Robust Visual Tracking from Dynamic Control of Processing

Alban Caporossi, Daniela Hall, Patrick Reignier and James L. Crowley*
PRIMA-GRAVIR, INRIA Rhône-Alpes, Montbonnot, France

Abstract

This paper presents a robust tracking system that employs a supervisory controller to dynamically control the selection of processing modules and the parameters used for processing. This system employs multiple pixel level detection operations to detect and track blobs at video rate. Groups of blobs can be interpreted as related components of objects during an interpretation phase. A central supervisor is used to adapt processing parameters so as to maintain reliable real time tracking. System performance is demonstrated on the PETS 04 data set.

1. Introduction

This paper presents an architecture for robust on-line tracking and interpretation of video streams. The system is based on a real time process managed by a supervisory controller. During each cycle, target blobs are observed and updated using simple pixel level detection processes. Detection procedures are then specified in a number of detection regions to detect new blobs. An evaluation phase is used to assess system performance and to adapt processing so as to maintain both reliability and real time (video rate) processing. An interpretation phase is then run to interpret groups of blobs as more abstract objects. Performance for this system is illustrated using the PETS 04 data set.

The paper starts with an overview of the system architecture. Section 3 describes the underlying principle of the core modules followed by technical details of the implementation. Section 4 describes a method for automatic adaption of the parameters necessary for the tracking system. The flexibility of the architecture is demonstrated in section 5. Section 6 evaluates the performance of this system on the PETS 04 data sets.

2. Architecture

Figure 1 shows the system architecture. The core of the tracking system is composed of a supervisor, a target initialisation module (*Detection Region*) and a tracking module (*Target Observation*). These modules are detailed in section 3.

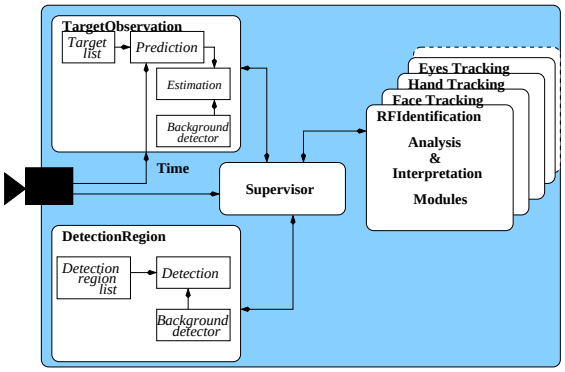

Figure 1. Visual tracking using a central supervisor architecture with core modules enables the flexible plug-in of higher level modules.

The supervisor acts as a process scheduler, sequentially executing modules in a cyclic process. Each cycle begins by acquiring the current image from an image buffering system (video demon). For each image, targets are tracked and new targets are detected. The supervisor enables a flexible integration of a several modules. During each cycle, for each target, the supervisor can call additional modules for analysis and interpretation as needed. During each cycle, the currently listed image processing operation for each target is applied to the target's region of interest. In this way, the appropriate image processing procedure can be changed and new image processing procedures can be added without changing the existing system architecture. Section 5 shows examples on this flexible architecture by adding modules for head and hand tracking, for eye detection and tracking and for general target identification.

*This research is supported by IST-CAVIAR 2001 37540

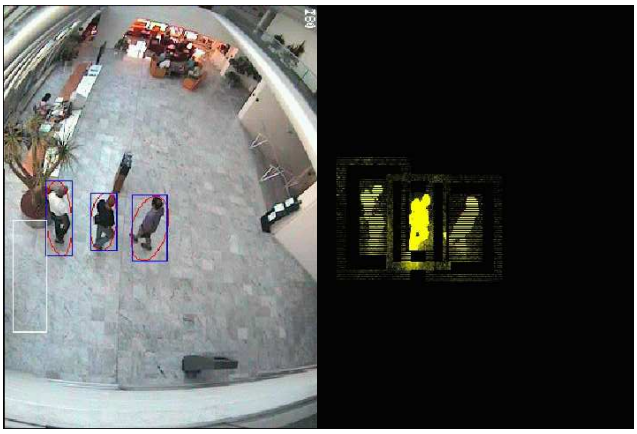

Figure 2. Target tracking by background differencing. The central person is tracked using all pixels whereas the two other persons are tracked using every second pixel.

3. The tracking system

In this section, we describe the theoretical aspects and the details on the actual implementation of the core tracking system.

3.1 Energy detection

Currently, targets can be detected by energy measurements based on background subtraction or intensity normalized color histograms. The background subtraction module computes a difference image I_d from the current frame $I = (I_{red}, I_{green}, I_{blue})$ and the background image $B = (B_{red}, B_{green}, B_{blue})$:

$$I_d = \tfrac{1}{3}\big(\mid I_{red} - B_{red} \mid + \mid I_{green} - B_{green} \mid + \mid I_{blue} - B_{blue} \mid \big)$$

The background image B is updated with each frame using a weighted averaging technique, with a strong weight applied to the previous background, and a small weight applied to the current image. This procedure constitutes a simple first order recursive filter along the time axis for each pixel. The background image is only updated for those pixels that do not belong to one of the target ROIs.

$$B_t(i,j) = \begin{cases} \alpha I_t(i,j) + (1-\alpha) B_{t-1}(i,j), (i,j) \in \text{bg} \\ B_{t-1}(i,j), \text{else} \end{cases} \quad (1)$$

Figure 2 shows an example of target tracking by background subtraction. The right image represents the background difference image I_d after processing of three ROI's. Three targets can be clearly identified. Notice that the center target appears as solid white, while the adjacent targets appear to be "hashed". This is the result of optimization that allows the processing to be applied to every Nth pixel. In this example, the two adjacent regions were processed with $N = 2$, while the center target was processed with $N = 1$. N is determined dynamically during each cycle by the process supervisor.

The position and extent of a target are determined by the moments of the detected pixels in the difference image I_d within the ROI. The center of gravity (or first moment) gives the position of a target. The covariance (or second moment) determines the spatial extent, and can be used to determine width, height, and slant of a target. These parameters also provide the target's search region in the next image.

Chrominance information can be used to provide probabilistic detection of targets. The intensity for each RGB color pixel within a ROI is normalized to separate chrominance from luminance.

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B} \quad (2)$$

These color components have the property to be robust to intensity variations [6].

The probability that a pixel takes on a particular color can be represented as a histogram of (r, g) values. The histogram h_T of chrominance values for a target, T, provides an estimate of the probability of a chrominance vector (r, g) given the target $p(r, g|T)$. The histogram of chrominance for all pixels h_{total} gives the global probability $p(r, g)$ of encountering a chrominance among the pixels. The probability of a target is the number of pixels of the target divided by the total number of pixels. Putting these values into Bayes rule shows that an estimate of the probability of the target for each pixel can be obtained by evaluating the ratio of the target histogram divided by the global histogram.

$$p(T|r,g) = \frac{p(r,g|T)p(T)}{p(r,g)} \approx \frac{h_T(r,g)}{h_{total}(r,g)} \quad (3)$$

For each image, a probability map, I_p, can be created by evaluating the ratio of histograms for each pixel in the image. Figure 3 shows an example of face detection using a ratio of chrominance histograms. The bottom image displays the probability map I_p. The probability map is only evaluated within the search region provided by the Kalman filter in order to increase processing speed.

A common problem in both background subtraction and histogram detection are spatial outliers. In order to increase the stability of target localization, we suppress the contribution of outliers using a method proposed by Schwerdt in [5]. With this method, the probability image I_p is multiplied by

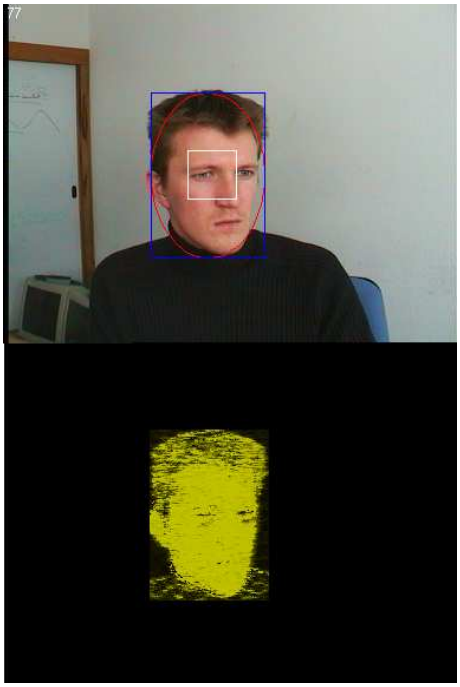

Figure 3. Target detection by normalized color histogram.

a Gaussian weighting function centered at the predicted target position. This corresponds to a filtering by a strong positional prior. The effect is that spatial outliers lose their influence on position and extent as a function of distance from the predicted Gaussian. In order to save computation time, this operation is performed only within the region of interest R of each target. Even for small regions of interest this operation stabilizes the estimated position and extent of targets.

$$I'_p = \begin{cases} I_p * G(\mu, \Sigma), (i,j) \in R \\ 0, \text{else} \end{cases} \tag{4}$$

where

$$G(\vec{x}; \mu, \Sigma) = e^{-\frac{1}{2}(\vec{x}-\mu)^T \Sigma^{-1} (\vec{x}-\mu)} \tag{5}$$

The center of gravity $\mu = [\widehat{x}_t^-, \widehat{y}_t^-]^T$ is the Kalman prediction of the target location. The spatial covariance Σ reflects the size of the target as well as the growing uncertainty about the current target size and location. The same principle can be applied to the background difference I_d.

3.2 Tracking process

The tracking system is a form of Kalman filter [7]. The state vector for each target is composed of position and velocity. The current target state vector $\widehat{x}_{t-1}$ is used to make a new prediction according to :

$$\widehat{x}_t^- = \Phi_t \widehat{x}_{t-1}, \text{ with } \Phi_t = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \tag{6}$$

and Δt the time difference between two iterations.

From the new position measurement z_t, estimation update is carried out.

$$\widehat{x}_t = \widehat{x}_t^- + K_t(z_t - H_t \widehat{x}_t^-) \tag{7}$$

This relation is important for balancing the estimation between measurement and prediction with the Kalman gain K_t. The estimated precision is a diagonal covariance matrix

$$P_t^- = \begin{bmatrix} \widehat{\sigma}_{xx}^2 & 0 & 0 & 0 \\ 0 & \widehat{\sigma}_{yy}^2 & 0 & 0 \\ 0 & 0 & \widehat{\sigma}_{v_{xx}}^2 & 0 \\ 0 & 0 & 0 & \widehat{\sigma}_{v_{yy}}^2 \end{bmatrix} \tag{8}$$

and is predicted by:

$$P_t^- = \Phi_{t-1} P_{t-1} \Phi_{t-1}^T + Q_{t-1} \tag{9}$$

where Q_{t-1} is the covariance matrix of the prediction error which represents the growth of the uncertainty in the current target parameters.

3.3 The core modules

The tracking process has been implemented in the ImaLab environment [4]. This environment allows real-time processing of frames extracted from the video stream. The basic tracking system is composed of two modules:

- *TargetObservation* predicts for each target the position in the current frame by a Kalman filter and then computes its real position by background subtraction or color histogram detection.
- *DetectionRegion* detects new targets by analysing the energy (background differencing or color histogram) within several manually defined detection regions.

Figure 1 shows the system architecture. Both core modules can be instantiated to use either background differencing or color histogram. For the PETS 04 experiments, we use tracking based on background subtraction.

3.4 Target initialization module

Detection regions are image regions where new targets can appear. Restricting detection of new targets to such regions allows the system to reduce the overall computing time. As a side effect, the use of detection regions also provides a reduction in the number of spurious false detections

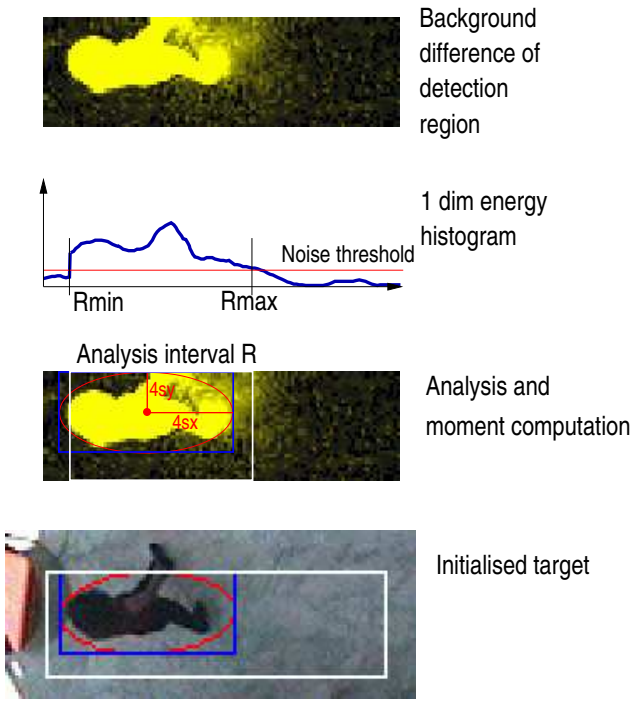

Figure 4. Initialisation of new target.

by avoiding detection in unlikely regions, but targets might be missed when the detection regions are not chosen appropriately.

For each scenario a different set of detection regions is determined. Currently, these regions are selected by hand. An automatic algorithm appears to be relatively easy to imagine. New targets are initialized automatically by analysing the detection regions in each tracking cycle. This analysis is done in two steps. In the first step, the subregion which is occupied by the new target is determined by creating a 1 dimensional histogram along the long axis of the detection region. The limits of the target subregion are characterized by an interval, R_{min}, R_{max}, whose values of the one dimensional histogram are above a noise threshold (see Figure 4). In the second phase, the energy density within the so specified subregion R is computed as

$$e_R = \frac{1}{|R|} \sum_{(i,j)\in R} I_d(i,j) \tag{10}$$

with $|R|$ number of pixels of R. A new target with mean μ_R and covariance Σ_R is initialised when the measured energy density e_R exceeds a threshold. This approach has the advantage, that targets can be detected independently of the size of the detection region.

3.5 Tracking module

The module *TargetObservation* implements the target tracking. The supervisor maintains a list of current targets. Targets of this list are sequentially updated by the supervisor depending on the feedback of the modules. For each target, a new position is predicted by a first order Kalman filter. This prediction determines a search region within which the target is expected to be found. A target is found by applying the specified detection operation to the search region. If the average target detection energy is above a threshold, the target observation vector is updated. This module depends on following parameters:

- Detection energy threshold: this represents the average energy threshold validating the target existence.
- Sensitivity threshold : this parameter thresholds the energy image (I_d in case of background differencing or I_p in case of chrominance detection). If the value is 0, the raw data of the energy image is used.
- Target area threshold: A step size parameter N enables faster processing for large targets by processing only 1 out of N pixels. When the target surface is larger than a threshold, N is increased. This temporary measure will be replaced by a more sophisticated control logic based on computing time. Figure 2 illustrates the use of this parameter.

3.6 Split and merge of targets

In real world video sequences, especially in the domain of video surveillance, it often happens that targets come together, move in the same direction for a while and then separate. It can also occur that close targets occlude each other. In that case only one target is visible at the time, but both targets are still present in the scene. To solve such problems, we use a method that allows merging and splitting of targets. This method enables to keep track of occluded targets and also to model common behavior of a target group. The PETS 04 sequences contain many examples of such group behavior.

A straight forward approach is applied for the detection of target split and merge. Merging of two targets that are within a certain distance from each other is detected by evaluating following equation:

$$c/(a+b) < threshold \tag{11}$$

where c is the distance between the gravity centers of both targets, a and b are the distances between the center of gravity and the boundary of the ellipse defined by the covariance of the respective target(see Figure 5 (left)). In our implementation we use a $threshold = 0.8$.

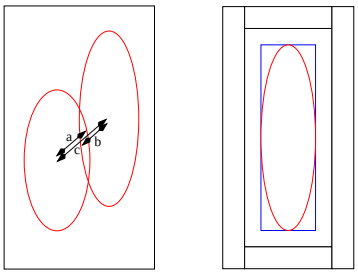

Figure 5. (left) Merging of targets as a function of the target relative position and size. (right) Splitting detectors are defined proportionally to the target size.

Splitting of targets is implemented by placing detection regions around the target as shown in Figure 5 (right). The size and location of the split detection regions are proportional to the target size. Within each split detection region, the average enery is evaluated in the same way as in the target initialisation module. A new target is created if this average energy is greater than the threshold $u = \textit{energy density} * \textit{split coefficient}$. The parameter *split coefficient* controls the constraints for target splitting.

4. Automatic parameter adaption

Target initialization and tracking by background differencing or histogram detection requires a certain number of parameters, as mentioned in the previous sections (detection energy threshold, sensitivity, density energy threshold, α, split coefficient, area threshold).

In order to preserve the re-usability of the tracking module and guarantee good performance in a wide range of different tracking scenarios, it is crucial to have a good parameter setting at hand. Up to now, parameter adaption is done manually. This is a very tedious job which might need frequent repetition when the scene setup has changed.

In this section we propose a first attempt of a module that automatically finds a good parameter setting. As a first step, we consider the tracker as a classical system with control parameters and noise perturbations (see Figure 6). The system produces an output $y(t)$ that depends on the input $r(t)$, some noise $d(t)$, and a set of parameters that affect the control module K [1].

4.1 Algorithm

First we need to explore the effect of particular parameters on the system. The goal of this step is to identify the important parameters, their relation and eventually discard

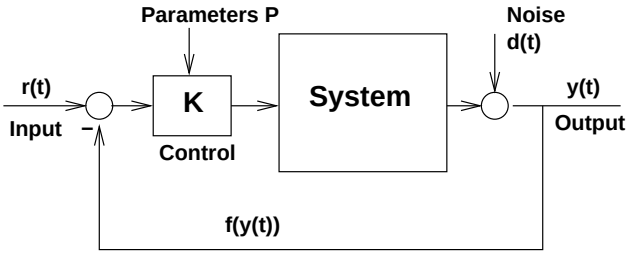

Figure 6. A controlled system

parameters with little effect. For a sequence for which the ground truth $r(t)$ is available we vary the parameters systematically and measure the output of the system, $y_{P_k}(t)$ for a particular parameter setting P_k in the parameter space P. $y_{P_k}(t)$ and $r(t)$ are split in m sections according to m intervals $s_i = [t_{i-1}, t_i], i = 1, \ldots, m$.

For each parameter setting P_k and each interval $r(s_i)$ and $y_{P_k}(s_i)$ are known. From these input/output correspondences we can compute the transfer function $f(y_{P_k}(s_i)) = r(s_i)$ by a least squares approximation. The overall error of the transfer function on the sequence is computed as follows:

$$\epsilon = ||r(t) - f(y_{P_k}(t))|| = \sum_{s_i} ||r(s_i) - f(y_{P_k}(s_i))|| \quad (12)$$

For each P_k, we determine the transfer function that minimizes this error. The average error ($\bar{\epsilon} = \epsilon/n$, n number of frames) is used to characterize the performance of the system with the current parameter setting. This is a very coarse approximation, but as we will see, the average error evolves smoothly over the parameter space.

We consider polynomial transfer functions of first and second order (linear and quadratic) of the following form

$$\vec{r}(t_k) = A_0\vec{y}(t_k) + \vec{b} \quad (13)$$

$$\vec{r}(t_k) = A_2(\vec{y}(t_k))^2 + A_1\vec{y}(t_k) + \vec{b} \quad (14)$$

with transfer matrices A_i and offset $\vec{b}$.

The measurements have either two or four dimensions. In the two dimensional case, the measurements contain the coordinates of the center of gravity of the target. The four dimensional case also contains the height and width of the target bounding box. We could have considered an additional dimension for the target slant, but we discarded this possibility due to the discontinuity of the slant measurement at $180°$.

The linear transfer function estimated from the data of the sequences *Walk1.mpeg* and *Walk3.mpeg* produce good results. We observe a transfer matrix A_0 that is close to identity. The quadratic transfer function has a smaller $\bar{\epsilon}$, but the transfer matrix A_2 has very low values and is therefore

not significant. This means that the linear transfer function is a good model for our system.

4.2 Exploration of the parameter space

The average error of the best transfer function evaluated on the entire test sequence is used to characterize the performance of the controlled system. The parameter space can be very high dimensional. Therefore, exploring the entire space can be time consuming. To cope with this problem we assume that some parameters evolve independently from each other. This allows to restrict the search of an optimal parameter value to a low dimensional hyperspace. In the experiment we use following default values for the constant parameters of the hyperspace: detection energy = 10, density = 15, sensitivity = 20, split coefficient = 2.0, α = 0.001, area threshold = 1500. We experiment on sequence *Walk1.mpeg* except for figure 7.

Figure 7 shows the surface produced by varying the detection energy threshold and the sensitivity threshold simultaneously. Figure 8 shows the error evolution by varying the split coefficient and the sensitivity. The optimal parameter value is different for each sequence. This means that the parameters are sequence dependent. In all cases the error evolves smoothly. This means that we are dealing with a controlled system and not with a system following chaotic or arbitrary rules.

Figure 9 (left) provides evidence to set $\alpha = 0.1$. Figure 9 (right) shows that the density threshold has no effect on the average error. This parameter is therefore a candidate that needs not be considered for further exploration of the parameter space.

Figure 10 shows the effect of the parameter area threshold. This parameter treats one pixel out of two for targets that are larger than area threshold pixels. This explains the increase of the error for small thresholds and the speed up in processing time. It is interesting to see, that the error increase is very small, less than 4% error increase for a 25% gain in processing time. Our method allows to identify this kind of relations between parameters.

4.3 Summary

We have shown a method to evaluate the performance of a system controlled by a set of parameters. The average error is used to understand the effect of single parameters and parameter pairs. This method allows to verify that our tracking system has a controlled behavior. We identified that the density parameter has no effect on the error performance and it can be removed from the parameter space. The area threshold parameter influences the overall processing time and the average error. With our method, we found that the increase in error is small with respect to the gain in processing time. This is an interesting result which a dynamic control system should take into account. The experiments show that the optimal parameter setting estimated from one sequence scenario must not be optimal for another sequence. This needs to be explored by evaluating more data sequences. Another important point is that the approach requires ground truth labelling. This means that our method can not find the optimal parameters when the ground truth is unknown. Likelihood may be appropriate in some cases to replace the ground truth, but the results will be inferior since the likelihood increases the noise perturbations.

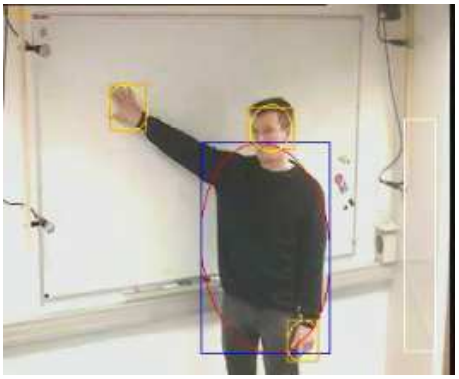

Figure 11. Modules for face and hand observation are plugged into tracking system.

5. Tracking : optional higher level modules

In this section we demonstrate the flexibility of our tracking system. The proposed architecture enables easy plug in of higher level modules which enables the system to solve quite different tasks.

5.1. Face and hand tracking for human computer interaction

Modules for face and hand tracking use color histogram detection. Face and hands are initialised automatically with respect to a body detected by background differencing. This means that the same tracking principle is applied to faces and hands at a higher level. An example is shown in Figure 11.

5.2. Eye detection for head pose estimation

This module detects facial features by evaluating the response to receptive field clusters [2]. The method detects facial features robust to scale, lighting variation, person and

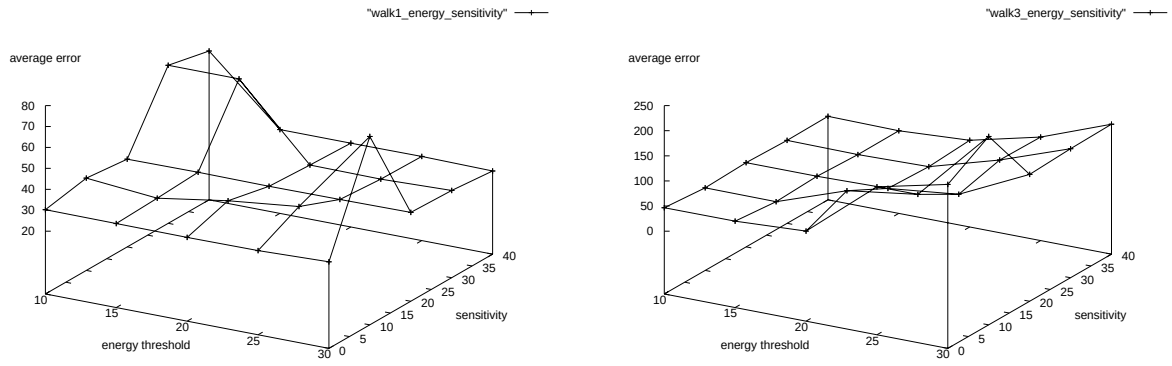

Figure 7. Evolution of the average error over detection energy threshold and sensitivity threshold (sequence Walk1.mpeg (left) and Walk3.mpg (right) and default values for free parameters).

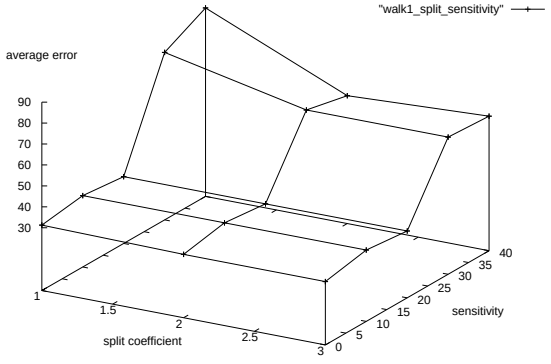

Figure 8. Evolution of the average error over split coefficient and sensitivity threshold.

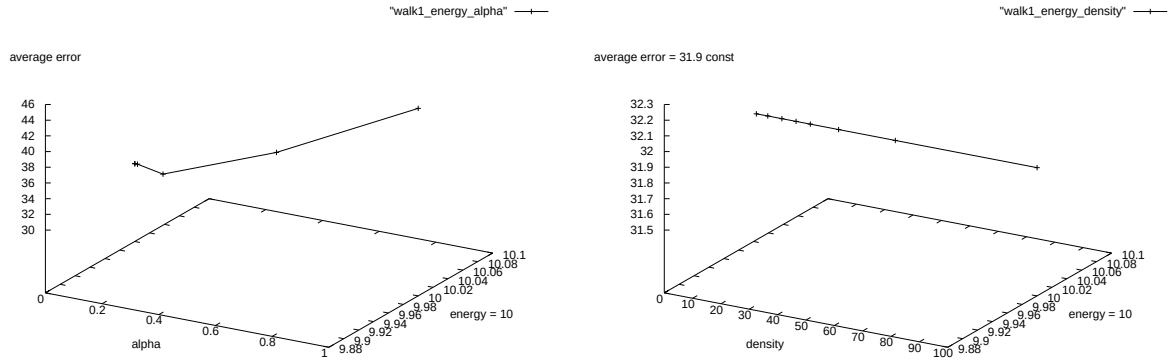

Figure 9. Evolution with varying alpha (left) and varying density (right). We can identify an optimal value for alpha ($\alpha = 0.1$), but the error is constant for all density values.

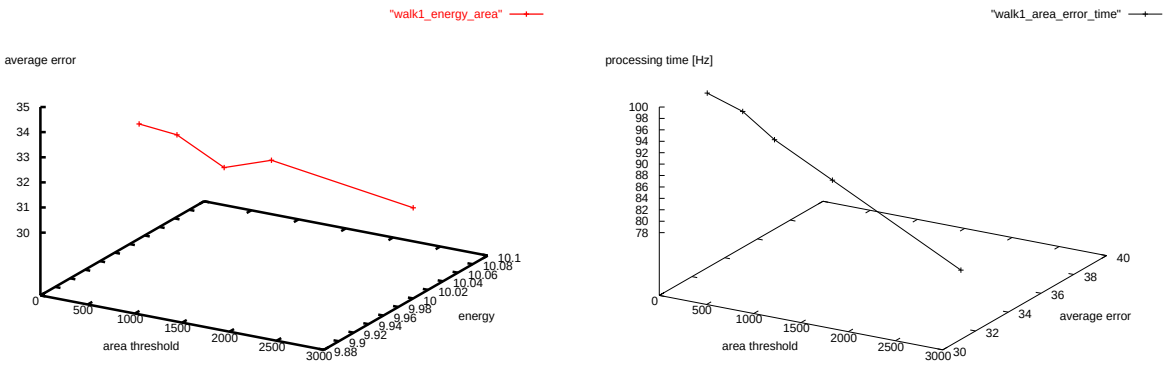

Figure 10. Evolution with varying area threshold (left). The error increases slightly with decreasing area threshold. The area threshold has a significant impact on the processing time (right).

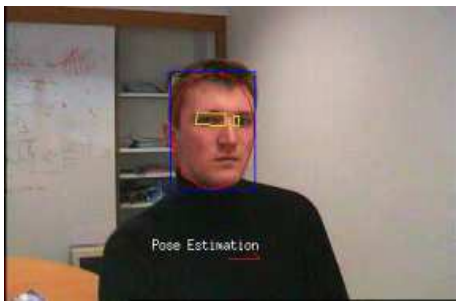

Figure 12. Real-time head pose estimation.

head pose. The tracking system provides the precise face location which allows the combined system to run in real time. Figure 12 shows an example of the eye tracking module.

5.3. Agent identification

The agent identification module provides an association between individual features and tracked targets by background subtraction. Identification of each tracked blob is carried out by elastic matching of labelled graphs where the labels are receptive field responses [2]. The degree of correspondence between the model and the observations extracted from the ROI provided by the tracking system is computed by evaluating a cost function. The cost function is a weighted sum of the spatial similarity and the appearance similarity [3, 8]. Figure 13 shows a successful identity recovery after a target occlusion. The system currently processes 10 frames/s.

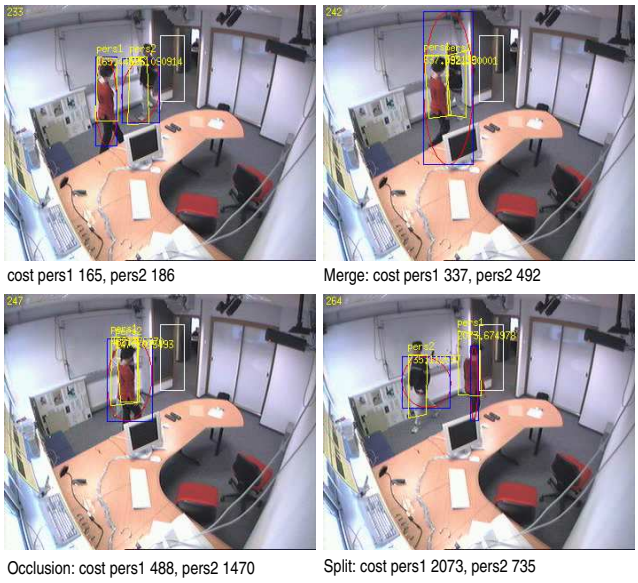

Figure 13. Example of a split and merge event with successful identity recovery.

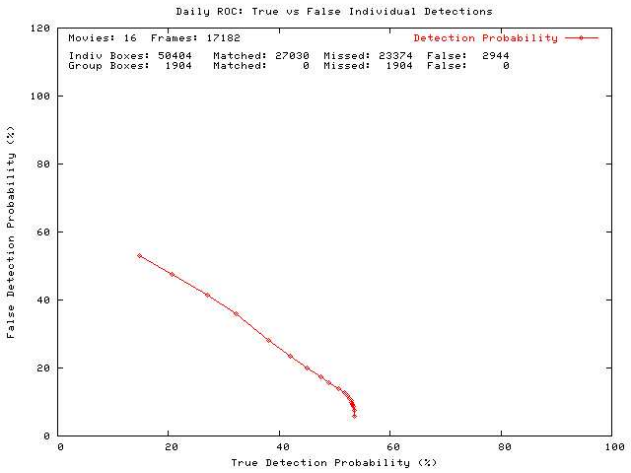

Figure 14. True versus False detections for individuals

6. Tracking performance of the core modules

In order to evaluate the performance of our tracking system, we have tested the core modules on 16 of the PETS 04 sequences (17182 frames containing 50404 targets marked by bounding boxes)[1]. In this section we give a brief summary of the tracking results.

Figure 14 shows the receiver operator curve for all 16 sequences. Our system has a low false detection probability of 9.8% and a true detection probability of 53.6%. This translates to a recall of 53.6% (27030 correct positives out of 50404 total positives) and a precision of 90.2% (27030 correct positives out of 29974 detections). The reason for the relatively low recall is the fact that the ground truth labeling takes into account targets that are already present in the scene and targets that pass on the gallery at the first floor. Our tracking system relies on the method of detection region for target initialization. Both type of targets are not detected by our tracking system, because they are not initialized.

The tracking results are evaluated with respect to other parameters such as errors in detected position, size, and orientation, the time lag of entry and exit. The performance of our system with respect to these parameters is summarized in Table 1. Our system performs very well in position detection, orientation estimation and exit time lag. The bounding box produced by the tracking system is significantly smaller than the bounding box of the ground truth. This is due to the fact that the tracking system estimates the bounding box from the covariance of the pixels with high energy whereas a human draw a bounding box that includes all pixels that belong to the target. The tracking system can produce a similar output by computing the connected components of the energy image. This is a costly operation. In the case where the connected components bounding box is used for position computation, the position become more unstable. For this reason we decided to use the first and second moments of energy pixels for target specification. The entry time lag is a problem related to the detection region. A human observer marks a new target as soon as it occurs. The detection region requires that the observed energy is above the energy density threshold.

[1]The sequences as well as the statistics are available at the CAVIAR home page http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm

Average error in	average value	maximum value
Position	6 - 7 pixels	13 - 15 pixels
Size	-160% to -240%	-240%
Orientation	$\pm 0.5\%$	$\pm 30\%$
Entry time lag	50 to 80 frames	100 to 160 frames
Exit time lag	1 frame	1 frame

Table 1. Evaluation of the tracking results with respect to measurement precision.

7. Conclusion

We have presented an architecture for a tracking system that consists of a central supervisor, a tracking module based on background subtraction or color histogram detection combined with Kalman filtering and an automatic target initialization module restricted to detection regions. These three modules form the core system. The central supervisor architecture has the advantage that additional modules can be plugged in very easily. New tracking systems can be created in this way that can solve different tasks.

The tracking system depends on a number of parameters that influence the performance of the system. Therefore, finding a good parameter setting for a particular scenario is essential. We have proposed to consider the tracking system as a classical controlled system and identified a method to evaluate the quality of a particular parameter setting. The preliminary experiments show that small variations of the parameters produce smooth changes of the average error function. Using this behavior, we can improve the performance of our tracking system by finding a good parameter setting using gradient descend in the parameter space. Unfortunately, the experiments on the automatic parameter adaption are preliminary and could not yet be integrated in the performance evaluation of the system.

References

[1] P. de Larminat. *Automatique commande des systèmes lineaires*. Hermes Science Publications, 2nd edition, 1996.

[2] D. Hall and J.L. Crowley. Détection du visage par caractéristiques génériques calculées à partir des images de luminance. In *Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle*, pages 1365–1373, Toulouse, France, 2004.

[3] M. Lades, J.C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Mahlsburg, R.P. Würz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *Transactions on Computers*, 42(3):300–311, March 1993.

[4] A. Lux. The imalab method for vision systems. In *International Conference on Vision Systems*, pages 319–327, Graz, Austria, April 2003.

[5] K. Schwerdt and J.L. Crowley. Robust face tracking using color. In *International Conference on Automatic Face and Gesture Recognition*, pages 90–95, Grenoble, France, March 2000.

[6] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[7] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, 2004.

[8] L. Wiskott, J.M. Fellous, N. Krüger, and C. von der Mahlsburg. *Face Recognition by Elastic Bunch Graph Matching*, chapter 11, pages 355–396. Intelligent Biometric Techniques in Fingerprint and Face Recognition. CRC Press, 1999.