KERNEL SELECTION BY MUTUAL INFORMATION FOR NONPARAMETRIC OBJECT TRACKING

Jean-Marc Berthommé[1], Thierry Chateau[1] and Michel Dhome[1]
[1]*LASMEA - UMR 6602, Université Blaise Pascal, 24 Avenue des Landais, 63177 Aubière Cedex, France*
{*j-marc.berthomme, thierry.chateau, michel.dhome*}*@univ-bpclermont.fr*

Keywords: Kernel selection, information theory, nonparametric tracking.

Abstract: This paper presents a novel method of selecting kernels for the subsampling of nonparametric models used in real-time object tracking in a video stream. We propose a method based on mutual information, inspired by the CMIM algorithm (Fleuret, 2004) for the selection of binary features. This builds, incrementally, a model of appearance of the object to follow, consisting of representative and independant kernels taken from points of that object. Experiments show gains, in terms of accuracy, compared to other sampling strategies.

1 INTRODUCTION

Object tracking in a sequence of images is a research area particularly explored whose benefits in the applications are very numerous, going from video-surveillance to Human/Machine interfaces.

Many methods of object tracking by vision are based on the construction of a model from an initialization of the object's position on the first frame of the sequence. Then the only available model is the appearance of this object in this frame. So to follow it, this is to look in the next frame, and in its neighborhood, an area whose appearance is closer to that model.

This modeling is called nonparametric because it makes no assumption about the data distribution. It just links the observations without using an underlying representation. This avoids having to explain the structure of the model even if we should now discern the observations of the object. Nonparametric estimation is largely based on the use of kernel functions whose bandwidth is critical. To adapt to local densities, Sylvain Boltz (Boltz et al., 2009) used the k-nearest neighbors (k-NN) instead of the Parzen windows (Parzen, 1962) and showed how to estimate the Kullback-Leibler divergence within this framework. In object tracking he made the connection with Mean-Shift, whose k-NN generalize the case of a number of points fixed inside the point cloud. This algorithm, described in 1975 by Fukunaga and Hostetler (Fukunaga and Hostetler, 1975), was revived in 2000 by Comaniciu and Meer (Comaniciu et al., 2000).

But k-NN sin by their computational complexity. For two populations of n and m points in dimension d, we have to make $O(nmd)$ calculations to get the distances and $O(nmlogm)$ to sort them. Garcia et al. (Garcia et al., 2008) have shown that GPU parallelization pushed the limit and was better than approximate k-NN. This technical resolution of the problem is not challenged here. We only seek to reduce "upstream" the number of points of the model.

For this we propose an original algorithm based on information theory to select representative and independant kernels. Once the model is built, the tracking is performed by a sequential particle filter whose observation function uses an approximation of the Kullback-Leibler divergence by the k-nearest neighbors.

The article is divided into four parts. After this introduction, the second part describes the principle of model construction and tracking. The third presents experiments on real data, to evaluate the proposed method compared to other sampling strategies. Finally, the last concludes and offers some prospects of this work.

2 METHOD

This section describes the principle of the tracking algorithm in two steps: 1) construct the object model with the CMIM (*Conditional Mutual Information Maximization*) algorithm and 2) track the object in a sequence with a sequential MCMC (*Markov Chain Monte Carlo*) particle filter.

The initialization takes place in the first frame. The method is assumed to access to the *Region of Interest* (ROI) containing the cropped portion of the object to follow. The labelling of the pixels, from the object or its neighborhood, is binary (0/1) and is the *a priori* to start. The image labels are noted Y. Figure 1 shows an example of appearance to track, to the left, and the map of the associated labels, to the right.

Figure 1: Labelling of the object of interest. To the left: image of the object to track. To the right: labels image.

2.1 Kernel Generation and Optimization

The aim is to select certain points and kernels associated with the object to track. For this we draw M pixels $\mathbf{x_m}(u_m, v_m)$ at random from the ROI. Each point is coded as a vector of dimension 5: U,V,R,G,B where U,V are the coordinates of a system linked to the ROI and RGB the red, green, blue coding colors of that pixel. For each one we build nine kernels in the D dimensions as follows: R, G, B, UVR, UVG, UVB, UV, RGB and $UVRGB$. Each dimension is normalized by its maximum amplitude, from 0 to 255 for color (8 bits), 1 to w for the width and 1 to h for the height. Color C includes the channels R, G, B. Then, 9 normalized Euclidean distances are calculated for each pixel $\mathbf{x}(u,v)$ of the ROI from each extracted pixel $\mathbf{x_m}$:

$$d_C(\mathbf{x},\mathbf{x_m}) = \left(\frac{C(u,v)-C(u_m,v_m)}{255}\right)^2$$

$$d_{UV}(\mathbf{x},\mathbf{x_m}) = 1/2\left\{\left(\frac{u-u_m}{w-1}\right)^2+\left(\frac{v-v_m}{h-1}\right)^2\right\}$$

$$d_{UVC}(\mathbf{x},\mathbf{x_m}) = 1/2\left\{d_{UV}(\mathbf{x},\mathbf{x_m})+d_C(\mathbf{x},\mathbf{x_m})\right\}$$

$$d_{RGB}(\mathbf{x},\mathbf{x_m}) = 1/3\left\{d_R(\mathbf{x},\mathbf{x_m})+d_G(\mathbf{x},\mathbf{x_m})+\ldots d_B(\mathbf{x},\mathbf{x_m})\right\}$$

$$d_{UVRGB}(\mathbf{x},\mathbf{x_m}) = 1/2\left\{d_{UV}(\mathbf{x},\mathbf{x_m})+d_{RGB}(\mathbf{x},\mathbf{x_m})\right\}$$

For each distance, we define a probability to belong to the object or not by plating a Gaussian kernel $\mathtt{K}$. This step introduces a parameter λ, related to the kernel bandwidth, which spreads more or less the distribution around $\mathbf{x_m}$. At this stage, no normalization is performed to remain homogeneous with the Boolean labels. However, round to a normalization, the notions of kernel and distribution remain valid. Thus:

$$P[\mathbf{x}\in Object] = \mathtt{K}_{\mathtt{D},\lambda,\mathbf{x_m}}(\mathbf{x}) \text{ with :}$$

$$\mathtt{K}_{\mathtt{D},\lambda,\mathbf{x_m}}(\mathbf{x}) = \begin{cases} e^{-\lambda.d_D(\mathbf{x},\mathbf{x_m})} & \text{if } Y(\mathbf{x_m})=1 \\ 1-e^{-\lambda.d_D(\mathbf{x},\mathbf{x_m})} & \text{if } Y(\mathbf{x_m})=0 \end{cases}$$

Subsequently, there is X the map of probabilities of the pixels associated with the kernel $\mathtt{K}_{\mathtt{D},\lambda,\mathbf{x_m}}$. Figure 2 shows the influence of λ on two maps from the same kernel based on a pixel $\mathbf{x_m}$ in dimension UV.

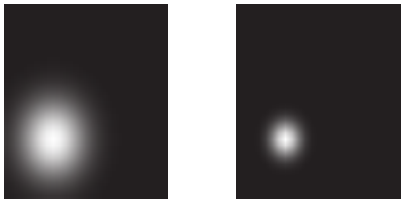


Figure 2: Map of probabilities X of the kernel $\mathtt{K}_{\mathtt{UV},\lambda,\mathbf{x_m}}$ for λ = 50 left and λ = 250 right. $\mathbf{x_m} = [8,22]$.

The parameter λ is very important in the coding of the model. We propose to optimize it by maximizing the amount of information exchanged between the probability map X and the labels Y. For this we evaluate their mutual information I (MacKay, 2003):

$$I(X;Y) = H(X)+H(Y)-H(X,Y)$$

The entropy $H(X)$ is based on $\overline{p_X}$, the average of the probabilities of X. So, in bits:

$$H(X) = -\overline{p_X}\ log_2\ \overline{p_X} - (1-\overline{p_X})\ log_2\ (1-\overline{p_X})$$

The entropy $H(Y)$ of the binary labels relies on the count of 0 and 1 in Y leading to the average probabilities p_0 and p_1 ($p_0+p_1=1$):

$$H(Y) = -p_0\ log_2\ p_0 - p_1\ log_2\ p_1$$

Joint entropy $H(X,Y)$ is equivalent to count the states 00, 01, 10 and 11. There are several ways to do but the fastest is to separate the probabilities of X according to their label to look at the probabilities of each set. With the convention $p_{01} = p_{(Y=0)\cap(X=1)}$ we have:

$$\begin{array}{lll} p_{00} = & (1-\overline{p_Y}) & (1-\overline{p_{X|Y=0}}) \\ p_{01} = & (1-\overline{p_Y}) & \overline{p_{X|Y=0}} \\ p_{10} = & \overline{p_Y} & (1-\overline{p_{X|Y=1}}) \\ p_{11} = & \overline{p_Y} & \overline{p_{X|Y=1}} \end{array}$$

$$H(X,Y) = -\sum_{i\in\{00,01,10,11\}} p_i\ log_2\ p_i$$

The optimization of I via, for example, the Levenberg-Marquardt method quickly converges to a solution λ^*. We do not detail it and simply present a possible evolution of I depending on λ in Figure 3. Note that I is bounded by the entropy of the labels

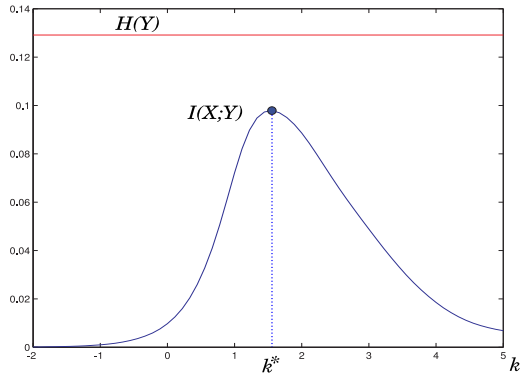


Figure 3: Kernel bandwidth optimization by maximizing the mutual information with the labels Y. $I = f(\lambda)$ with $\lambda = 10^k$ where k = [-2, 5].

$H(Y)$ in red.

Figure 4 shows the evolution of X from white to black depending on the kernel bandwidth λ. When λ becomes small the map of probabilities becomes all white ($\lim_{\lambda \to 0^+} e^{-\lambda . d} = 1$) and the information exchanged with the labels null. Similarly, when λ becomes large the probability map is all black ($\lim_{\lambda \to +\infty} e^{-\lambda . d} = 0$) and mutual information with the labels null again. Between the two there is a λ^* solution as $0 \leq I(\lambda^*) \leq H(Y)$.

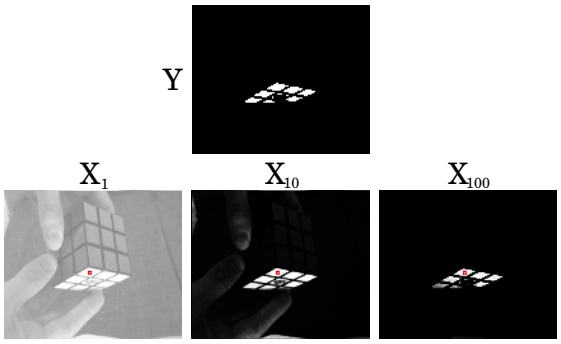


Figure 4: Illustration of the evolution of the map of probabilities X based on the kernel bandwidth. Y *vs* X for λ = {1, 10, 100}

2.2 Kernel Selection

At this stage, $N = 9M$ kernels, optimal in the sense of parameter λ, were generated. The selection phase is to choose K kernels with $K \ll N$. The goal is to have a set as small and as representative as possible of the labels Y of the ROI of the first image. For this we propose a method derived from the CMIM algorithm (*Conditional Mutual Information Maximization*) (Fleuret, 2004) with a novelty that is to compare booleans with probabilities and not with other booleans.

CMIM is an incremental algorithm based on the MinMax principle. At each step it looks for the element the most characteristic of labels knowing all what has already been explained. The first loop initializes a score $s[n]$ (where $n \in [1,N]$) for each kernel X_n that corresponds to the mutual information exchanged with the labels Y: $I(Y;X_n)$. The kernel which has the highest score wins and its index n is stored at position $\nu[1]$.

In the following loop, we calculate the conditional mutual informations between the kernels X_n and Y knowing $X_{\nu[1]}$ just selected: $I(Y;X_n|X_{\nu[1]})$. Scores $s[n]$ are then updated by retaining the minimum of $I(Y;X_n)$ and $I(Y;X_n|X_{\nu[1]})$, *i.e.* between $s[n]$ and $I(Y;X_n|X_{\nu[k]})$ for iteration k in the general case. This induces a selection of kernels as independent as possible. The new kernel with the maximum score is drawn to store its index in $\nu[k]$. It loops back. At the end all the indices of the K selected kernels are recovered in ν.

for n=*1...N* **do**
  s[n] $\leftarrow I(Y;X_n)$
end
for k=*1...K* **do**
  $\nu[k]$ = arg max$_n$ s[n]
  for n=*1...N* **do**
    s[n] $\leftarrow$ min{s[n], $I(Y;X_n|X_{\nu[k]})$}
  end
end

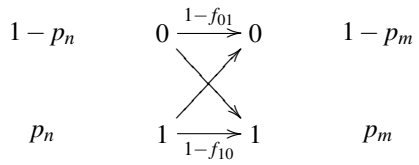
Algorithm 1: CMIM algorithm

We can accelerate this algorithm by stopping the calculation of the scores after a certain threshold due to their decline. For more details we refer to the original paper (Fleuret, 2004). What interests us most is to show how to calculate $I(Y;X_n|X_m)$ when Y is a binary image and X_n and X_m probabilities maps ($n,m \in [1,N]$). According to (MacKay, 2003) :

$$I(Y;X_n|X_m) = H(Y,X_m) - H(X_m) - H(Y,X_n,X_m) + H(X_n,X_m)$$

$H(Y,X_m)$ and $H(X_m)$ are calculated as above with $I(X;Y)$. The following two terms are more subtle. Let's start with $H(X_n,X_m)$ in the simple case where X_n and X_m have one element. It comes down to joint entropy of two Boolean random variables with known occurrences p_n and p_m (p_n means $P(X_n = 1)$) but hidden realizations. On these latter, an optimistic *a priori* is set. X_n and X_m are assumed to get as much information as possible together. For instance, if $p_n = 0.8$ and

$p_m = 0.7$ then $p_{(X_n=1)\cap(X_m=1)} = min(p_n, p_m) = 0.7$. This exchange of information can be represented by an oriented noisy channel ($H(X_n) \leq H(X_n)$) as follows:



The error rates f_{01} and f_{10} match to the percentages of 0 turned into 1 and 1 into 0. The optimistic approach assumes they are minimum and so it saturates the horizontal transitions (arrows) at the expense of the diagonals. A pessimistic approach would do exactly the opposite. The evaluation of the probabilities of the states 00, 01, 10 and 11 is then as follows:

$$\begin{array}{l} p_{11} = min(\,p_n,\,p_m) \\ p_{00} = min(1-p_n,\,1-p_m) \\ \left\{ \begin{array}{ll} p_{10} = 0 \text{ et } p_{01} = 1-p_{00}-p_{11} & \text{if } p_{11} = p_n \\ p_{01} = 0 \text{ et } p_{10} = 1-p_{00}-p_{11} & \text{if } p_{11} = p_m \end{array} \right. \end{array}$$

In the case, more common, where X_n and X_m have several elements, this operation is repeated for each pair of values. Then probabilities are averaged along the states 00, 01, 10 and 11. The total joint entropy is then:

$$H(X_n,X_m) = -\sum_{i\in\{00,01,10,11\}} \overline{p_i}\; log_2\; \overline{p_i}$$

The calculation of $H(Y,X_n,X_m)$ is close to that of $H(X_n,X_m)$ since Y has only two exclusive states: 0 and 1. Let's first consider a triplet of scalars (Y,X_n,X_m). We write $p_4 = [p_{00}, p_{01}, p_{10}, p_{11}]$, the probabilities associated with X_n and X_m, calculated exactly as above. The probabilities of the triplet are also gathered together into a single vector $p_8 = [p_{000}, p_{001}, p_{010}, p_{011}, p_{100}, p_{101}, p_{110}, p_{111}]$, following the convention $p_{101} = p_{(Y=1)\cap(X_n=0)\cap(X_m=1)}$. They comply with the following rule:

$$p_8 = \left\{ \begin{array}{ll} [p_4,0,0,0,0] & \text{if } Y = 0 \\ [0,0,0,0,p_4] & \text{if } Y = 1 \end{array} \right.$$

In the case that Y, X_n and X_m have several items this operation is repeated for each triplet. Then, the probabilities of the 8 states 000, 001, ... 111 are still averaged over each one to get the overall entropy:

$$H(Y,X_n,X_m) = -\sum_{i\in\{000,001,\ldots,111\}} \overline{p_i}\; log_2\; \overline{p_i}$$

These calculations maintain the following properties:

$$\begin{array}{l} H(X,X) = H(X) \\ H(Y,X,X) = H(Y,X) \\ I(Y;X|X) = 0 \end{array}$$

Figure 5 shows an example of the selection of five kernels by CMIM based on Figure 1:

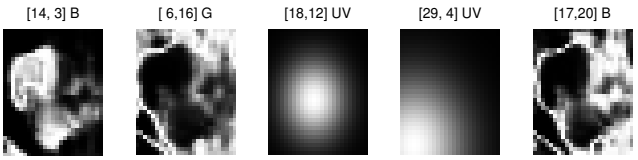


Figure 5: Selection of 5 kernels by CMIM.

2.3 k-NN MCMC Tracking

Tracking a ROI, reflection of the object in the image, couples equal radiometric (color RGB) and geometric (position UV) information. The goal is to find in each new image the closest ROI to the original. Information is normalized by dimension and weighted on UVRGB by the respective weights $[1/4, 1/4, 1/6, 1/6, 1/6]$. After each image can be viewed as a probability density function (PDF) and compared with another one by a similarity measure, namely the Kullback-Leibler divergence. Boltz et al. (Boltz et al., 2009) (Boltz, 2008) showed how to estimate it in a k-NN framework and why it is well aware of the local densities in high dimensions.

For a reference population R of n_R points in dimension d and a target population T of n_T points in the same dimension ($n_R \neq n_T$ a priori), with $\rho_k(U,s)$, the Euclidean distance between the point s and its k^{th} nearest neighbor in U ($U \equiv R$ or T), the Kullback-Leibler divergence $\mathfrak{D}_{KL}(T,R)$ can be estimated, in an unbiased way, by:

$$\mathfrak{D}_{\mathrm{KL}}(T,R) \overset{\text{k-NN}}{=} \log\frac{n_R}{n_T-1} + \frac{d}{n_T}\sum_{s\in T}\log\frac{\rho_k(R,s)}{\rho_k(T,s)}$$

An ideal estimation would consider all the pixels in R and T. However, to avoid the combinatorial explosion or just speed up the calculations, we try to downsample these regions. The question is how. Subsequently we will compare three types of subsamples: two regular, one random and one from the kernels selected by CMIM.

Trackings use a sequential particle filter (Arulampalam et al., 2002) with a Markov chain (Khan et al., 2005). Each particle represents a region of the image. The upper left corner of the reference ROI R indicates the first position. From the second we apply N_p random transformations φ_i to R. It generates N_p particles or ROI targets T_i whose weight is: $w_i = e^{-\mu.\mathfrak{D}_{\mathrm{KL}}(T_i,R)}$ with μ a fixed constant. The particle with the maximum weight takes on the new position. The others are not forgotten. They will generate new particles in the subsequent iterations. The repoduction chance is governed by the Metropolis-Hastings rule: $min(1, w_{new}/w^*) < rand()$. Few particles of low weight are also "burned" at each iteration.

Finally we will make big assumptions: that the reference R does not change, that the object is far from the camera and therefore that the ROI does undergo only translations and remains fixed size.

3 EXPERIMENTS

Evaluations show how the selection of kernels is better than a regular or random subsampling in terms of filtering for an equivalent tracking performance.

3.1 Evaluation

All experiments are based on a video of the CAVIAR database (see acknowledgements). A manual tracking of targets gave the ground truth GT. On this same sequence the particle filter has been launched in different configurations. For each track of the algorithm, its truth AT was recorded. Thus AT and GT contain series of ROI indexed by frame number.

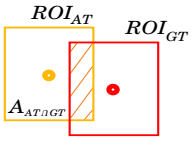


Figure 6: Intersection area $\mathcal{A}_{AT\cap GT}$ of the ROI of the algorithm AT and of the ground truth GT in the same image.

To determine the quality η of a tracking we compare the ROIs of AT and GT by pair in each image i. We calculate their reports of intersection area and union and check whether for a given tolerance τ it makes "fit" the tracked ROIs to the ground or not. This state, good or not, is rated β. Then for N_i images:

$$\beta_i(\tau) = \begin{cases} 1 & \text{if } \frac{\mathcal{A}^i_{AT\cap GT}}{\mathcal{A}^i_{AT\cup GT}} \geq \tau \\ 0 & \text{else} \end{cases}$$

$$\eta(\tau) = \frac{1}{N_i}\sum_{i=1}^{N_i}\beta_i(\tau)$$

Note that since the particle filter involves random numbers, each curve indicating a configuration is based on ten trials. The average case was drawn in solid lines and the worst and best cases in dashed lines. Then the configurations are compared according to the quality of their tracking and the number of points extracted by their subsampling, that refers to a percentage of extraction because of the size of the ROI.

Four significant subsamplings are presented here among all those tested: regular on raw data ("Raw"), regular on data smoothed by a Gaussian kernel ("Gauss"), random on data averaged per Voronoi cell ("RandVor"), and finally from the convolution of kernels selected by CMIM with optimized bandwidth ("KerOpt").

A regular subsampling involves taking a pixel of 2, 3, etc. following U and V. We note that a pixel of 1 leads to exhaustive sampling without loss. The size of an average or Gaussian kernel is directly related to the sampling period. So for a pixel of p it is simply smoothed by a kernel of size [p×p]. In the case of random subsampling "RandVor", the Voronoi cells are defined by all the pixels which are closest to a site as defined in the standard l_2-norm. These sites match to pixels randomly selected in the ROI.

Note that each subsampling is frozen all along a follow-up. Even if it is random, it is instantiated in the first ROI and stored for the whole sequence. It applies both to the reference ROI as to the ROI target. The similarity measure employed is the Kullback-Leibler divergence. It allows to compare sets of different sizes which is useful when the ROI reaches the edge of the image.

3.2 Results

Results are based on the sequence "Walk-ByShop1cor" of CAVIAR between images 192 and 309. We tried to follow the head of a man on a window of size 31×25 pixels. The curves show the variation of good tracking η depending on the tolerance τ on the percentage of area common with the ground truth. The four curves in green, khaki, brown and red in Figure 8 represent subsamplings of a pixel of 1, 4, 7 and 10 *i.e.*: 775, 56, 20 and 12 points for the configurations "Raw", "Gauss" and "RandVor", from top to bottom. Down on the same figure, the three curves gray, violet and cherry compile the "KerOpt" configuration for respectively: 55, 30 and 5 points filtered from the selected kernels.

3.3 Analysis

It is easy to see that the more there are points in the model and the more the tracking is good as indicated by the shades of green curves "above" the red ones. However getting more points drives to more expensive calculations as mentioned in the introduction. We also see that all configurations are struggling to cross the milestone of 90% of good tracking (horizontal asymptote for "Raw"). Indeed, at the end of the sequence, the filter astrayed when the man came out of the stage, leaving little relevant points in the final good ROI (see Figure 7).

The case "Gauss" also reveals an interesting point. One might expect that the exhaustive sampling in green (N = 775 with 1 pixel of 1 and thus a useless convolution of $[1\times1]$) would have lead to the best tracking as lossless, but in reality the associated convergence basins are somewhat rough. So any spatial smoothing softens them. This explains why the khaki curve (N = 56 with a pixel of 4 and a convolution of $[4\times4]$) passes over the green.

The case "RandVor" is a little benchmark. Presumably, this random case is equivalent or better than the 2 regular cases, but it is only true on average. Given the standard deviations it is the one that leads to the most unpredictable trackings, which makes sense after all.

The case "KerOpt" is equivalent to the others in terms of pure tracking. However, as the the number of sample points (K = 5, 30 and 55) resulting of the number of selected kernels by filtering (each probability map X has provided a standard filter on $UVRGB$) is reduced, performance can be judged superior. In general, the loss of information due to filtering degrades the sharpness of comparison between ROIs, although this lightens the appearance model. By choosing some observations, representative of the remote object and its environment, CMIM limits the damage. The ROIs whose informations *a priori* from the object ($Y = 1$) appear close to the reference emerge with the k-NN. k-NN provide flexibility to the comparisons while kernels manage the diffusion of probabilistic assumptions.

4 CONCLUSION

Our goal was to track a target with a low cost appearance model, *i.e.* based on few points. With access to a clipping of the object of interest in the first frame and based on notions of information theory, we have shown how to build a light model made up of independent and representative kernels of the prime appearance. Trackings made with this filtering were compared to other more traditional and showed equivalent performance for a number of points lower.

However, despite this result, we are still far from satisfied. Many things escape us and has to be improved. Here's a partial list: 1) integrate the temporal coherence in addition to the spatial coherence by introducing explicit time T in a new representation, for instance: $UVRGBT$, 2) make evolve the reference ROI R along the tracking by an on-line learning of new likely labels, 3) add new parameters to the transformation φ to consider rotations or homotheties or even why not see φ in a nonparametric way by considering each point of the model as a single control point connected to other by a consistent deformable mesh, 4) penalize the estimate of $\mathfrak{D}_{\mathrm{KL}}$ according to the degree of internal consistency of the transformations φ of the different pixels ranging from T to R via their k-NN, 5) understand the mechanisms of association of points and dimensions in order to ease the weighting policies (soft *vs* hard clustering). All these points seem difficult but not unattainable.

ACKNOWLEDGEMENTS

Thanks to the EC Funded CAVIAR project/IST 2001 37540 for having provided the data. This work is supported by the French Ministry of Higher Education and Research (MESR fellowship).

Figure 7: "WalkByShop1cor" sequence. Example of a ROI (yellow rectangle) tracked in images 200, 250 and 275. Tracking loss in image 295.

REFERENCES

Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188.

Boltz, S. (2008). *A statistical framework in variational methods of image and video processing problems with high dimensions*. PhD thesis, University of Nice - Sophia Antipolis, France. supervised by E. Debreuve and M. Barlaud.

Boltz, S., Debreuve, E., and Barlaud, M. (2009). High-dimensional statistical measure for region-of-interest tracking. *IEEE Transactions on Image Processing*, 18(6):1266–1283.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 2, pages 142–149. Published by the IEEE Computer Society.

Fleuret, F. (2004). Fast binary feature selection with conditional mutual information. *The Journal of Machine Learning Research*, 5:1531–1555.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

Garcia, V., Debreuve, E., and Barlaud, M. (2008). Fast k nearest neighbor search using GPU. In *CVPR Workshop on Computer Vision on GPU (CVGPU)*, Anchorage, Alaska, USA.

Khan, Z., Balch, T., and Dellaert, F. (2005). Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1805–1918.

MacKay, D. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.

Parzen, E. (1962). On the estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076.

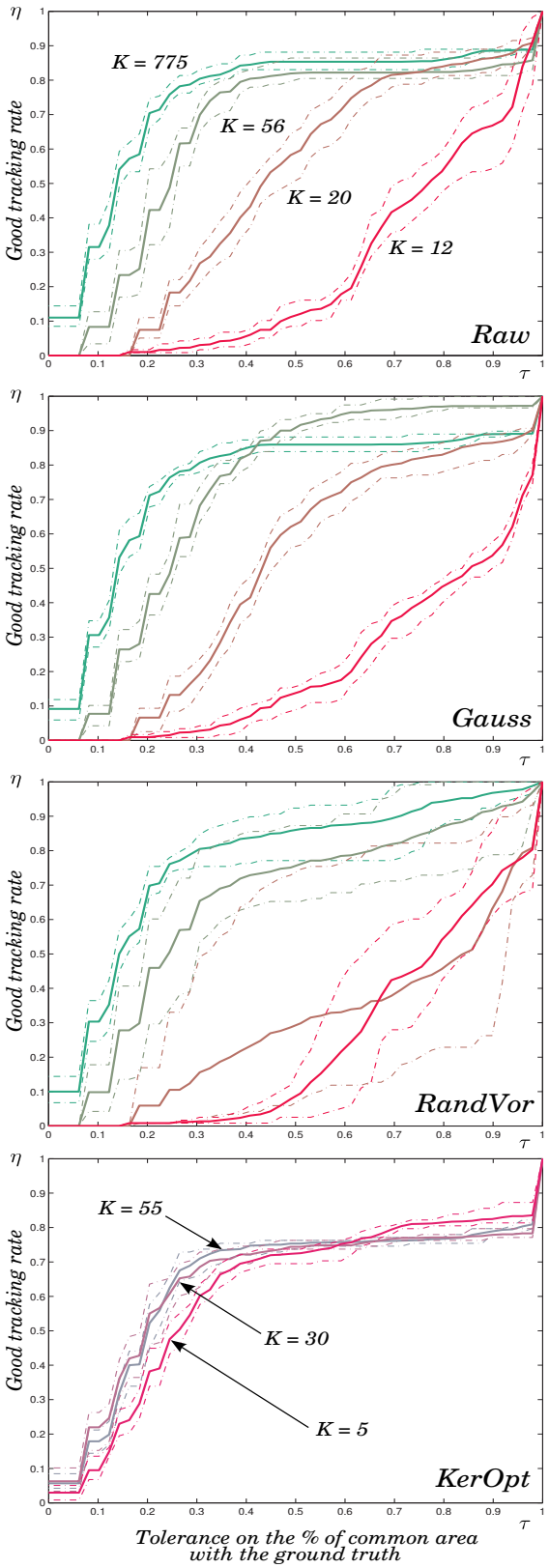


Figure 8: Tracking performances $\eta(\tau)$ of the subsamplings "Raw", "Gauss" and "RandVor" for K = {775, 56, 20, 12} points and "KerOpt" for K = {55, 30, 5} points.