Automatic parameter regulation for a tracking system with an auto-critical function

Daniela Hall
INRIA Rhône-Alpes, St. Ismier, France
Email: Daniela.Hall@inrialpes.fr

Abstract— **In this article we propose an architecture of a tracking system that can judge its own performance by an auto-critical function. Performance drops can be detected which trigger an automatic parameter regulation module. This regulation module is an expert system that searches a parameter setting with better performance and returns it to the tracking system. With such an architecture, a robust tracking system can be implemented which automatically adapts its parameters in case of changes in the environmental conditions. This article opens a way to self-adaptive systems in detection and recognition.**

I. Introduction

Parameter tuning of complex systems is often performed manually. A tracking system requires different parameter settings as a function of the environmental conditions and the type of the tracked targets. Each change in condition requires a parameter update. There is a great need to design an expert system that performs the parameter regulation automatically. This article proposes an approach and applies it to a real-time tracking systems. The here proposed architecture for auto-regulation is valid for any complex system whose performance depends on a set of parameters.

Automatic regulation of parameters can significantly enhance performance of systems for detection and recognition. Surprising little previous work has been published in this domain [5]. A first step towards performance optimization is the ability of the system to be auto-critical. This means that the system must be able to judge its own performance. A performance drop, detected with this kind of auto-critical function, can trigger an independent module for auto-regulation. The task of the regulation module is to propose a set of parameters to improve system performance.

The auto-critical function detects a performance drop when the measurements with respect to a scene reference model diverge. In this case the automatic regulation module is triggered to provide a parameter setting with better performance. Section II explains the architecture of the tracking system and the architecture of the regulation cycle. Section III explains the details of the auto-critical function, the generation of the scene reference model and the measure used for performance evaluation. In section IV we explain the use of the regulation module. We then show experiments that demonstrate the utility of our approach. We finish with conclusions and a critical evaluation.

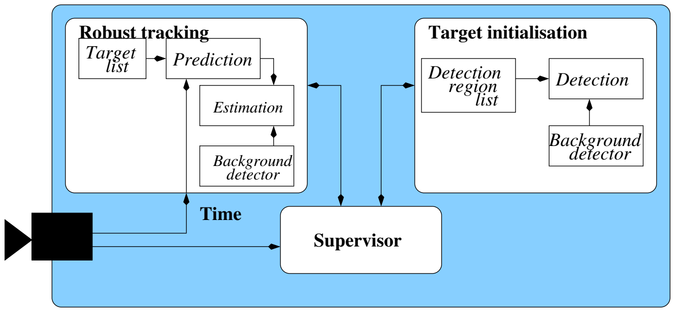

Fig. 1. Architecture of the tracking and detection system controlled by a supervisor.

II. System architecture

In order to demonstrate the utility of our approach for auto-regulation of parameters we choose a detection and tracking system as previously described in [2]. Figure 1 shows the architecture of the system. The tracking system is composed of a central supervisor, a target initialisation module and a tracking module. This modular architecture is flexible such that competing algorithms for detection can be integrated. For our experiments we use a detection module based on adaptive background differencing using manually defined detection regions. Robust tracking is achieved by a first order Kalman filter that propagates the target positions in time and updates them by measurements from the detection module.

The tracking system depends on a number of parameters such as detection energy threshold, sensitivity for detection, energy density threshold to avoid false detections due to noise, a temporal parameter for background adaptation, and a split coefficient to enable merging and splitting of targets (i.e. when two people meet they merge to a single group target, a split event is observed when a person separates from the group).

Figure 2 shows the integration of the parameter regulation module and the auto-critical function. The auto-critical function evaluates the current system performance and decides if parameter regulation is necessary. If this is the case, the tracker supervisor sends a request to the regulation module. It provides the its current parameter setting and current performance as well as other data that is needed by the regulation module. When the regulation module has found a better parameter setting (or after a maximum number of iteration) it stops processing and sends the result to the system supervisor that

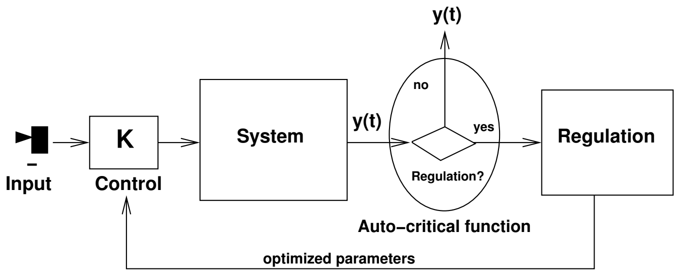

Fig. 2. Integration of the regulation module in a complex system.

updates the parameters and reinitialises the modules.

It is difficult to predict the performance gain of the auto-regulation. Since the module can test only a discrete number of parameter settings, there is no guarantee that the global optimal parameter setting is found. For this reason, the goal of the regulation system is to find a parameter setting that increases system performance. Subsequent calls of the regulation module allow then to obtain a constantly increasing system performance. The modular architecture enables the use of different methods and apply the regulation to different system kinds.

III. The auto-critical function

The task of the auto-critical function is to provide a fast estimation of the current tracking performance. A performance evaluation function requires a reliable measure to estimate the current system performance. The used measure (described in Section III-B) is based on a probabilistic model of the scene which allows to estimate the likelihood of measurements. The probabilistic scene model is generated by a learning approach. Section III-C explains how the quality of a model can be measured. Section III-D discusses different clustering schemes.

A. Learning a probabilistic model of a scene

A model of a scene describes what usually happens in the scene. It describes a set of target positions and sizes, but also a set of paths of the targets within the scene. The model is computed from previously observed data. A valid model allows to describe everything that is going to be observed. For this reason we require that the training data is representative for what usually happens in the scene.

The ideal model of a scene allows to decide in a probabilistic manner which measurements are typical and which measurements are unusual. With such a model we can compute the probability of single measurements and of temporal trajectories. Furthermore, we can detect outliers that occur due to measurement errors. The model represents the typical behaviour of the scene. Furthermore it enables the system to alert a user when unusual behavior takes place. This is a feature which is useful for the task of a video surveillance operator.

In this section we describe the generation of a scene reference model which gives rise to a goodness measure that can compute the likelihood of measurements $y(t_i)$ with respect to the scene reference model. We know that a single mode is insufficient to provide a valid scene description. We need a model with several modes that associate spatially close measurements and provide a locally valid model. The model is composed from data using a static camera.

An important question is which training data should be used to create an initial model. The CAVIAR test case scenarios [4] contain 26 image sequences and hand labelled ground truth. We can use the ground truth to generate an initial model. If the initial model is not sufficient, the model can be refined by adding tracking observations where the measurements with low probability which are likely to contain errors are removed.

For the computation of the scene reference model, we use the hand labelled data of the CAVIAR data set (42000 bounding boxes). We divide the model into a training and a test set of equal size. The observations consist of spatial measurements $\vec{y}_{spatial}(t_i) = (\mu_x, \mu_y, \sigma_x^2, \sigma_y^2)$ (first and second moments of the target observation in frame $I(t_i)$). We can extend these observations to spatio-temporal measurements $\vec{y}_{spatiotemp}(t_i) = (\mu_x, \mu_y, \sigma_x^2, \sigma_y^2, \Delta\mu_x, \Delta\mu_y, \Delta\sigma_x^2, \Delta\sigma_y^2)$ by considering observations at subsequent time instances t_i and t_{i-1}. Such measurements have the advantage that we take into account the local motion direction and speed. A trajectory $y(t)$ is a sequence of spatial or spatio-temporal measurements $\vec{y}(t_i)$. Single measurements are noted as vectors $\vec{y}(t_i)$ whereas trajectories $y(t)$ are coded as vector lists. The following approach is valid for both types of observed trajectories $y(t)$.

To obtain a multi-modal model we have experimented with two types of clustering methods: k-means and k-means with pruning. K-means requires a fixed number of clusters that must be specified by the user a priori. K-means converges to a local minimum that depends on the initial clusters. These are determined randomly, which means that the algorithm produces different sub-optimal solutions in different runs. To overcome this problem, k-means is run several times with the same parameters. In section III-C we propose a measure to judge the quality of the clustering result. With this measure we select an optimal clustering solution as our scene reference model.

The method k-means with pruning is a variation of the traditional k-means that produces stabler results due to subsequent fusion of close clusters. In this variation, k-means is called with a large number of clusters, $k \in [500, 2000]$. Clusters that are close within this solution are merged subsequently and clusters with few elements are considered as noise and removed. This method is less sensitive to outliers and has the characteristics of a hierarchical clustering scheme and at the same time can be computed quickly due to the initial fast k-means clustering. Figure 3 illustrates this algorithm.

B. Evaluating the goodness of a trajectory

A set of Gaussian clusters modeled by mean and covariance is an appropriate representation for statistical evaluation of measurements. The probability $P(\vec{y}(t_i)|C)$ can be computed according to equation 2.

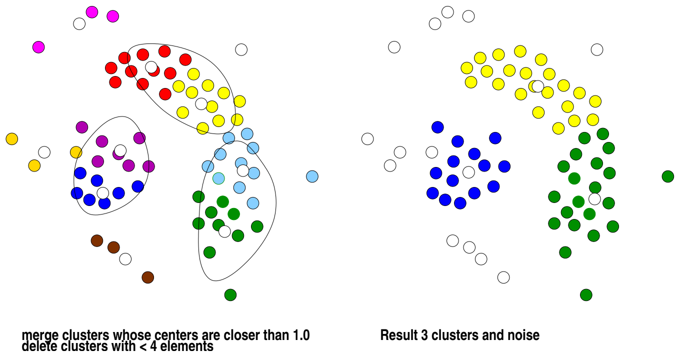

Fig. 3. K-means with pruning. After initial k-means clustering close clusters are merged and clusters with few elements are assigned to noise.

The auto-regulation and auto-critical module need a measure to judge the goodness of a particular trajectory. A simple goodness score consists of the average probability of the most likely cluster for the single measurements. The goodness $G(y(t))$ of the trajectory $y(t) = (\vec{y}(t_n), \ldots, \vec{y}(t_0))$ with length $n+1$ is computed as follows:

$$G(y(t)) = \frac{1}{n+1}\sum_{i=0}^{n}\max_k(P(\vec{y}(t_i)|C_k)) \tag{1}$$

with

$$P(\vec{y}(t_i)|C) = P(\vec{y}(t_i)|\vec{\mu};U) \tag{2}$$
$$= \frac{1}{(2\pi)^{dim/2}|U|^{1/2}}e^{(-0.5(\vec{y}(t_i)-\vec{\mu})^T U^{-1}(\vec{y}(t_i)-\vec{\mu}))}$$

with $\vec{\mu}$ mean and U covariance of cluster C. Trajectories have variable length and may consist of several hundred measurements. The proposed goodness score is high for trajectories composed of likely measurements and small for trajectories that contain many unlikely measurements (errors). This measure allows to reliably classify good and bad trajectories independent of their particular length.

On the other hand, the goodness score does not take into account the sequential structure of the measurements. The sequential structure is an important indicator for the detection of local measurement errors and errors due to badly adapted parameters. To study the potential of a goodness score that is sensitive to the sequential structure, we propose following measure (see equation 3).

$$G_{seq(v)}(y(t)) = \frac{1}{m}\sum_{i=0}^{m-1} log(\tilde{P}(y(s_i))) \tag{3}$$

which is the average log likelihood of the dominant term $\tilde{P}(y(s))$ of the probability of a sub-trajectory $y(s)$ of length v. We use the log likelihood because $\tilde{P}(y(s))$ is typically very small.

A trajectory $y(t) = (y(s_0), y(s_1), \ldots, y(s_{m-1}))$ is composed of m sub-trajectories $y(s_i)$ of length v. We develop the measure for $v = 3$, the measure for any other value v is developed accordingly. The probability of the sub-trajectories is defined as:

$$\begin{aligned} P(y(s_i)) &= P(\vec{y}(t_2), \vec{y}(t_1), \vec{y}(t_0)) \\ &= \tilde{P}(y(s_i)) + r \\ &= P(C_{k_2}|\vec{y}(t_2))P(C_{k_1}|\vec{y}(t_1))P(C_{k_0}|\vec{y}(t_0)) \\ &\quad P(C_{k_2}C_{k_1}C_{k_0}) + r \end{aligned} \tag{4}$$

$P(y(s_i))$ is composed of the probability of the most likely path through the modes of the model $\tilde{P}(y(s_i))$ plus a term r which contains the probability of all other path permutations. Naturally, the $P(y(s_i))$ will be dominated by $\tilde{P}(y(s_i))$, and r tends to be very small. This is the reason, why we use in the final goodness score only the dominant term $\tilde{P}(y(s_i))$. $P(C_{k_i}|\vec{y}(t_i))$ is computed using Bayes rule. The prior $P(C_k)$ is set to the ratio $|C_k|/(\sum_u |C_u|)$. The normalisation factor $P(\vec{y}(t_i))$ is constant. Since we are interested in the maximum likelihood, we compute:

$$\begin{aligned} P(C_{k_i}|\vec{y}(t_i)) &= \frac{P(\vec{y}(t_i)|C_{k_i})P(C_{k_i})}{P(\vec{y}(t_i))} \\ &\sim P(\vec{y}(t_i)|C_{k_i})\frac{|C_{k_i}|}{\sum_u |C_u|} \end{aligned} \tag{5}$$

where $|C_{k_i}|$ denotes the number of elements in C_{k_i}. $P(\vec{y}(t_i)|C_{k_i})$ is computed according to equation 2.

The joint probability $P(C_{k_2}C_{k_1}C_{k_0})$ is developed according to

$$P(C_{k_2}C_{k_1}C_{k_0}) = P(C_{k_2}|C_{k_1}C_{k_0})P(C_{k_1}|C_{k_0})P(C_{k_0}) \tag{6}$$

We simplify this equation by assuming a Markov constraint of first order:

$$P(C_{k_2}C_{k_1}C_{k_0}) = P(C_{k_2}|C_{k_1})P(C_{k_1}|C_{k_0})P(C_{k_0}) \tag{7}$$

To compute the conditional probabilities $P(C_i|C_j)$, we need to construct a transfer matrix from the training set. This can be obtained by counting for each cluster C_i the number of state changes and then normalise such that each line in the state matrix sums to 1. The probabilistically inspired sequential goodness score of equation 3 is computed using equations 4 to 7.

C. *Measuring the quality of the model*

K-means clustering is a popular tool for learning and model generation because the user needs to provide only the number of desired clusters [3], [7], [8]. K-means converges quickly to a (locally) optimal solution. K-means clustering starts from a number of randomly initialised cluster centers. Therefore, each run produces a different sub-optimal solution. In cases where the number of clusters is unknown, k-means can be run several times with varying number of clusters. A difficult problem is to rank the different k-means solutions and select the one that is the most appropriate for the task. This section provides a solution to this problem which is often neglected.

For a particular model (clustering solution) we can compute the probability of a measurement belonging to the model. To ensure that the computed probability is meaningful, the model must be representative. A good model assigns a high probability to a typical trajectory and a low probability to an unusual trajectory. Based on these notions we define an evaluation criteria for measuring the quality of the model. We need to have a model that is neither too simple nor too complex. The complexity is related to the number of clusters [1]. A high number of clusters tends to over-fitting and a low number of clusters provides an imprecise description.

Model quality evaluation requires a positive and negative example set. Typical target trajectories (positive examples) are provided within the training data. It is more difficult to create a negative example. A negative example trajectory is constructed as follows. First we measure the mean and variance of all training data. This represents the distribution of the data. We can now generate random measurements by drawing from this distribution with a random number generator. The result is a set of random measurements. From the training set, we generate a k-means clustering with a large number of clusters (K=100). For each random measurement we compute $p(\vec{y}(t_i)|model_{100})$. From the original random 5000 measurements we keep the 1200 measurements with the lowest probability. This gives the set of negative examples. Figure 4 shows an example of the positive and negative trajectory as well as the hand labelled ground truth and a multi-modal model obtained by k-means with pruning.

For any positive and negative measurements we compute the probability $P(\vec{y}(t_i))$. Classification of the measurements in positive and negative can be obtained by thresholding this value. For a threshold d the classification error can be computed according to equation 8. The optimal threshold, d, separates positive from negative measurements with a minimum classification error [1].

$$\begin{aligned} P_d(error) &= P(x \in R_{bad}, C_{pos}) + P(x \in R_{good}, C_{bad}) \qquad (8) \\ &= \int_0^d p(x|C_{good})P(C_{good})dx + \int_d^1 p(x|C_{bad})P(C_{bad})dx \end{aligned}$$

with $R_{bad} = [0, d]$ and $R_{good} = [d, 1]$.

We search the optimal threshold d such that $P_d(error)$ is minimised. We operate on a histogram using logarithmic scale. This has the advantage that the distribution of lower values is sampled more densely. The optimal threshold d with minimum classification error can be estimated precisely with the method.

This classification error $P(error)$ is a measurement for the quality of the cluster model. Furthermore, less complex models should be preferred. For this reason we formulate the quality constraint of clustering solutions as follows: the best clustering has the lowest number of clusters and an error probability $P(error) < q$ with $q = 1\%$. The values of q are chosen depending on the task requirements. This measure is a fair evaluation criteria which enables to choose the best model among a set of k-means solutions.

D. Clustering results

We test two clustering methods: K-means and k-means with pruning. The positive trajectory is a person walking across the hall, the negative trajectory consists of 1200 measurements constructed as described above. The training set consists of 21000 hand labelled bounding boxes from 15 CAVIAR sequences (see Figure 4).

Table I shows characteristics of the winning models with highest quality defined by minimum classification error and minimum number of clusters. The superiority of the k-means with pruning is demonstrated by the results. For the constraint $P(error) < 1\%$, k-means with pruning requires only 20 or 19 clusters respectively whereas the classical k-means needs a model of clusters to obtain the same error rate. The best overall model is obtained for spatio-temporal measurements using k-means with pruning.

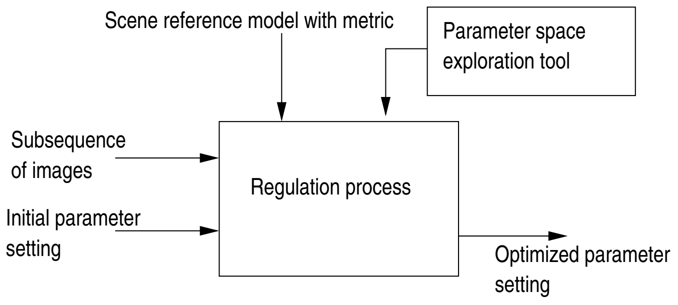
Scene reference model with metric
Parameter space exploration tool
Subsequence of images
Initial parameter setting
Regulation process
Optimized parameter setting

Fig. 5. A process for automatic parameter regulation.

IV. The module for automatic parameter regulation

The task of the module for automatic regulation is to determine a parameter setting that improves the performance of the system. In the case of a detection and recognition system, this corresponds to increasing the number of true positives and reducing the number of false positives. For this task, the module requires an evaluation function of the current output, a strategy to choose a new parameter setting and a subsequence which can be replayed to optimize the performance.

A. Integration

When the parameter regulation module is switched on, the system tries to find a parameter setting that performs better than the current parameter setting on a subsequence that is provided by the tracking system. The system uses one of the goodness scores of section III-B.

In the experiments we use a subsequence of 200 frames for auto-regulation. The tracker is run several times with changing parameter settings on this subsequence and the goodness score of the trajectory is measured for each parameter setting. The parameter setting that produces the highest goodness score is kept. Parameter settings are obtained from a parameter space exploration tool whose strategies are explained in the section IV-B and IV-C.

The automatic regulation can only operate on sequences that produce a trajectory (something observable must happen in the scene). To allow a fair comparison, the regulation module must process the same subsequence several times. For this

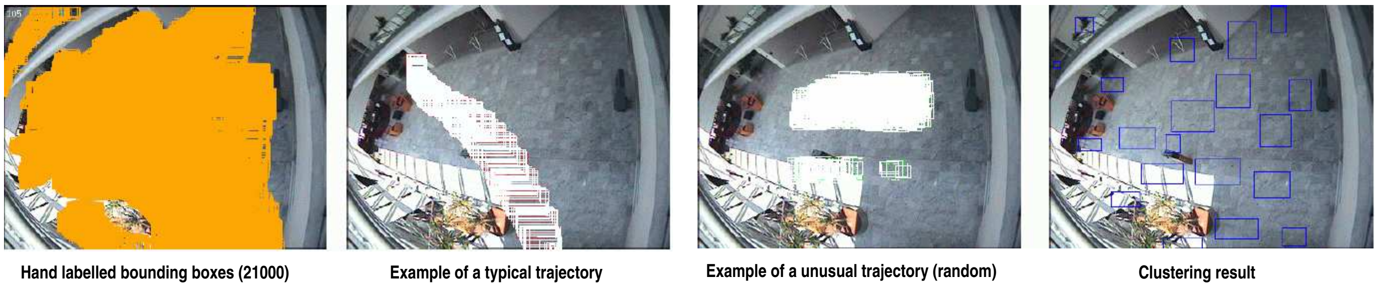
Hand labelled bounding boxes (21000)
Example of a typical trajectory
Example of a unusual trajectory (random)
Clustering result

Fig. 4. Ground truth labelling for the entrance hall scenario, examples of typical and unusual trajectories and clustering result using k-means with pruning.

Measurement type	Clustering method	# clusters	optimal threshold d	$P(error)$
Spatial	K-means	35	0.0067380	0.0007
	K-means with pruning	20	0.0067380	0.0061
Spatio-temporal	K-means	35	0.00012341	0.0013
	K-means with pruning	19	0.00012341	0.0034

TABLE I
BEST MODEL REPRESENTATIONS AND THEIR CHARACTERISTICS (FINAL NUMBER OF CLUSTERS, OPTIMAL THRESHOLD, AND CLASSIFICATION ERROR).

reason the regulation process requires a significant amount of computing power. As a consequence, the regulation module should be run on a different host such that the regulation does not slow down the real time tracking.

B. *Parameter space exploration tool*

To solve the problem of the parameter space exploration we propose a parameter exploration tool that provides the next parameter setting to the regulation module. The dimensions of the parameter space as well as a reasonable range of the parameter values are given by the user. In our tracking example the parameter space is spanned by detection energy, density, sensitivity, split coefficient, α, and area threshold.

In the experiments we tested two strategies for parameter setting selection. An enumerative method, that defines a small number of discrete values for each parameter. At each call the parameter space exploration tool provides the next parameter setting in the list. The disadvantage of this method is that only a small number of settings can be tested and the best setting may not be in the predefined list. The second strategy for parameter space exploration is based on a genetic algorithm. We found genetic algorithms perfectly adapted to our problem. It enables feedback from the performance of previous settings. We have a high dimensional feature space which makes hill climbing methods costly, whereas genetic algorithms explore the space without need of a high dimensional surface analysis.

C. *Genetic algorithm for parameter space exploration*

Among the different optimization schemes that exist we are looking for a particular method, that fulfills several constraints. We are not requiring to reach a global maximum of our function, but we would like to reach a good level of performance quickly. Furthermore we are not particularly interested in the shape of the surface in parameter space. We are only interested in obtaining a good payoff with a small number of tests. According to Goldberg [6], these are exactly the constraints of an application for which genetic algorithms are appropriate.

Hill climbing methods are not feasible because the estimation of the gradient of a single point in a 6 dimensional space requires 2^6 tests. Testing several points would therefore require a higher number of tests than we would like.

Genetic algorithms are inspired by the mechanics of natural selection. Genetic algorithms require an objective function to evaluate the performance of an individual and a coding of the input variables. Typically the coding is a binary string. In our example, each parameter is represented by 5 bit, which gives an input string of length 30.

Genetic algorithms have three major operators: reproduction, crossover and mutation. Reproduction is a process in which individuals are copied according to their objective function values. Those individuals with high performance are copied more often than those with low performance. After reproduction, crossover is performed as follows. First, pairs of individuals are selected at random. Then, a position k within the string of length l is selected at random. Two new individuals are created by swapping all characters of position $k+1$ to l. The mutation operator selects at random a position within the string and swaps its value.

The power of genetic algorithms comes from the fact, that individuals with good performance are selected for reproduction and crossing of high performance individuals speculates on generating new ideas from high performance elements of past trials.

For the initialisation of the genetic algorithm, the user needs to specify the boundaries of the input variable space, coding of the input variables, the size of the initial population and the probability of crossover and mutation. Goldberg [6] proposes to use a moderate population size, a high cross over probability and a low mutation probability. The coding of the input variables should use the smallest alphabet that allows to express the problem. In the experiment we use a population of size 16, we estimate 7 generations, the crossover probability is set to 0.6 and the mutation probability to 0.03.

V. Experimental evaluation

In this section we evaluate the proposed approach on the CAVIAR entry hall sequences[1]. The system is evaluated by recall and precision of the targets compared to the hand-labelled ground truth.

$$\text{recall} = \frac{\text{true positives}}{\text{total \# targets}} \tag{9}$$

$$\text{precision} = \frac{\text{true positives}}{(\text{true positives} + \text{false positives})} \tag{10}$$

We use the results of the manual adaptation as an upper benchmark. These results were obtained by a human expert who processed several times the sequences and hand tuned the parameters. Quickly the expert gained experience which kind of tracking errors depend on which parameter. The automatic regulation module does not use this kind of knowledge. For this reason, the recall and precision of the manual adaptation is the best we can hope to reach with an automatic method. We do not have manual adapted parameters for all sequences, due to the repetitive and time consuming manual task.

A lower benchmark is provided by the tracking results that do no adaptation. This means all 5 sequences are evaluated using the same parameter setting. Choosing parameters with high values[2] produce little recall and bad precision. Choosing parameters with low values[3] increase the recall but the very large number of false positives is not acceptable.

Table II shows the tracking results using a spatial and a spatio-temporal model and two parameter space exploration schemes. The first uses a brute force search (enumerative method) of the discrete parameter space composed of the discrete values for detection energy $\in [20, 30, 40]$, density $\in [5, 15, 25]$, sensitivity $\in [0, 20, 30, 40]$, split coefficient = 2.0, α = 0.001, area threshold = 1500. The method tests 36 parameter settings. The second exploration scheme uses a genetic algorithm as described in section IV-C.

The enumerative method has several disadvantages, that are reflected by the rather low performance measurements of the experiments. The sampling of the parameter space is coarse and therefore it happens frequently that none of the parameter settings provide an acceptable improvement. The same arguments are true for random sampling of the parameter space.

The spatial model using brute force method and the simple score has a small recall, but a better precision than the lower benchmark. The spatio-temporal measurements using the same parameter selection and evaluation measure produces superior results (higher recall and higher precision). This seems to be related to the spatio-temporal model. The precision can be further improved using the genetic approach and the more complex evaluation function (recall 39.7% and precision 78.8%).

[1]available at http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

[2]detection energy=30, density=15, sensitivity=30, split coefficient=1.0, α = 0.01, and area threshold=1500

[3]detection energy=10, density=15, sensitivity=20, split coefficient=2.0, α = 0.01, and area threshold=1500

VI. Conclusions and outlook

We presented an architecture for a tracking system that uses an auto-critical function to judge its own performance and an automatic parameter regulation module for parameter adaptation. This system opens the way to self-adaptive systems which can operate under difficult lighting conditions. We applied our approach to tracking systems, but the same approach can be used to increase the performance of other systems who depend of a set of parameters.

An auto-critical function and a parameter regulation module require a reliable performance evaluation measure. In our case, this measure is computed as a divergence of the observed measurements with respect to a scene reference model. We proposed an approach for the generation of such a scene reference model and developed a measure that is based on the measurement likelihood.

With this measure, we can compute a best parameter setting for pre-stored sequences. The experiments show that the auto-regulation greatly enhances the performance of the tracking output compared to a tracking without auto-regulation. The system can not quite reach the performance of a human expert, who uses knowledge based on the type of tracking errors for parameter tuning. This kind of knowledge is not available to our system.

The implementation of the auto-critical function can trigger the automatic parameter regulation. First successful tests have been made to host the system on a distributed system. The advantages of the distributed system architecture is that the tracking system can continue the real time tracking. There rests the problem of re-initialisation of the tracker. Currently, existing targets are destroyed when the tracker is reinitialised.

The current model relies entirely on ground truth labelling. The success of the method strongly depends on the quality of the model. In many cases, a small number of hand labelled trajectories can be gathered, but often their number is not sufficient for the creation of a valid model. For such cases we envision an incremental modeling approach by generating an initial model from few hand-labelled sequences. The initial model is then used to filter the tracking results, such that they are error free. These error free trajectories are then used to refine the model. This corresponds to a feed back loop in model generation. After a small number of iterations a valid model should be obtained. The option of such an incremental model is essential for non-static scenes.

Acknowledgment

This research is funded by the European commission's IST project CAVIAR (IST 2001 37540). Thanks to Thor List for providing the recognition evaluation tool.

Auto-regulation method	Recall	Precision	Total # targets	true positives	false positives
Manual adaptation (benchmark)	49.7	91.0	23180	11520	1136
Spatio-temporal model, (genetic approach,$G_{seq(10)}$)	39.7	78.8	21564	8556	2304
Spatio temporal model, (genetic approach, simple score G)	39.4	73.2	21564	8492	3108
Spatio temporal model, (brute force, simple score G)	38.1	72.2	21564	8224	3160
Spatial model, (brute force, simple score G)	29.2	68.7	21564	6302	2872
No adaptation (low thresholds)	68.0	24.5	21564	14672	45131
No adaptation (high thresholds)	28.3	47.5	21564	6109	6746

TABLE II

PRECISION AND RECALL OF THE DIFFERENT METHODS EVALUATED FOR 5 CAVIAR SEQUENCES (OVERLAP REQUIREMENT 50%).

REFERENCES

[1] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] A. Caporossi, D. Hall, P. Reignier, and J.L. Crowley. Robust visual tracking from dynamic control of processing. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 23–31, Prague, Czech Republic, May 2004.

[3] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, Prague, Czech Republic, May 2004.

[4] R.B. Fisher. The PETS04 surveillance ground-truth data sets. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, Prague, Czech Republic, May 2004.

[5] B. Géoris, F. Brémond, M. Thonnat, and B. Macq. Use of an evaluation and diagnosis method to improve tracking performances. In *International Conference on Visualization, Imaging and Image Proceeding*, September 2003.

[6] D.E. Goldberg. *Genetic Algorithms in Search and Optimization*. Addison-Wesley, 1989.

[7] T. Leung and J. Malik. Recognizing surfaces using three-dimensional textons. In *International Conference on Computer Vision*, Corfu, Greece, September 1999.

[8] C. Schmid. Constructing models for content-based image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 39–45, Kauai, USA, December 2001.