Face Tracking: An implementation of the Kanade-Lucas-Tomasi Tracking algorithm

Ben M Herbst
Stellenbosch University

Face Tracking : An implementation of the Kanade-Lucas-Tomasi Tracking algorithm

Dirk W. Wagener, Ben Herbst

Department of Applied Mathematics, University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa
dwagen@dsp.sun.ac.za, herbst@ibis.sun.ac.za

Abstract

The aim of this project is to develop a robust and effective face (head) tracker based on the Kanade-Lucas-Tomasi (KLT) tracking algorithm. The project concentrates mainly on robust tracking despite excessive background clutter.The problem of disappearing features and the re-generation of features on a persons head is also investigated. The system can also be used with two video cameras to track a person's head in three dimensions. The following report discusses the steps necessary to achieve the tracking.

1. Introduction

It is easy for the human eye to follow another person's head while he or she moves around. Pose the same challenge to a computer and it is a totally different story. In the world of computer vision tracking has long been a formidable task. There are basically three steps which has to take place during human face tracking:

- Locating the face in an image stream.
- Selecting features to track.
- Tracking the features.

In the sections that follow we will presume that the face location is already known. The second and third items are discussed in more detail.

2. Image Processing

2.1. Feature selection

2.1.1. Introduction to feature selection

The most important step before tracking is the selection of "trackable" features. How do we select features which are trackable? The keyword during feature selection is texture. Areas with a varying texture pattern, are mostly unique in an image, while uniform or linear intensity areas are often common and not unique. It is clear that we have to look for areas of texture and the first logical step would thus be to investigate the image intensity gradients as discussed in [2].

2.1.2. Finding a good feature

A texture pattern can only exist if we look at multiple pixels in an area. The features that we are tracking are more accurately described as feature windows containing texture information. The area of such a feature window can vary, depending on the number of features which needs to be located. Let us consider a feature window of size W (typically 8× 8 pixels). An example image figure 1 will be used to locate face features. The image gradient is defined as:

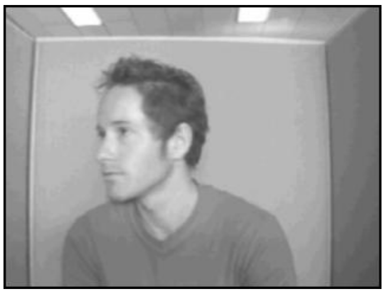

Figure 1: *First image in a sequence.*

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \underline{\nabla} I$$

The image gradients are shown in figures 2 and 3. Now consider the product

$$\begin{aligned} \mathbf{g}\mathbf{g}^{\mathrm{T}} &= \begin{bmatrix} \frac{\delta(I)}{\delta x} \\ \frac{\delta(I)}{\delta y} \end{bmatrix} \begin{bmatrix} \frac{\delta(I)}{\delta x} & \frac{\delta(I)}{\delta y} \end{bmatrix} \\ &= \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \end{aligned}$$

If we now integrate the matrix derived above over the area W, we get:

$$Z = \iint_W \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \omega d\mathbf{x}$$

with $\mathbf{x}$ = (x,y) and ω a weighting function.

The 2× 2 matrix Z contains pure texture information and by analyzing its eigenvalues we can classify the texture in the region W. Two small eigenvalues represents a roughly constant intensity pattern within W. One small and one big eigenvalue means that a linear (unidirectional) pattern was detected. If Z has two large eigenvalues, then the area W contains a feature which can be tracked. This feature can either be a corner, a so called salt-and-pepper texture or any texture pattern which has a prominent enough change in intensity in more than one direction. The equation for Z forms an intricate part of the Kanade-Lucas-Tomasi tracking equation. It will become clear later that choosing features according to the criterium described above

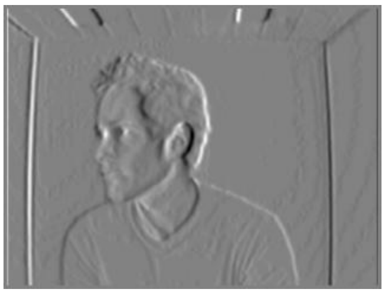

Figure 2: *Image gradient in the x - direction.*

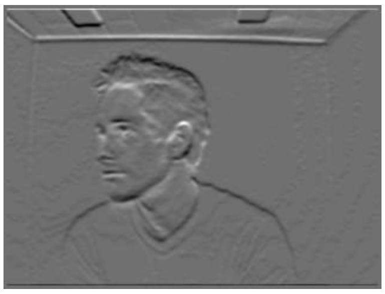

Figure 3: *Image gradient in the y - direction.*

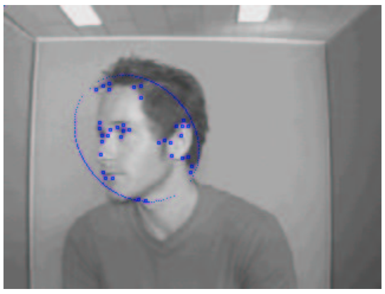

Figure 4: *Detected features with an ellipse around them.*

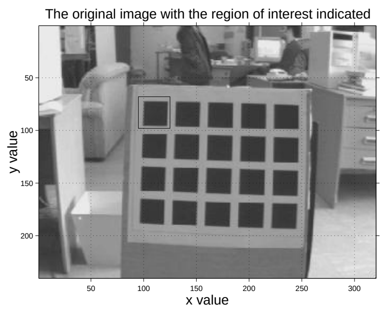

Figure 5: *An image to illustrate corner detection.*

yields the features which can be best tracked by the tracking algorithm.

The question now is how do we decide when an eigenvalue is big enough ? It is necessary to establish a minimum threshold for the value of the eigenvalues. Let's call the eigenvalues λ_1 and λ_2 with $\lambda_2 \geq \lambda_1$. Another criterium for choosing suitable features is the Harris "cornerness" function,

$$R = \det \mathbf{Z} - k(\text{trace}\mathbf{Z})^2. \quad (1)$$

The constant k is usually taken as 0,04 (originally suggested by Harris). Consider figure 5 of a chess board image. The region that we are going to inspect is indicated with a rectangular box.

Calculating Z for all the W's in this region and then determining λ_1,λ_2 and R, produces figure 6. The values of λ_1,λ_2 and R are plotted against the x -value of the image. From figure 6 it is clear that λ_1 and R have distinctive peaks in the regions where the x-values correspond to corners. It is also clear that the amplitudes of the R peaks are much bigger than the amplitudes of the λ_1 peaks and thus are easier to detect. Figure 7 is a three dimensional surface plot of R taken over the entire area indicated by the rectangle in figure 5. The trackable features in this plot are clearly the local maxima of the R function. We thus have a suitable criterium for selecting the relevant features. Figure 4 shows the features selected by the algorithm for its first image figure 1.

2.2. KLT Tracking

2.2.1. Introduction to the method

Assuming that a human face has already been located in an image sequence the subsequent tracking is based on the Kanade-Lucas-Tomasi tracking equation [1]. In a nutshell the technique can be described as follows: First of all features have to be selected which can be tracked from image to image in a video image stream. The selection of the features is based on texture (intensity information). A number of fixed-sized feature windows are selected on the head of a person in the first image of a sequence. These feature windows are tracked from one image to the next using the KLT method [2]. This method calculates the sum of squared intensity differences between a feature in the previous image and the features in the current image. The displacement of the specific feature is then defined as the displacement that minimizes the sum of differences. This is done continuously between sequential images so that all the features can be tracked.

2.2.2. Tracking facial features

The fundamentals of tracking can be explained by looking at two images in an image sequence. Let us assume that the first image was captured at time t and the second image at time t + τ. It is important to keep in mind that the incremental time τ depends on the frame rate (capture rate) of the video camera and should be as small as possible. A higher frame rate allows for better tracking.

A grayscale image is a pattern of intensities where the intensity values range from 0 to 255. Figure 8 illustrates that an image can be represented as function of variables x and y. We add the variable t as the time the image was captured. Thus, any section of an image can be defined by an intensity function I(x,y,t). This is illustrated by figure 8.

Let us now define a window in an image taken at time t+τ as I(x,y,t+τ). The basic assumption of the KLT tracking algorithm is

$$I(x, y, t + \tau) = I(x - \Delta x, y - \Delta y, t). \quad (2)$$

From (2) it is clear that every point in the second window can by obtained by shifting every point in the first window by an

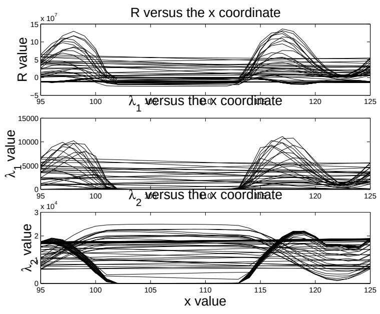

Figure 6: *The calculated texture variables.*

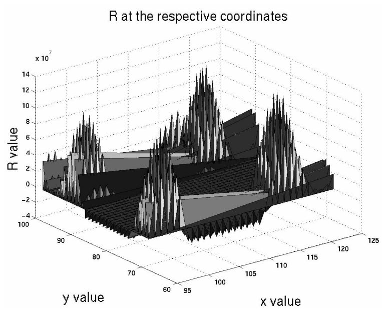

Figure 7: *A surface plot of R for the area W.*

amount (Δ x,Δy). This amount can be defined as the displacement $\mathbf{d}$ = (Δx,Δy) and the main goal of tracking is to calculate $\mathbf{d}$.

It is clear that only facial features that remain visible in an image sequence will obey (2). This brings us to an important problem of successful robust face tracking: How do we keep track of features when they disappear due to a rotation of the subject or due to occlusion? We will deal with this problem later. At this stage only visible face surface markings will be tracked.

2.2.3. Calculating feature displacement

The basic information has now been established to solve the displacement $\mathbf{d}$ of a face feature from one window to the next. For simplicity, we redefine the second window as B($\mathbf{x}$) = I(x,y,t+τ) and the first window as A($\mathbf{x}$ - $\mathbf{d}$) = I($\mathbf{x}$ - $\mathbf{d}$) = I(x-Δx,y-Δy,t) where $\mathbf{x}$ = (x,y). The relationship between the two images is given by

$$B(\mathbf{x}) = A(\mathbf{x} - \mathbf{d}) + n(\mathbf{x}). \tag{3}$$

n($\mathbf{x}$) is a noise function caused by ambient interference such as scattered light and reflections. At this stage we can define an error function which has to be minimized in order to minimize the effect of noise:

$$\epsilon = \iint_W [A(\mathbf{x} - \mathbf{d}) - B(\mathbf{x})]^2 \omega d\mathbf{x} \tag{4}$$

W is the window within the first image over which the double integral is taken. The function ω in (4) is a weighting function which can be adjusted according to the expected intensities in the image and is often chosen ω = 1. If the center of each scan window W carries the most important intensity information, ω can be set to a Gaussian-mask to emphasize those areas.

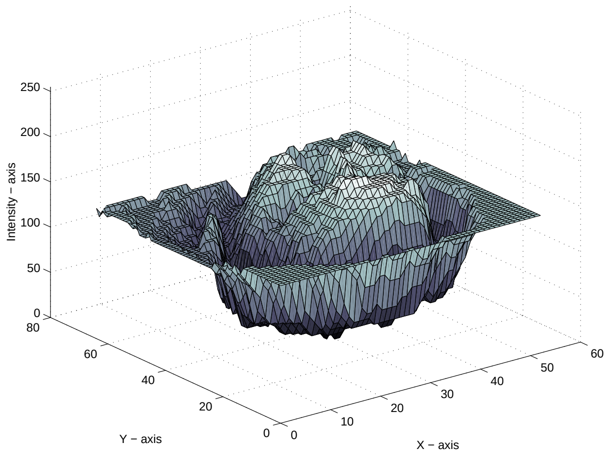

Figure 8: *Surface representation of a section of an image.*

It is clear from (4) that a displacement $\mathbf{d}$ must be chosen to minimize the error function ϵ and thus minimize the noise. We know that the displacement $\mathbf{d}$ is a small incremental vector, hence the earlier requirement of a high frame rate. A Taylor expansion yields

$$A(\mathbf{x} - \mathbf{d}) = A(\mathbf{x}) - \mathbf{g} \bullet \mathbf{d}. \tag{5}$$

In (5) $\mathbf{g}$ refers to the image gradient as defined before. Now we can write the error equation (4) as,

$$\epsilon = \iint_W [A(\mathbf{x}) - B(\mathbf{x}) - \mathbf{g}^{\mathrm{T}}\mathbf{d}]^2 \omega d\mathbf{x}. \tag{6}$$

We can see from the right-hand side of (6) that the ϵ is an quadratic function of $\mathbf{d}$. To minimize ϵ we simply have to differentiate ϵ with respect to $\mathbf{d}$ and set the result equal to zero:

$$\frac{\delta\epsilon}{\delta\mathbf{d}} = \iint_W [A(\mathbf{x}) - B(\mathbf{x}) - \mathbf{g}^{\mathrm{T}}\mathbf{d}]\mathbf{g}\omega dA_r = 0 \tag{7}$$

Where A_r refers to the area of window W. If the terms of (7) are rearranged and we use the fact that ($\mathbf{g}^{\mathrm{T}}\mathbf{d}$)$\mathbf{g}$ = ($\mathbf{g}\mathbf{g}^{\mathrm{T}}$)$\mathbf{d}$, it follows that

$$\left(\iint_W \mathbf{g}\mathbf{g}^{\mathrm{T}} \omega dA_r\right) \mathbf{d} = \iint_W [A(\mathbf{x}) - B(\mathbf{x})]\mathbf{g}\omega dA_r. \tag{8}$$

Equation (8) is known as the Kanade-Lucas-Tomasi tracking equation and can be rewritten as follows:

$$Z\mathbf{d} = \mathbf{e} \tag{9}$$

Where Z is a 2$\times$2 matrix

$$Z = \iint_W \mathbf{g}\mathbf{g}^{\mathrm{T}} \omega dA_r \tag{10}$$

and $\mathbf{e}$ is a vector in two dimensions

$$\mathbf{e} = \iint_W [A(\mathbf{x}) - B(\mathbf{x})]\mathbf{g}\omega dA_r \tag{11}$$

The displacement $\mathbf{d}$ is now simply the solution of equation (9).

2.2.4. *Interpretation of the method*

Now that we have covered the mathematics, we look at what happens physically during the execution of the tracking algorithm. The basic tracking steps are as follows: First of all we assume that we have one feature window W in the first image and we want to determine the displacement of this window in the second image. W in the first image is used to calculate the Z matrix in 10. The vector $\mathbf{e}$ is calculated by subtracting W in the second frame from W in the first frame and multiplying the result with the gradient calculated for (10) and a weighting function applied to W. The displacement can now be solved from equation (9). To find the displacement $\mathbf{d}$, ϵ in 6) has to be minimized.

Processing an image with a high resolutions is computationally expensive. Seeing that face tracking is a real-time application, we need to minimize the processing time in order to keep up with a moving person. The KLT-algorithm can be used in a multi-resolution image pyramid for processing the images. We explain it by using an example shown in figure 9. Consider the following figure:

The original image was captured with a resolution of 320×240. A resolution pyramid with 4 levels was chosen and the subsampling was set to 2. In other words every lower level in the pyramid has double the resolution of the level above it. Before any tracking can take place, features must be selected. This was covered in the previous section. The feature selection algorithm was used on the first image in the video stream to select the features that will be tracked. All the features are stored in a feature list containing their x and y coordinates in the image and a value identifying the current status of the feature. Each feature in the list is tracked from the top of the pyramid to the bottom. In other words the smallest resolution (in this case 40×30) is used to best locate the feature. The new location of the feature is then passed to the next level of the pyramid and is used as the starting point for the search for the higher-resolution location of the feature. This process is repeated until the final level of the pyramid is reached. If a feature is successfully tracked, the last level in the pyramid will provide the final location of the feature.

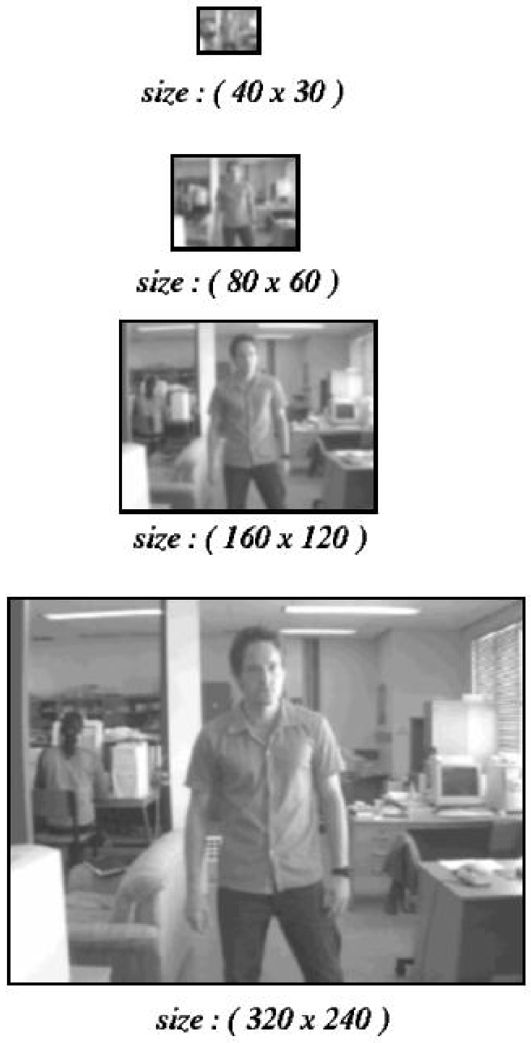

Figure 9: *"Resolution" pyramid consisting of 4 images.*

2.3. Ellipse fitting

One of the biggest problems encountered while tracking an object is the loss of features. Features can only be tracked while they are physically visible to the camera. The main causes of feature loss are the occlusion of features (caused by a rotation of the head) and sudden changes in light intensity.

These problems are handled as follows: During the initial feature selection an ellipse is statistically fitted around the features on the subjects head. The features inside the ellipse are the only ones which are tracked from image to image. As soon as any of the features within the ellipse disappear, new ones are generated so that the number (within a certain threshold) of trackable features remain the same. Figure 4 shows an ellipse plotted around the subject's head. This allows us to keep track of the region of interest.

3. Conclusion

Figures 10 to 15 shows a sequence of images produced by the KLT - tracking algorithm. The original images where captured at a frame rate of 20 frames per second. The figures shown are 10 frames apart in the original captured data. We can clearly make the conclusion that the Kanade - Lucas - Tomasi tracking algorithm is remarkably robust for tracking facial images against cluttered backgrounds.

4. References

[1] Stan Birchfield, Derivation of Kanade-Lucas-Tomasi Tracking Equation, Unpublished, May 1996.

[2] Carlo Tomasi and Takeo Kanade, Detection and Tracking of Point Features, Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

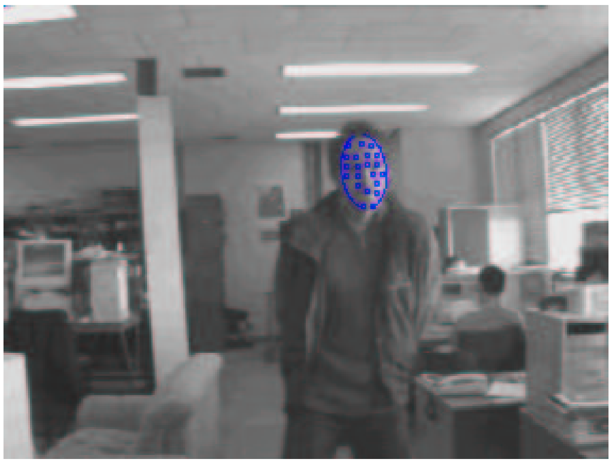
Figure 10: *Frame 0 in a sequence.*

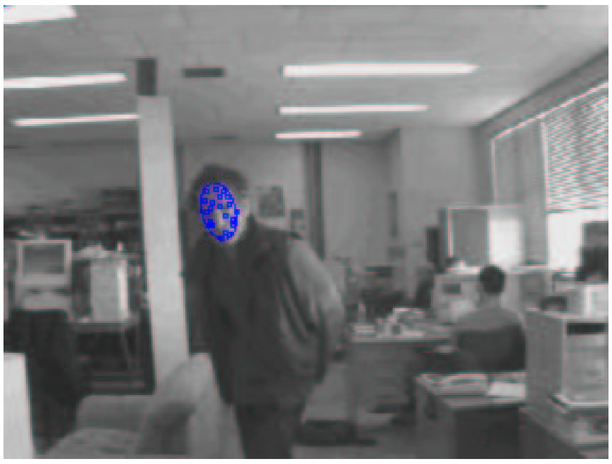
Figure 13: *Frame 3 in a sequence.*

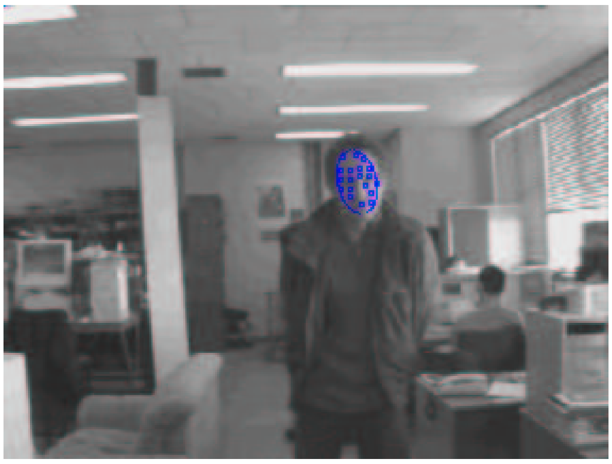
Figure 11: *Frame 1 in a sequence.*

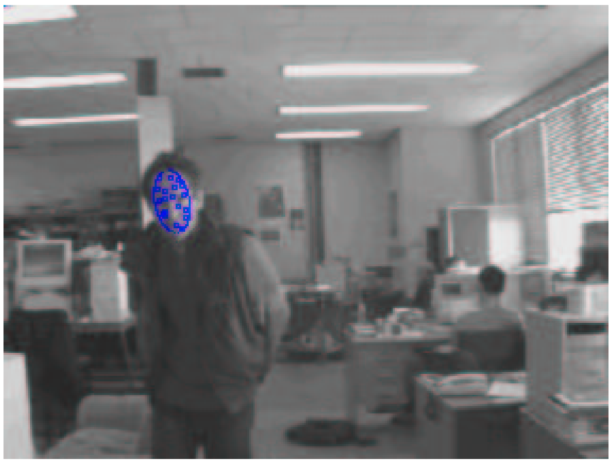
Figure 14: *Frame 4 in a sequence.*

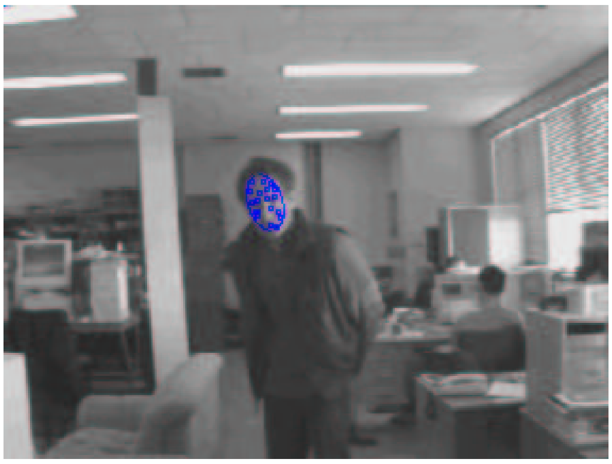
Figure 12: *Frame 2 in a sequence.*

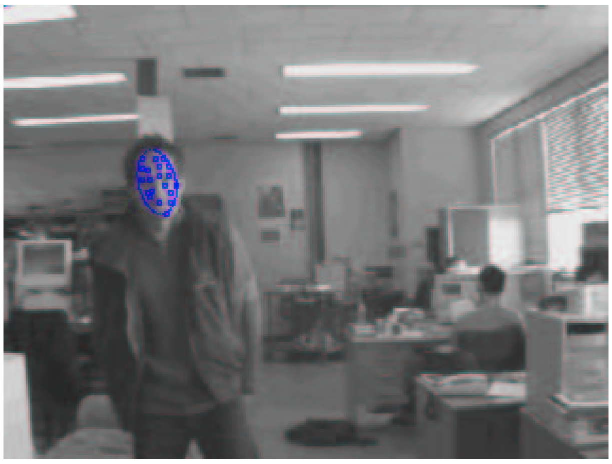
Figure 15: *Frame 5 in a sequence.*