small neighbourhood, e.g. 5x5, we get more equations than unknowns. To solve this problem we should use the least squares method, getting

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x^2(p_i) & \sum_i w_i I_x(p_i) I_y(p_i) \\ \sum_i w_i I_x(p_i) I_y(p_i) & \sum_i w_i I_y^2(p_i) \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(p_i) I_t(p_i) \\ -\sum_i w_i I_y(p_i) I_t(p_i) \end{bmatrix}. \quad (8)$$

This method makes a local calculation of the flow vector. Unlike Horn and Schunck's iterative technique, this does not rely on the entire image to achieve a good estimate. This means that Lucas and Kanade's method is able to separate different regions with different optical flows, while in the other method vectors from different regions affect each other. However, this technique does not fill in homogeneous regions. Nevertheless, this method is considered one of the best optical flow computation techniques as it performs consistently and robustly and it is of easy implementation [10].

2.3. Object Classification

The extracted moving region detected by the techniques mentioned in the previous section may represent different objects. Hence, we have to use other methods to recognize features that allow us to determine which object is the one that we want to track. This presents many challenges since the same object may be seen from different viewpoints, the illumination may vary and there may be occlusions.

2.3.1 Harris Detector

The Harris detector, proposed by Harris and Stephens [6], is a corner detector. The term corner in this context is used to represent points in the 2D image for which there are two dominant and different edge directions in a local neighbourhood of the point. Looking at a local window and shifting it through the region of interest (ROI) that we are studying, the corner is detected by a large change in the intensity of some pixels in that window. Corners are considered good interest points because they can give us true matches, while in flat zones and along the edges there are a lot of different pixels that don't correspond to the actual point (false matches). In order to find these changes in the intensity, this method suggests to calculate an error based on the auto-correlation function:

$$E(u, v) = \sum_{x,y} w(x, y)^2 (I(x + u, y + v) - I(x, y))^2, \quad (9)$$

where $w(x, y)$ is the weight function and $I(x+u, y+v)$ represents the shifted intensity. Applying a Taylor expansion to the shifted intensity we get:

$$\begin{aligned} E(u, v) &\approx \sum_{x,y} w(x, y)^2 (uI_x + vI_y)^2 \\ &= \sum_{x,y} w(x, y)^2 \Big(\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \Big)^2 \\ &= \begin{bmatrix} u & v \end{bmatrix} S_a \begin{bmatrix} u \\ v \end{bmatrix}, \end{aligned} \quad (10)$$

where $S_a = \sum_{x,y} w(x, y)^2 \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$. S_a describes the shape of the local auto-correlation function at the origin. The eigenvalues of the matrix S_a (λ_1 and λ_2), are proportional to the principal curvatures of the local auto-correlation function and develop a rotationally invariant description of S_a. Therefore we have three possibilities: if both curvatures are small, that is, λ_1 and λ_2 are small, it means that E is almost constant in all directions, the auto-correlation function is flat, so, we are in a flat region; if $\lambda_1 >> \lambda_2$ or $\lambda_1 << \lambda_2$, we found an edge, the auto-correlation function is ridge shaped; if λ_1 and λ_2 are large and with similar value, it means E increases in all directions, we found a corner To measure the corner response, Harris formulated the following equation:$R = Det(S_a) - c(Tr(S_a))^2$, where Det is the determinant of the matrix, Tr is the trace, and c is an empirical constant with usual value between

0.04 and 0.06. Notice that R only depends on the eigenvalues of S_a. This equation gives positive values if we are in a corner region, negative in edges and small in flat regions. Using thresholds, we are able to find good interest points. Harris detector is covariant to image translation and rotation and it is partial invariant to affine intensity change (since it only uses derivatives, it is invariant to intensity shifts but not to intensity scaling). However it is not covariant to image scaling [6].

2.3.2 Viola–Jones Detector

Paul Viola and Michael Jones presented a solution in 2001 for object detection and classification that stands apart from all the others because of its speed in doing so. Their method was the first to achieve real-time face detection [19]. This technique uses Haar-like features. We apply to the image a series of rectangular masks, each feature results in a single value calculated by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle. The region where the feature has a high value corresponds to the one to which the mask was designed. Viola-Jones algorithm usually uses a window of 24x24 pixels and performs different kinds of masks with different sizes to the entire image, obtaining more than 160000 features per window. To help computing all the pixels sums, it was used the idea of integral image. In an integral image the value of a certain pixel is the sum of the pixels above and to the left of that pixel. Although each feature is calculated very efficiently, computing all the 160000 is very time consuming. Viola and Jones suggest a way to find out which features are more relevant to detect a given object, using Adaptive Boosting, or AdaBoost. AdaBoost is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire [4]. It works by assigning weights and choosing the relevant features (called weak classifiers):

$$C(x) = w_1 cl_1(x) + w_2 cl_2(x) + w_3 cl_3(x) + ..., \tag{11}$$

where $C(x)$ is the final strong classifier (it informs if we found the object or not), w_j represents the normalized weights and cl_j represents the weak classifiers related to the different features. The weak classifiers take a binary value, according to a threshold: they are one if they performed well and found features in the images; zero if they did not. The weights are related to the miss classified images. We have to apply this strong classifier to each 24x24 window in our image. If the classifier is really extensive it would take a lot of computation time. So, it is proposed the implementation of a cascade of classifiers. First we apply a classifier that is related to a group of ten main features. This way, we quickly exclude sub-windows that, most certainly, our object will not have. Then, more specific groups of classifiers will be used to exclude the remaining false positives. This algorithm works really well for real time object recognition with distinctive patterns, such as a human face.

2.3.3 Scale-Invariant Feature Transform (SIFT)

Scale-invariant feature transform is a very important method for extracting distinctive invariant features from images, developed by David Lowe in 2004 [9]. Besides finding many interest points with quality, this technique also makes a correct match against a large database of features from many images. This detector and descriptor is invariant to image scale and rotation and it is robust to change in 3D viewpoint, addition of noise and affine (rotation, scale, shear) distortion. SIFT has also the advantage of being able to perform close to real-time.

SIFT finds local features by detecting blob (binary large object) regions in an image, where some properties are constant. This recognition is made by applying the Laplacian of Gaussian (LoG) filter to the image, also called Mexican hat filter because of its graphic appearance:

$$\begin{aligned} \nabla^2 L(x,y,\sigma) &= \nabla^2 \Big(G(x,y,\sigma) * I(x,y) \Big) \\ &= \nabla^2 \Big(G(x,y,\sigma) \Big) * I(x,y), \end{aligned} \tag{12}$$

where $G(x,y,\sigma)$ is a Gaussian filter. Different values for σ give different scales to the transformed image. To find which scale is the best to apply to a certain point of the image we have to execute different LoG filters to that point, with different values for σ, and observe which one gives the higher response value. The filter that generates the maximum value in the response is the one that we choose for that pixel. To compute all these filters efficiently, it is proposed to apply Gaussian pyramids to the image, because it can be proved that there is a relation between LoG filters and the difference of Gaussian (DoG) filters ($D(x,y,\sigma) = (G(x,y,c\sigma) - G(x,y,\sigma)) * I$, where c is a constant). To choose the right pixel to be an interest point, we have to compare every pixel in a 3x3 neighbourhood region in the current and adjacent pyramid levels (scales). As most sample points are going to be eliminated in the first checks, the computational cost of this comparison is low. However, this does not necessary provide their exact location. To find the precise location, Lowe suggests a detailed fitting of the keypoints into the nearby data. Based on a Taylor series expansion it is possible to fit a quadratic function to the local sample points and determine the interpolated location of the real maximum. Nevertheless, some keypoints are not good candidates to be interest points because they have low contrast. To eliminate those, we apply a threshold (usually 0.03) and only if the value of $D(x, y, \sigma)$ is larger than that pre-determined threshold, we consider that to be a keypoint. The search for good interest points is not over yet because we have to eliminate those points which are poorly localized along edges, as DoG function gives a strong response there. A poorly defined peak will have very low curvature along the edge and a high across the edge. To obtain its principal curvatures we have to compute the Hessian matrix (H) of D. Using the same approach used by Harris, we can find its eigenvalues (λ_1 and λ_2) from the calculation of trace (Tr) and determinant (Det). To eliminate the points with low curvature in one of the directions we establish a maximum limit for the ratio (usually $r = 10$) between the eigenvalues of H. We have now selected good keypoints that guarantee the scale invariant aspect of our descriptor.

Besides finding interest points, SIFT also describes the region around them, which will help tremendously with the classification of that object. A SIFT descriptor starts by computing the gradient magnitude ($m(x, y)$) and orientation ($\theta(x, y)$) around the keypoint, through pixel differences. With those values we calculate weighted orientation histograms, for each small region of the neighbourhood, and concatenate them achieving our final descriptor vector. Empirically, Lowe found out that the best results were when we computed 4x4 array of histograms with 8 orientation bins each. Therefore, our descriptor vector will have 128 elements. To guarantee the invariance of our descriptor to changes in illumination, we have to normalize the vector to unit length. To overcome non-linear illumination changes we should make a threshold to reduce the influence of large gradient magnitudes, and then we renormalise the vector to unit length again. Experimentally it was found that the threshold should have the value 0.2. To achieve the classification of the object, we need to match our local features to the ones of the images that we have in the database. This is done by calculating the minimum Euclidean distance between our interest point and the ones that are in the database, finding the nearest neighbour. As the Harris detector (section 2.3.1), SIFT also provides a way to select interest points in a image, with the advantage of being invariant to scale, but the really benefit of SIFT is that it is a descriptor, giving us details about the region around the keypoint [9].

2.3.4 Histograms of Oriented Gradients (HOG)

As SIFT, the histogram of oriented gradients (HOG) is a feature descriptor. It was first described by Navneet Dalal and Bill Triggs in 2005 with the intention to detect humans, but it has proved to have many more applications [3]. This method also starts by analysing the gradients of the image, but it will not use the smoothed image. The HOG descriptor computes the magnitude and the orientation gradients only from the x and y derivatives of the image. If we have a colour image we should choose the colour channel with the highest gradient magnitude for each pixel. As HOG was initially designed for human identification, it goes over images with 64x128 dimension. It divides the image in 16x16 pixel blocks with 50% overlap, giving a total of 105 blocks of 2x2 cells. To create the orientation histogram of each cell from each block, we quantize the gradient orientation into 9 bins (from 0° to 180°). Then each pixel contributes to the histogram according to a linear interpolation between neighbouring bin centres. The vote can also be weighted with a Gaussian distribution to down weight the pixels near to the edges of the block. Then, we concatenate all the histograms, obtaining a 1D vector with length 3780 ($105 \cdot 4 \cdot 9$). HOG is considered a global feature descriptor but we can also use it with a feature detector to obtain better results. After obtaining the vector we can use the technique mentioned at the end of section 2.3.3 to match images and classify our object [3].

2.4. Object Tracking

The aim of an object tracker is to generate the trajectory of that object over time, by locating its position in every frame of the video [20]. After detecting the object that we want to track through the methods described in sections 2.2 and 2.3, we need to apply algorithms to establish a correspondence of the objects across frames. These tracking algorithms should also be able to predict the object trajectory, saving, this way, computational memory and reducing the noise. In this thesis we will explain how this is done through methods based on mean shifts and through a Kalman Filter.

2.4.1 Mean Shift

The mean shift algorithms are an efficient approach to tracking objects whose appearance is defined by histograms. So, they should be used in parallel with the SIFT or the HOG descriptors (sections 2.3.3 and 2.3.4). These methods estimate the motion of the object by iteratively shifting a data point to the average of the data points in its neighbourhood.

The mean shift vector is given by:

$$M_h(\mathrm{y}_0) = \left(\frac{\sum_{i=1}^{n_\mathrm{x}} w_i(\mathrm{y}_0)\,\mathrm{x}_i}{\sum_{i=1}^{n_\mathrm{x}} w_i(\mathrm{y}_0)} \right) - \mathrm{y}_0, \qquad (13)$$

where n_x is the number of points in the neighbourhood, y_0 is the initial location of our point, x_i are the data points vector, $w_i(\mathrm{y}_0)$ is the weight associated to the point i in relation to our initial point y_0 and h is the radius of the ROI, also called kernel radius. One of the techniques based on the mean shift help creating a confidence map in the new image based on the colour histogram of the object in the previous image, and use mean shift to find the peak of a confidence map near the object's old position. The confidence map is a probability density function on the new image, assigning each pixel of the new image a probability, which is the probability of the pixel colour occurring in the object in the previous image. These methods do not work well with occlusion, nor when we want to track more than one object [2].

2.4.2 Kalman Filter

The Kalman filter has many uses, including applications in control, navigation and also in computer vision. It is really useful to help with tracking objects because this filter is able to estimate the state of a dynamic system, taking into account Gaussian noise disturbance [8]. This filter has two recursive stages. At each iteration, it predicts the current location of the object, based on previous observations and then, it corrects the models, considering the measurement of the object's actual location. Although this is an extraordinary filter that produces very good results in an efficient time, it has a big constraint that it can only be applied to linear Gaussian models. To overcome this problem it was developed a non-linear version, the Extended Kalman Filter. This filter produces faster results in terms of iterations, but at a higher cost.

3. Interface Development within the Senses Places Project

Senses Places (SP) is a dance-technology collaborative project that aims to create playful mixed reality performance environments for audience participation. This project was initiated by Isabel Valverde and Todd Cochrane in 2010, respectively as artistic and technical directors [18]. Through scheduled events, welcoming anyone who wants to join, participants from all over the world, with different cultures and areas of expertise meet physically and/or virtually, in the 3D Virtual World (3DVW), Second Life (SL), and perform together.

3.1. Interface Design

The webcam interface within SP was first presented in 2010. It is implemented on a web page in an embedded Adobe®Flash® component. The goal of this interface is to recognize the participant's movements through a webcam and to connect them to the avatar's movements in the 3DVW. The web page displays a video of the webcam view that is super imposed with a graphical abstraction in the form of disks connected by lines, representing the proposed position of the detected movement of a head, hands and torso a skeleton. This interface has some limitations, as it only considers the movement of the upper body in a disfigured and very restrained way, see figure 1.

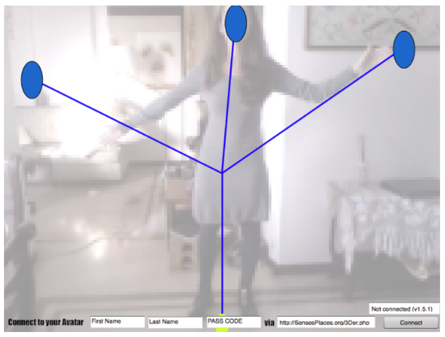

Figure 1: SP's first webcam interface. The blue disks represent the hands and the head and the yellow represents the torso. The disk that symbolizes the head has the freedom to move in both axis (x and y), the other ones only move vertically (in the y axis)

What we proposed to do within the SP project was a better recognition of the human movements, maintaining the easiness of the interface online use, so that the user does not have to install any other software. We decided to implement the new interface using javascript and HTML5 since with the new Canvas element, HTML5 can draw and manipulate videos in the screen quicker. Nevertheless, tracking different body parts online in real time is not an easy thing to do. We started by detecting movement and then we tried to come up with a way to distinguish what body part performed the corresponding movement.

After testing different algorithms, we concluded that what works best is to apply the frame difference algorithm (section 2.2.1) to every frame. With this new interface we wanted to track more body parts, so we decided to use 7 nodes symbolizing the position of the head, hands, chest, pelvis and feet. To help distinguish the movements made by the difference body parts, we chose to delegate each node to a region where they could move, restricting the movement a little bit, see figure 2.

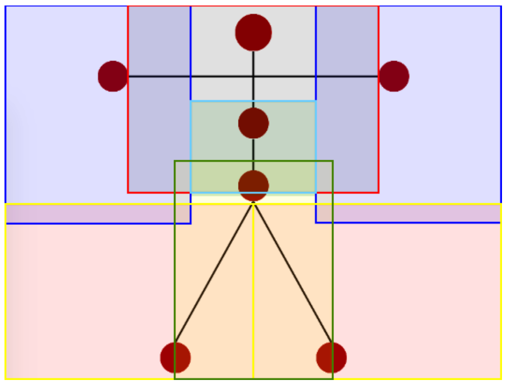

Figure 2: Regions that detect the different body parts, considering seven nodes

Computationally we found out that it was more efficient to perform the frame difference algorithm by region. We started searching for movement in the upper left side of each area, performing the subtraction pixel by pixel from the left to the right, top to bottom. As soon as it would find a difference between pixels, the node moved to there. To avoid false positives related to noise, we only allowed the node to move to a place where the movement was recognized in four consecutive pixels. In overlapping areas the node that moved was the one closest to that zone.

At this moment, we had an interface that recognized movement in different parts of the frame, but it did not actually recognized the moving object. We also concluded that most of the techniques that we explored on the previous chapter for object classification were not suitable for real time multiple objects recognition, because of their expensive computation time. The Viola-Jones detector (section 2.3.2), requires the training of good classifiers, which is particularly difficult in case the objects do not possess distinctive features, as for example, the chest, pelvis or feet. Our hands do have a very distinctive form, however, they are so articulated that during our movement they can take many different shapes, making it hard to recognize them through patterns. The only body part that does not change that much and has unique features, making it easier to distinguish from all the other objects, is the head. Thus, we decided to incorporate a face recognition based on Viola-Jones method to our movement recognition interface. If it is not able to find a face, it resumes with the frame difference technique for that node. To improve the interface further more, making it always assume a human-like skeleton representation and avoiding confusion by the overlap of nodes, we imposed some constraints regarding the maximum and minimum distances between nodes. These distances were found empirically.

3.2. 3DVW Interaction

The connection of the new interface to the 3DVW was established almost in the same way as the first webcam interface, implemented by Todd Cochrane [18]. When a movement is detected, the coordinates to where the node moved, along with the information about which node it was, are sent to a PHP script on a web server. The Heads Up Displays (HUDs) that are given to the participants in the virtual world, contain a script that pulls this information and animations that will make the avatar move, accordingly.

In the previous interface, the 3DVW receives information about what body part has moved, and then it randomly selects between four animations, previously determined to be connected to that node. Now, we implemented two different kinds of 3DVW scripts. One is based on the previous one, just described. This script is only concerned about which body part has moved, and after that recognition it arbitrarily selects one of the four animations previously chosen to be related to that body part. The second script, besides detecting what node has moved, it also analyses where the nodes are located and according to their location triggers a different animation. To improve the choreographic goals of the project, we created animations to the avatar in the biovision hierarchy format (BVH), specifying how the avatar will move when a certain animation is activated (which body part will move, at what speed, etc...). This way, we are also able to establish a better correspondence between the location of the nodes and the chosen animation that the avatar will perform. The created animations for this second script involve small movements of the body parts, but it also allows their combinations. Thus, the developed animations created complement each other, bringing a full body movement to the avatar. The new data, more detailed and accurate, allow participants a more fulfilling engagement and a way to perceive the avatar as a dance partner, providing a stronger sense of immersion.

4. Results and Interface Testing

In order to detect bugs in our program and test how well it works, we first performed several tests, both on the webcam interface and on the 3DVW scripts. Then, we shared the interface with the SP community to receive feedback about how it fits the performance and the participatory experience.

4.1. Movement Tracker

When performing isolated movements of a body part with low or high speed, the interface makes a correct tracking, within the imposed regions shown in figure 2. We may see the results when we capture images of the left arm performing a descent movement with different speeds, in figure 3.

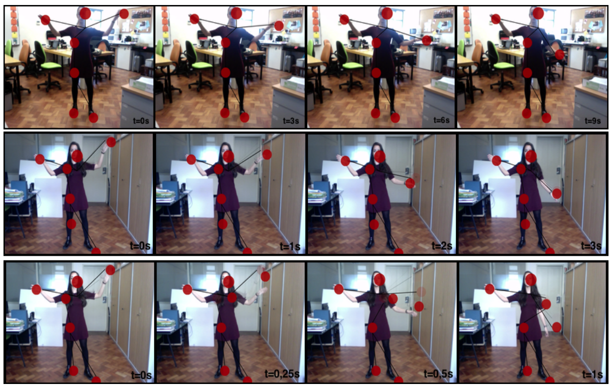

Figure 3: Captured frames of arm movements with different speeds, $\Delta t = 3s$, $\Delta t = 1s$ and $\Delta t = 0.25s$, respectively.

The feet nodes have more difficulties to stay connected to the user's feet. They tend to rise a little too much, and may even reach the knees. Nevertheless, the interface tracks well the lower part of the body, which was an important goal for the new SP webcam interface. When performing movements that comprise different body parts simultaneously, it is harder to track with precision. The chest node easily jumps to the elbows, and the pelvis sometimes jumps to the knees, but they will quickly go back to track what they are supposed to, as can be seen in figure 4.

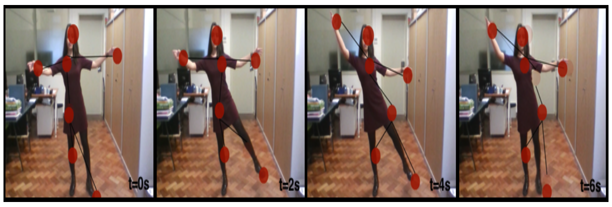

Figure 4: Captured frames of a movement with intervals $\Delta t = 2s$.

It was important that the interface would be able to track movements closer to the floor, so that the participant could express more freely. In figure 5 we can see that we have succeeded on using the interface to follow up an down movements.

Moreover, the face recognition algorithm works really well, as can be seen in all the figures presented in this section.

4.2. 3DVW Scripts and Animations

We are able to correctly connect the webcam interface located in a web page to the virtual world, with the password provided by the 3DVW script. The

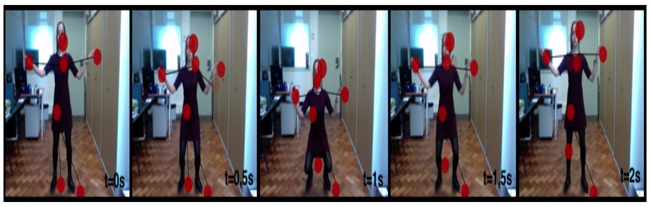

Figure 5: Captured frames of the user getting lower, $\Delta t = 0.5s$.

data informing what node has moved, and the coordinates to where it moved is sent to the PHP web page through the online webcam interface every two seconds, and it is pulled by the virtual world in the same time interval (these intervals can grow bigger, when the internet connection is weak). Hence, we have to take these delays into account when testing the interface in connection with the virtual world. In figures 6 and 7 we can see some examples of the avatar's reaction to human movements.

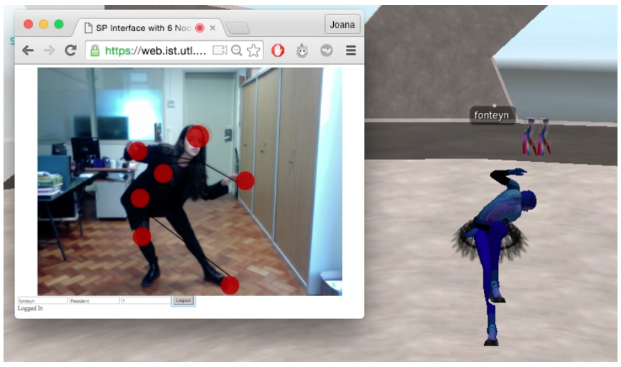

Figure 6: Reaction of the avatar to a movement out of the verticality.

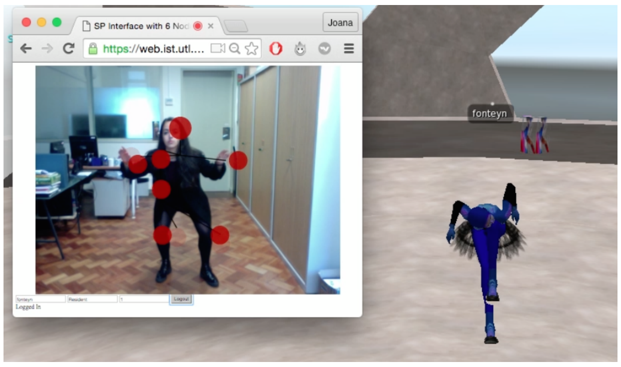

Figure 7: Reaction of the avatar to a low movement with the arms bent.

The tests confirm that no animation inhibits others, because when we created the HUDs we were careful enough to update animations with the same level of priority. Notice that this was a justified concern, once with the former webcam interface (see figure 1) the avatar often would get stuck in some positions and animations, and the only way to emend that was by taking the HUD off.

4.3. Users' Response

After making our own critical analysis on the software that has been developed, we decided that it was time to be made public and shared with the SP's community and with whoever was interested. To better understand their opinion and identify the flaws that the users could find, we prepared a questionnaire for them to answer. We gathered information from twenty one respondents (ten males and eleven females), from ages between nineteen to sixty two. The majority of them have background in engineering and some in arts. Despite some of them having experienced a MUVE, more than half of the respondents who answered to our questionnaire were not SP collaborators and have never participated in a mixed reality performance and environment before. Although the internet connection was not the best for a few participants, the overall evaluation of the movements tracked by the online interface was positive (4.09, where 5 corresponded to "very good"). Those people who had the opportunity to experience the interface in combination with the SL script rated the animations and the HUDs provided by us with an average of 4.14 (where 5 corresponded to "very appropriate"). Overall the twenty one participants enjoyed the interface and classified the experience in an average of 4.52 (where 5 corresponded to "very enjoyable"), figure 8.

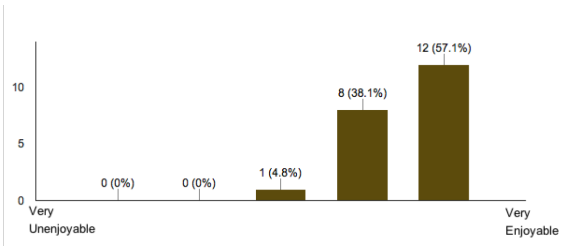

Figure 8: Rating of the overall experience. The average of the 21 answers was 4.52, where 1 corresponded to "very unenjoyable" and 5 "very enjoyable".

At the end of the questionnaire, we asked for individual opinions and bug reports. The main criticism made by several participants was that they had to place themselves too faraway from the camera, in order to get the nodes correspond properly to the body parts as they were supposed to. Since our interface was designed to operate as a dance interface, it is understandable that we intended that the participants experienced it in a large room, where they may dance and move freely in space.

5. Conclusions

This dissertation addresses at large the topic of object tracking through video capturing, which is a very important topic within the research field of computer vision. It has numerous applications in the most different areas, namely in engineering, medicine, security, entertainment and so on. Particularly, we focused on the application of this technology to the performing arts. The ultimate goal of the present work was to create a webcam interface for a dance-technology project, Senses Places, to help establishing a deeper connection between the virtual and physical worlds. We presented several methods to recognize and track objects via video, with the goal of choosing the right ones to develop our own movement tracking interface.

The interface developed within the SP project is based on the frame difference algorithm and the Viola and Jones method. We also created Second Life scripts to help reinforce the choreographic approach of the project, based on the unpredictability and kinaesthetic engagement. Our work is innovative in a sense that it makes a satisfactory online real time recognition of different objects moving without the use of special cameras or settings. We want to remark that an early description of the developed interface was submitted as a short paper, presented at the International Symposium of Electronic Arts in Hong Kong (ISEA 2016), and it is now published in the conference proceedings [11].

The work developed in this thesis had the specific main goal to serve a dance-technology project. However, if we adapt the code, we can use it for other applications. For instance, it can help create accessible interfaces for technological control, motion sensors and simulators.

Acknowledgements

I would like to express my gratitude for all the support and encouragement to my advisor Ana Moura Santos, to the artistic director of the Senses Places project Isabel Valverde, to my friends and to my parents and my brothers.

References

[1] S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. *Video Communications and Image Processing*, 2004.

[2] C. Comniciu and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, 2003.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *The Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[4] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 1997.

[5] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, Jan. 1999.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[7] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 1981.

[8] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82, 1960.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2002.

[10] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of Imaging Understanding Workshop*, pages 121–130, 1981.

[11] J. Martins, T. Cochrane, I. Valverde, and A. Moura-Santos. A webcam interface for somatic-technological dance experiences. In *Proceedings of the ISEA 2016*, 2016.

[12] H. Parekh, D. Thakore, and U. Jaliya. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(2), 2014.

[13] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2–3):210–220, 2006.

[14] A. Stalin and A. Wahi. BSFS: Background subtraction frame difference algorithm for moving object detection and extraction. *Journal of Theoretical and Applied Information Technology*, 60(3):1559–1565, 2014.

[15] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. Technical report, The Artificial Intelligence Laboratory, M.I.T, Cambridge, MA, 1998.

[16] I. Valverde. *Interfaces Dança-Tecnologia: Um Quadro Teórico Para a Performance no Domínio Digital.* Fundação Calouste Gulbenkian, 2010.

[17] I. Valverde. Senses places: developing a somatic dance-technology approach. Unpublished paper, 2015.

[18] I. Valverde and T. Cochrane. Innovative dance-technology educational practices within senses places. *Procedia Technology*, 13, 2014.

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *The Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[20] A. Yilmaz, O. Javed, and M. Shah. Object tracking: a survey. *ACM Computing Surveys*, 38(4), 2006.