Electronic Letters on Computer Vision and Image Analysis 8(1):27-43, 2009

Realtime Kernel based Tracking

T. Chateau* and J.T. Lapresté*

* *Lasmea, CNRS/Blaise-Pascal University, Clermont-Ferrand, France*

Received 26 May 2008; accepted 13 January 2009

Abstract

We present a solution for realtime tracking of a planar pattern. Tracking is seen as the estimation of a parametric function between observations and motion and we propose an extension of the learning based approach presented simultaneously by Cootes and al. and by Jurie and Dhome. We show that the hyperplane classic algorithm is a specific case of a more generic linearly-weighted sum of fixed non-linear basis functions model. The weights associated to the basis functions (kernel functions) of the model are estimated from a training set of perturbations and associated observations generared in a synthetic way. The resulting tracker is then composed by several iterations on trackers learned with coarse to fine magnitude of perturbations. We compare the performance of the method with the linear algorithm in terms of accuracy and convergence frequency. Moreover, we illustrate the behaviour of the method for several real toy video sequences including different patterns, motions and illumination conditions, and for several real video sequences sampling from rear car tracking databases.

Key Words: realtime planar tracking, kernel based regression functions, rear-car tracking

1 Introduction

Tracking a planar textured pattern is a popular topic in computer vision [2]. The aim is to estimate the unknown motion (generally expressed by a homography) of the pattern observed into a video sequence. Solutions to this problem are classified according to the video information available at the estimation time. First approaches, called *offline* use all the video information to solve the motion estimation problem. In this case, global optimization techniques can be applied. In [6], the author presents an optimization solution to pedestrian trajectories estimation from a video sequence. Moreover, in [16], a multiple target tracking system is proposed, into a global Monte-Carlo framework. Second approaches are called *online* and only past and present video information are available. Real-time tracking are *online* methods where the result of the motion estimation must be produced before the next video frame. The method presented in this paper is a real-time method.

Generally, real-time tracking can be modelized with a regression function between observations (the image) and a motion model of the template to be tracked. Some methods (called model based approaches) rely on an analytical model of the regression function. In [8], the authors compute a first order approximation of the relation between the observations (luminance) and the motion model and propose to use the Jacobian matrix

Correspondence to: <Thierry.CHATEAU@univ-bpclermont.fr>

Recommended for acceptance by <Thomas Breuel>
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

to define the regression function. Recently, Benhimane et al. [3] extend the method to a second order approximation of the Hessian matrix. Other methods (called learning based approaches), use a parametric form for the regression function where the parameters are estimated from a training set of motions and associated observations. Jurie and Dhome and [10] and Cootes and al. [4] have simultaneously proposed a first order hyperplane model for the regression function where the parameters of the linear matrix between motion and observations are estimated using a least-square error criteria computed from a training set. Generally, image features used are normalized luminance of pixels. However, In [11], Chateau et al. use Haar-like wavelets to describe the image. We propose to extend the learning based approach presented simultaneously by Jurie and Dhome [10] and Cootes and al. [4] to a parametric regression model using non linear basis functions. Kernel based regression functions have been used recently for tracking objects but using simple motion models (translation/scale). The pioneering work is the one of Avidan [1] (*support vector tracking*) which uses the output of an SVM based regression function to perform a tracking task. The idea is to link the SVM scores with the motion of the pattern between two images shots. This method provides a way to track classes of objects. No model of the current object is learnt but the classifier uses a generic model learnt offline. Williams [15] proposes a probabilistic interpretation of SVM. He presents a solution based on RVM (relevance vector machine) ([13]), combined with a Kalman temporal filtering. RVM is used to link the image luminance measure to the relative motion of the object with a regression relation. Recently, Thayananthan and al. [12] have presented a learning based approach to track articulated human body motion extending the RVM kernel based machine to multivariate MVRVM.

This paper is organized as follows. Section two deals with the general framework of visual tracking. Section three presents the regression function proposed: a kernel based parameter model where parameters are estimated from a training set. The resulting solution is compared with the classical one (hyperplane approximation) in section four, in terms of accuracy and convergence frequency related to several relevant parameter such as the magnitude of perturbations chosen for the learning step or noisy obsevations. Moreover, we illustrate the behaviour of the method for several real toy video sequences including different patterns, motions and illumination conditions, and for several real video sequences sampling from rear car tracking databases.

2 Visual Tracking

We present a generic solution for pattern tracking based on the definition on a regression function between a motion model and the variation of the template appearance.

Let us define $\mathtt{I}_k$ be an image extracted from a video sequence at time k, and $\mathcal{W}$ a planar pattern to be tracked, defined by four corner points into the image. Let us define a state associated to the image position of the pattern by:

$$\mathbf{p}_k \doteq (\mathbf{p}_k^1, \mathbf{p}_k^2, \mathbf{p}_k^3, \mathbf{p}_k^4),$$

a vector composed by the four corners of the pattern, where $\mathbf{p}_k^i = (u_k^i, v_k^i)^t$ denotes the position of $\mathbf{p}_k^i$ expressed into the image based reference frame.

Temporal matching of $\mathcal{W}$ can be seen as the estimation $\hat{\mathbf{p}}_k$ of the state system, for each new image of the video sequence. It can be realized in a iterative way, from the motion estimation $\boldsymbol{\delta}_{\mathbf{p}k}$ of the four template corners between two successive images:

$$\hat{\mathbf{p}}_k = \mathbf{p}_{k-1} + \boldsymbol{\delta}_{\mathbf{p}k} \tag{1}$$

Let $\mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}_k)$, be an observation function which provides a feature vector associated to $\mathcal{W}$ for the position defined by the state $\mathbf{p}_k$ in the image at time k (for example the luminance computed on a sub-sampling grid of $\mathcal{W}$). A direct consequence of the so-called image constancy assumption can be expressed as follows:

$$\forall i, j \in \{1, .., K\},\ \mathbf{z}(\mathtt{I}_i, \mathcal{W}, \mathbf{p}_i) = \mathbf{z}(\mathtt{I}_j, \mathcal{W}, \mathbf{p}_j) = \mathbf{z}^*_{\mathcal{W}} \tag{2}$$

with K is the number of images of the video-sequence. $\mathbf{z}^*_{\mathcal{W}}$ is a feature vector extracted to the pattern in first image.

Now, the variation of the observations between two successive images, given the previous state, is defined by:

$$\boldsymbol{\delta}_{\mathbf{z}k} \doteq \mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}_{k-1}) - \mathbf{z}(\mathtt{I}_{k-1}, \mathcal{W}, \mathbf{p}_{k-1}) \tag{3}$$

Using (2), the previous equation becomes:

$$\boldsymbol{\delta}_{\mathbf{z}k} = \mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}_{k-1}) - \mathbf{z}^*_{\mathcal{W}} \tag{4}$$

We propose to link motion $\boldsymbol{\delta}_{\mathbf{p}k}$ and observation variation $\boldsymbol{\delta}_{\mathbf{z}k}$ by a regression function:

$$\boldsymbol{\delta}_{\mathbf{p}k} = \mathbf{f}(\boldsymbol{\delta}_{\mathbf{z}k}; \mathbf{w}_k) + \boldsymbol{\epsilon}_k \tag{5}$$

where $\boldsymbol{\epsilon}_k$ denotes a random noise and $\mathbf{w}_k$ is the vector of parameters of the regression function. Since this formulation depends on time (k), parameters must be estimated at each iteration. The idea is to find a new relation, in which the parameters of the regression function have to be estimated only for the first image of the sequence.

Let $\mathtt{H}_k$ be the homography between the position of the four corners of the pattern to be tracked and a canonical reference frame. $\mathbf{p}_k$ is projected into $\mathbf{P}_k = \mathbf{P}_0 = \left((0,0)^t, (0,1)^t, (1,1)^t, (1,0)^t\right)$ with :

$$\tilde{\mathbf{P}}_0 = \tilde{\mathbf{P}}_k \propto \mathtt{H}_k.\tilde{\mathbf{p}}_k \tag{6}$$

Notation $\tilde{\mathbf{p}}_k$ is introduced to define $\mathbf{p}_k$ with homogeneous coordinates. $\mathtt{H}_k$ is computed simply from matching $\mathbf{P}_0$ with $\mathbf{p}_k$ and solving the resulting linear system [9]. The variation of the state vector can be expressed in the canonical reference frame by:

$$\tilde{\boldsymbol{\delta}_{\mathbf{P}}}_k = \mathtt{H}_{k-1}.\tilde{\boldsymbol{\delta}_{\mathbf{p}}}_k \tag{7}$$

$\boldsymbol{\delta}_{\mathbf{P}k}$ is the variation of the projection of $\mathbf{p}_{k-1}$ into the canonical reference frame using the previous homography $\mathtt{H}_{k-1}$. We propose to link $\boldsymbol{\delta}_{\mathbf{P}k}$ with the variation of the observation by the following regression function:

$$\boldsymbol{\delta}_{\mathbf{P}k} = \mathbf{F}(\boldsymbol{\delta}_{\mathbf{z}k}; \mathbf{w}) + \boldsymbol{\varepsilon}_k \tag{8}$$

where $\boldsymbol{\varepsilon}_k$ is a random noise. In this expression, the parameter vector $\mathbf{w}$ to be estimated does not depend on time. So, it can be estimated once for all frames of the sequence.

The resulting template tracking method is summarized into the algorithm 1. and, Figure 1 illustrates the principle of the method.

3 Kernel based Regression Functions

3.1 Learning

A key point associated with the method proposed in the previous section concerns the model of the regression function $\mathbf{F}$ and the estimation of the associated vector of parameters $\mathbf{w}$. $\mathbf{F}$ can be either linear [8] [10], or non-linear [3].

The vector of parameters $\mathbf{w}$ can be estimated either in an analytical way (estimation of the Jacobian matrix [8] or a second order matrix in [3]), or using machine learning techniques [10].

Let $\mathcal{V} \doteq \{\boldsymbol{\delta}_{\mathbf{P}}^{(n)}, \boldsymbol{\delta}_{\mathbf{z}}{}^{(n)}\}$ be a learning set built from random motions $\{\boldsymbol{\delta}_{\mathbf{P}}^{(n)}\}_{n=1}^{N}$ of the pattern (projected into a reference frame), associated to the variations of the feature vector (observation) $\{\boldsymbol{\delta}_{\mathbf{z}}{}^{(n)}\}_{n=1}^{N}$. The learning motions are applied to the reference image (generally the first image of the video sequence).

Jurie and Dhome propose to to use a hyperplane model for the regression function:

$$\mathbf{F}(\boldsymbol{\delta}_{\mathbf{z}}; \mathtt{W}) = \mathtt{W}\boldsymbol{\delta}_{\mathbf{z}} \tag{9}$$

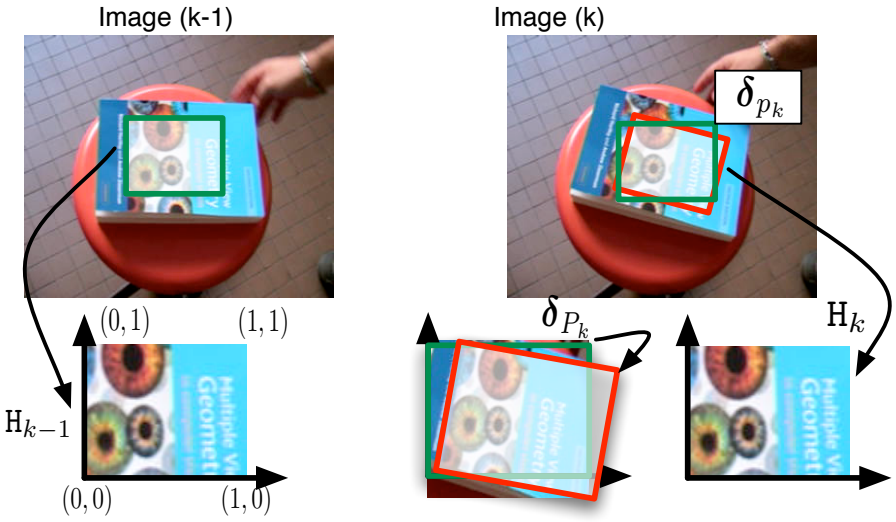

Figure 1: Illustration of the tracking method. The reference pattern, at time k is projected into a canonical reference frame using the homography $\mathtt{H}_{k-1}$. A regression function estimates the motion ($\boldsymbol{\delta}_{\mathbf{p}_k}$) between $k-1$ and k according to the observed variation. The motion of the pattern is then simply given by $\tilde{\boldsymbol{\delta}}_{\mathbf{P}k} = \mathtt{H}_{k-1}.\tilde{\boldsymbol{\delta}}_{\mathbf{p}_k}$ and the homography is updated.

The parameter matrix $\mathtt{W}$ is learnt from the training set $\mathcal{V}$ by minimizing a "least-square" error measure.

We propose to use models which are a linearly-weighted sum of M fixed non-linear basis functions:

$$\mathbf{F}(\boldsymbol{\delta}_\mathbf{z};\mathtt{W}) = \mathtt{W}.\boldsymbol{\phi}(\boldsymbol{\delta}_\mathbf{z}) = \sum_{m=1}^{M} \mathbf{w}_m \boldsymbol{\phi}_m(\boldsymbol{\delta}_\mathbf{z}) \tag{10}$$

$\boldsymbol{\phi}(\boldsymbol{\delta}_\mathbf{z})$ denotes a vector of M basis functions:

$$\boldsymbol{\phi}(\boldsymbol{\delta}_\mathbf{z}) = [\phi_1(\boldsymbol{\delta}_\mathbf{z}), \phi_2(\boldsymbol{\delta}_\mathbf{z}), ..., \phi_M(\boldsymbol{\delta}_\mathbf{z})]^T \tag{11}$$

and with $\phi_m(\boldsymbol{\delta}_\mathbf{z}) = k(\boldsymbol{\delta}_\mathbf{z}, \boldsymbol{\delta}_\mathbf{z}^{*\mathbf{m}})$, a kernel function, applied to $\boldsymbol{\delta}_\mathbf{z}$ and a basis vector denoted $\boldsymbol{\delta}_\mathbf{z}^{*\mathbf{m}}$.

Let $\mathtt{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M)$ be a matrix of parameters associated to the M basis functions. The objective is to find values for $\mathtt{W}$ such that $\mathbf{F}(\boldsymbol{\delta}_\mathbf{z};\mathtt{W})$ makes good predictions for new data: i.e. it models the underlying generative function.

A classic approach to estimating $\mathtt{W}$ is "least- square", minimization of the error measure:

$$E_\mathcal{D}(\mathtt{W}) = \frac{1}{2}\sum_{n=1}^{N} \left|\left| \delta_\mathbf{P}^{(n)} - \mathbf{W}.\boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(n)}) \right|\right|^2 \tag{12}$$

This can be rewritten in the following system:

$$\left(\boldsymbol{\delta}_\mathbf{P}^{(1)}, \boldsymbol{\delta}_\mathbf{P}^{(2)}, ..., \boldsymbol{\delta}_\mathbf{P}^{(N)}\right) = \mathtt{W}\left(\boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(1)}), \boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(2)}), ..., \boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(N)})\right) \tag{13}$$

Let denote $\Phi \doteq \left(\boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(1)}), \boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(2)}), ..., \boldsymbol{\phi}({\boldsymbol{\delta}_\mathbf{z}}^{(N)})\right)$ and $\Delta_P \doteq \left(\boldsymbol{\delta}_\mathbf{P}^{(1)}, \boldsymbol{\delta}_\mathbf{P}^{(2)}, ..., \boldsymbol{\delta}_\mathbf{P}^{(N)}\right)$; the system can be expressed under a more compact form:

$$\Delta_P = \mathtt{W}.\Phi \tag{14}$$

Algorithm 1 Tracking

Input : state $\mathbf{p}_0$, image $\mathtt{I}_0$ and regression function $\mathbf{F}$

Output : set of states $\{\mathbf{p}_k\}_{k=1}^K$

Initialisation : $k = 0$, extraction of the reference feature vector $\mathbf{z}^*_{\mathcal{W}} \doteq \mathbf{z}(\mathtt{I}_0, \mathcal{W}, \mathbf{p}_0)$ and estimation of the canonical homography $\mathtt{H}_0$

for $k = 1$ to K (loop on the images of the video sequence) **do**

Observation : Extraction of the feature vector $\boldsymbol{\delta}_{\mathbf{z}k} = \mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}_{k-1}) - \mathbf{z}^*_{\mathcal{W}}$

Estimation : Estimation of the motion into the canonical reference frame, then into the image reference frame :

$$\boldsymbol{\delta}_{\mathbf{P}k} = \mathbf{F}(\boldsymbol{\delta}_{\mathbf{z}k}; \mathbf{w})$$

$$\tilde{\boldsymbol{\delta}}_{p_k} \propto \mathtt{H}_{k-1}^{-1}.\tilde{\boldsymbol{\delta}}_{P_k}$$

Update : state vector and homography

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \boldsymbol{\delta}_{\mathbf{p}k}$$

$\mathtt{H}_k$, Homography between $\mathbf{P}_0$ and $\mathbf{p}_k$

end for

and the estimation of the parameter matrix $\mathtt{W}$ using (12) is given by:*

$$\mathtt{W}_{LS} = \Delta_P \Phi^+, \tag{15}$$

Alternative methods based on the "least-square" criterion can be used to estimate the parameter matrix $\mathtt{W}$. A solution is to place a prior over $\mathtt{W}$ in order to set many weights to zero. The resulting model is then called sparse linear model. SVM (*Support Vector Machine*) [14] is a sparse linear model where the weights are estimated by the minimization of a Lagrange multipliers based functional. Other sparse linear models, like RVM (*Relevance Vector Machines*) [13] or multivariate RVM [12] may also be employed.

Generally, vectors used in basis functions ($\boldsymbol{\delta}_{\mathbf{z}}^{*\mathbf{m}}$) can be chosen from the training set. It is also possible to use the entire training set and in this case $N = M$. Moreover, we can notice that the hyperplane model presented in eq. (9) is a special case of the model (10) with linear basis functions $\phi_m(\boldsymbol{\delta}_{\mathbf{z}})$.

We make the common choice to use Gaussian data-centred basis functions:

$$\phi_m(\boldsymbol{\delta}_{\mathbf{z}}) = \exp\left[-\left(\boldsymbol{\delta}_{\mathbf{z}} - \boldsymbol{\delta}_{\mathbf{z}}^{*\mathbf{m}}\right)^2 / \sigma^2\right], \tag{16}$$

which gives us a "radial basis function" (RBF) type model from which the parameter σ must be adjusted. On one hand, if σ is too small, the "design matrix" Φ is mostly composed of zeros. On the other hand, if σ is too large, Φ is mostly composed of ones ($\exp(0)$). We propose to adjust σ using a non-linear optimization maximizing a cost function based on the sum of the variances computed for each line of Φ:

$$\sigma = \arg\max_{\sigma}[C(\sigma)] \tag{17}$$

with

$$C(\sigma) = \sum_{n=1}^{N}\sum_{m=1}^{M}\left(\phi_m(\boldsymbol{\delta}_{\mathbf{z}}^{(n)}) - \overline{\phi}(\boldsymbol{\delta}_{\mathbf{z}}^{(n)})\right)^2 \tag{18}$$

and

$$\overline{\phi}(\boldsymbol{\delta}_{\mathbf{z}}^{(n)}) = \frac{1}{M}\sum_{m=1}^{M}\phi_m(\boldsymbol{\delta}_{\mathbf{z}}^{(n)}) \tag{19}$$

* Φ^+ denotes the pseudo-inverse of Φ.

An overview of the learning method proposed in this section, called KBT (*Kernel based Machine Learning Tracker*) is given in algorithm 2. The learning method is called L times, for different values of b, following a coarse to fine scheme [5, 7]

Algorithm 2 KBT: learning step

Input : Size of the training set N, training parameter b, number of basis functions M, initial homography $\mathtt{H}_0$, initial std. of the kernel function σ_0.
Output : parameter matrix $\mathtt{W}$ and std. of the kernel function σ.
Pattern motion generation : A set of N motion vectors is drawn according to a uniform law on the interval $[-b, b]$: $\{\boldsymbol{\delta}_{\mathbf{P}}^{(n)}\}_{n=1}^{N}, \mathbf{P}^{(n)} \sim \mathcal{U}(-b, b)$.
Observation : Compute $\{\boldsymbol{\delta}_{\mathbf{z}}{}^{(n)}\}_{n=1}^{N}$ such as :

$$\boldsymbol{\delta}_{\mathbf{z}}{}^{(n)} = \mathbf{z}(\mathtt{I}_0, \mathcal{W}, \mathbf{p}^{(n)}) - \mathbf{z}^*_{\mathcal{W}},$$

with

$$\tilde{\mathbf{p}}^{(n)} \propto \mathtt{H}_0^{-1}\tilde{\mathbf{P}}^{(n)}.$$

Draw basis functions: draw $\{\boldsymbol{\delta}_{\mathbf{z}}^{*\mathbf{m}}\}_{m=1}^{M}$, using an uniform law from $\{\boldsymbol{\delta}_{\mathbf{z}}{}^{(n)}\}_{n=1}^{N}$.
estimation of the kernel function parameter σ : non-linear optimization of σ, such as:

$$\sigma = \arg\max_{\sigma} \left\{ \sum_{n=1}^{N} \sum_{m=1}^{M} \left(\phi_m(\boldsymbol{\delta}_{\mathbf{z}}^{(n)}) - \overline{\phi}(\boldsymbol{\delta}_{\mathbf{z}}^{(n)}) \right)^2 \right\}$$

estimation of the weight (parameter) matrix $\mathtt{W}$:

$$\mathtt{W} = \Delta_P.(\Phi^T(\Phi\Phi^T)^{-1})$$

with $\Phi \doteq \left(\boldsymbol{\phi}(\boldsymbol{\delta}_{\mathbf{z}}{}^{(1)}), \boldsymbol{\phi}(\boldsymbol{\delta}_{\mathbf{z}}{}^{(2)}), ..., \boldsymbol{\phi}(\boldsymbol{\delta}_{\mathbf{z}}{}^{(N)})\right)$

3.2 Tracking

The learning step provides, for each training level l:

1. The weight matrix $\mathtt{W}_l$
2. The set of basis functions $\{\boldsymbol{\delta}_{z,l}^{*m}\}_{m=1}^{M}$
3. The width parameter associated with the kernel function: σ_l.

These parameters are then used to build the regression function $\mathbf{F}(\boldsymbol{\delta}_{\mathbf{z}}; \mathtt{W}_l) = \sum_{m=1}^{M} \mathbf{w}_{m,l}\boldsymbol{\phi}_{m,l}(\boldsymbol{\delta}_{\mathbf{z}})$. Moreover, it is possible to call the regression function several times in order to increase the tracking precision. The resulting method is presented in algorithm 3.

4 Experiments

This section presents the experiments achieved in order to compare the proposed method to the reference linear algorithm. After a description of the datasets and the methodology, experimental results are presented and then discussed.

Algorithm 3 KBT: tracking step

Input: Regression function parameters: $\mathtt{W}_l, \{\boldsymbol{\delta}_{z,l}^{*m}\}_{m=1}^{M}$, initial state $\mathbf{p}_0$ and reference image $\mathtt{I}_0$
Output: set of the states $\{\mathbf{p}_k\}_{k=1}^{K}$
Initialisation: $k = 0$, extract reference measure vector $\mathbf{z}_{\mathcal{W}}^{*} \doteq \mathbf{z}(\mathtt{I}_0, \mathcal{W}, \mathbf{p}_0)$ and compute the initial homography $\mathtt{H}_0$
for $k = 1$ to K (loop on the images) **do**
  $\mathbf{p}' = \mathbf{p}_{k-1}$ and $\mathtt{H}' = \mathtt{H}_{k-1}$
  for $l = 1$ to L (loop on the levels) **do**
    for $i = 1$ to I (loop for each level) **do**
      Observation: features vector extraction $\boldsymbol{\delta}_{\mathbf{z}} = \mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}') - \mathbf{z}_{\mathcal{W}}^{*}$
      Estimation : Motion estimation into the canonical reference frame, then into the image reference frame:

$$\boldsymbol{\delta}_{\mathbf{P}}{}' = \sum_{m=1}^{M} \mathbf{w}_{m,l}\phi_{m,l}(\boldsymbol{\delta}_{\mathbf{z}})$$

$$\tilde{\boldsymbol{\delta}}_{\mathbf{p}}{}' \propto (\mathtt{H}')^{-1}.\tilde{\boldsymbol{\delta}}_{\mathbf{P}}{}'$$

      Update: state vector and homography

$$\mathbf{p}' = \mathbf{p}' + \boldsymbol{\delta}_{\mathbf{p}}{}'$$

      Estimate $\mathtt{H}'$, solving $\tilde{\mathbf{P}}_0 \propto \mathtt{H}'.\tilde{\mathbf{p}}'$
    end for
  end for
  Final update: $\mathbf{p}_k = \mathbf{p}'$ and $\mathtt{H}_k = \mathtt{H}'$
end for

4.1 Datasets and Methodology

Methodology for evaluation of region based tracking methods has been already proposed for both rigid [10, 3, 11, 8] and non rigid objects [4, 7]. Experiments done can be classified in three categories:

1. **Accuracy of motion parameters estimation**. The aim of this experiment is to show the estimated variation of motion parameters related to the true variation. Experimental data used for this test are generated from a static image. Virtual motion parameters variation and a synthetic resulting region are achieved and stored in a ground truth database. Motions are usually generated in a marginalized way (the variation is applied for one parameter, a translation coordinates for example). The tracking algorithm is then tested using the generated dataset and the motion parameter estimation is compared with the true estimation. Several tests are usually achieved related to the most relevant parameters of the tracking algorithm.

2. **convergence frequency**. The aim of this test is to compute the convergence frequency of tracking algorithms. In [3], the algorithm diverges when the final SSD motion error is bigger than the initial SSD. A database containing motions and the resulting synthetic region is generated and then used to compute the rate of convergence of the algorithm. This frequency is then presented compared to the amplitude variation of the motion parameters , or compared to the appearance perturbations like illumination variations (affine or gaussian for example).

3. **illustration on real sequences**. The aim of the experiment is to illustrate the behaviour of the algorithm for several real sequences with illumination variation, clutter background, or partial occlusion. Since ground truth can not be known for such sequence, key samples of the video are extracted and presented with the superimposed tracking region.

The methodology proposed here is close to the one already proposed to evaluate similar methods. We compare two algorithms in terms of accuracy and convergence rate for a synthetic image. Moreover three algorithms have been compared on real videos. Two classical algorithms:

- LBT: the Linear Based Tracker algorithm proposed in [10].

- ESM: The ESM visual tracking software is based on a fast optimization technique called ESM (Efficient Second-order Minimization). The ESM technique has been proposed for improving standard visual servoing techniques. Thanks to its generality, it has been extended to improve template-based visual tracking techniques [3]. Since we have used the ESM MATLAB (TM) toolkit provided by the author, the method has been used with default parameters.

The proposed algorithm:

- KBT : the Kernel Based Tracker.

Since LBT and KBT are very close, several experiments compare the two algorithms in relation with their most relevant parameters. The same learning dataset is used for both algorithms. The latter is generated from random (gaussian law) disturbance of motion parameters (variation of the position of the four corners or the patterns). In the following, we define b as the standard deviation of the gaussian law in percentage of the tracked pattern width.

The learning step of KBT can be achieved by two methods: least square linear optimization or multivariate relevant vector machine learning. The two algorithms have been tested and results obtained are very close ; so the following results are achieved with the second approach because the learning step is much longer for MVRVM.

The observation function is based on the luminance of pixels extracted from a regular sampling of the pattern to track. In the following experiments, the size of the sampling grid is 15×15 pixels (225 points). The resulting

feature vector is then zero-normalized in order to be invariant according to affine luminance transformations. Moreover, all the tests presented here have been realized with a coarse to fine learning strategy using three levels and three loops per level.

4.2 Results

In order to assess the algorithms in different controlled conditions, we synthesize images from a template. Moreover, LBT and KBT algorithms have been implemented on a PC Desktop (PIV 3.2GHz) and run at a 6 ms by image for KBT and 3 ms for LBT.

The first test shows the ability of the method to estimate a known variation (cf. fig. 2). A horizontal displacement of the pattern is generated from a static image. The training step is realized for three levels ($L = 3$), with $N = 400$ and $M = N$ basis functions. Moreover, three iterations are applied to the tracker ($I = 3$). The figure shows the mean translation error related to the translation amplitude (% of the pattern width), for four values of the training parameter b (disturbance magnitude used to build the training set for the first level). The figure shows the accuracy of LBT and KBT algorithm.

The second test shows the convergence frequency of for LBT and KBT, in relation with the disturbance magnitude used for the training set. We define a successful convergence with a threshold on the quadratic error between the estimated position (the four corners of the pattern to track) and the real position. A random perturbation (gaussian law) is applied to the four corners of the pattern and a resulting synthetic dataset is generated using one thousand realisations of this process. Figure 3 shows the convergence rate according to the quadratic motion generated, and for $b = 0.1$ (training parameter) for the left sub figure and $b = 0.2$ for the right sub figure.

The third test compares the convergence frequency for LBT and KBT, in relation with noisy observations. A subset of the observation vector $\mathbf{z}(\mathtt{I}_k, \mathcal{W}, \mathbf{p}_{k-1})$ (randomly selected) has been replaced by a random value drawn from an uniform distribution between 0 and 255 (the gray level range). Figure 6 shows the convergence rate for LBT and KBT related to the percentage of noisy features.

The two above described algorithms have been compared on three sets of real data. For the first set (see fig. 7), the two methods have been compared with ESM. The figure shows the output of the three algorithms for key images of the video. The toy video selected provides high rotations and scale variations. The pattern to be tracked is selected on the first image. The second set is composed by five other toy sequences, with several patterns, motions and illumination conditions. Figure 6 shows the results of the tracking process for 4 sampled images of each sequence. The "Box" sequence is quite simple, with no specularity, slow motion and scale variation. "Crisp 1" and "Crisp 2" sequences contain strong rotations with blur and specularities. "Lipton 1" and "Lipton 2" sequences provides large scale variations. Moreover, for "Lipton 2", the learning step is achieved using a low resolution pattern. For each sequence, we measure the number of successfully tracked frames by each method before divergence of the algorithm. Table 1 summarizes the resulting frequency rates. The third set is composed by four real video sequences extracted from the PETS[†] dataset which provides traffic videos from a camera embedded within a vehicle. One of the related application is collision avoidance or automatic cruise control using rear car vision based tracking. Figure 7 shows the results of the tracking process for 4 sampled images of each sequence and table 1 summarizes the computed frequency rates.

4.3 Discussion

The accuracy test (2) shows that for small training amplitudes $b = 0.1$ and $b = 0.2$, motion translation estimation is correct for values within the training area (lower than the training amplitude b) and accuracy is the similar for LBT and KBT. For high translation, the error increases for both the LBT and the KBT method. For a training amplitude $b = 0.3$, the LBT fails while the KBT still provides correct translation estimation. The

[†]Performance Evaluation of Tracking and Surveillance

Seq. name (#nb. frames)	LBT Nb. (%) of successfully tracked frames	KBT Nb. (%) of successfully tracked frames
Box (#198)	191 (96%)	198 (100%)
Crisp 1 (#200)	15 (7%)	200 (100%)
Crisp 2 (#249)	165 (66%)	249 (100%)
Lipton 1 (#144)	17 (12%)	144 (100%)
Lipton 2 (#341)	107 (31%)	267 (78%)

Table 1: Convergence frequency of LBT and KBT for five toy sequences.

Seq. name (#nb. frames)	LBT Nb. (%) of successfully tracked frames	KBT Nb. (%) of successfully tracked frames
Vehicle. 1 (#495)	60 (12%)	495 (100%)
Vehicle. 2 (#458)	57 (12%)	198 (43%)
Vehicle. 3 (#765)	0 (0%)	180 (23%)
Vehicle. 4 (#97)	7 (7%)	97 (100%)

Table 2: Convergence frequency of LBT and KBT for four sequences related to rear car tracking.

first order approximation used for the LBT learning step is only correct for small motions. Since the KBT is a non-linear model, it is possible to learn the non linear regression function for large motions (until $b = 0.4$).

The convergence rate test presented in fig. 3 shows that KBT has higher convergence rates than the LBT algorithm. The convergence rate of the LBT decreases because the first order approximation made in the method is correct only for small motion. These results are linked to the accuracy test. Since KBT can learn higher motions than LBT, the convergence area is higher for KBT than for LBT. It results a higher convergence basin for KBT.

Fig. 3 shows that the convergence rate of the LBT decreases to 50% with only 0.5% of noisy features. The KBT provides a higher convergence rate than the LBT with 50% of convergence for 4.4% of noisy features. The reason is that when a noisy observation occurs, the vector $\boldsymbol{\delta}_{\mathbf{z}}$ produces high values. For the LBT, $\boldsymbol{\delta}_{\mathbf{z}}$ is directly multiplied with the interaction matrix and generates high displacements. For the KBT,$\boldsymbol{\delta}_{\mathbf{z}}$ is used in kernel functions and the resulting scalar vector is often small; so the motion generated is only slightly modified by the noisy observation. This is an important property of the KBT because if the features are far from the features used in the learning dataset, the estimated motion is near zero.

Both the accuracy and the convergence tests show that KBT has good performances compared to LBT. In order to illustrate these results on real data, six video sequences have been chosen, with different conditions. Table 1 reports the number of successfully tracked frames by each method before divergence. KBT algorithm give a higher successfully tracked frames rate than LBT.

The last experiment illustrates the performance of the method for a rear vehicles tracking application. Figure 7 shows the results of the tracking process for 4 sampled images of each sequence. Table 2 shows the number of successfully tracked frames by each method before divergence of the algorithm. For all the sequences, the frequency rate of the KBT method is higher than the frequency rate of LBT.

5 Conclusion

We have proposed an extention of the linear based planar pattern tracking algorithm to non-linear models. The learning based framework provides an efficient way to build a regression function between observation and motion. Moreover, an empirical rule is proposed to estimate the parameters of the kernel function in order to give an informative design matrix.

This method has been implemented using simple gray-level features and experiments have been performed to compare it to the linear algorithm in terms of accuracy and frequency convergence. For small learning magnitudes, accuracy is the similar for LBT and KBT. Moreover, frequency basin is higher for KBT than for LBT. However, further tests on a large number of patterns are needed to establish this firmly.

The algorithm has been also tested for realtime rear car tracking, a necessary subtask for applications like vision based automatic cruise control or vision based collision avoidance. Results indicates that KBT has better convergence frequency than LBT for the sampled video sequences.

Future works will be done in order to improve robustness against noisy data provided by partial occlusion or specularities. Since the method works at 6 ms for one image, more complex kernel function may be used (robust kernel functions).

References

[1] S. Avidan. Support vector tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, Hawaii, December 2001.

[2] S. Baker and I. Matthews. Lucas-Kanade 20 years on : A unifying framework. *IJCV International Journal on Computer Vision*, 56(3):221–255, 2004.

[3] S Benhimane and E Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE/RSJIROS*, Japan, October 2004.

[4] T.F. Cootes, G.J. Edwards, and Taylor C.J. Active appreance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[5] C. Dehais, M. Douze, V. Charvillat, and G. Morin. Augmented reality through real-time tracking of video sequences using a panoramic view. *Int. Conf. on Pattern Recognition*, 2004.

[6] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[7] V. Gay-Bellile, A. Bartolli, and P. Sayd. Feature-driven non-rigid image registration. In *BMVC, British Machine Vision Conference*. Warwick, United Kindom, September 2007.

[8] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, October 1998.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] F. Jurie and M. Dhome. Real time template matching. In *International Conference on Computer Vision*, pages 544–549, Vancouver, Canada, July 2001.

[11] T. Chateau, F. Jurie, M. Dhome, and X. Clady. Real-time tracking using Wavelets Representation. In *Symposium for Pattern Recognition, DAGM'02,*, pages 523–530, Zurich, September 2002. Springer.

[12] A. Thayananthan, R. Navaratnam, B. Stenger, P. Torr, and R. Cipolla. Multivariate relevance vector machines for tracking. In *ECCV, European Conference on Computer Vision*, 2006.

[13] M.E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Larning Research*, 1:211–244, 2001.

[14] V.N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[15] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. pages 353–361, Nice, France, 2003.

[16] Q. Yu, G. Medioni, and I. Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

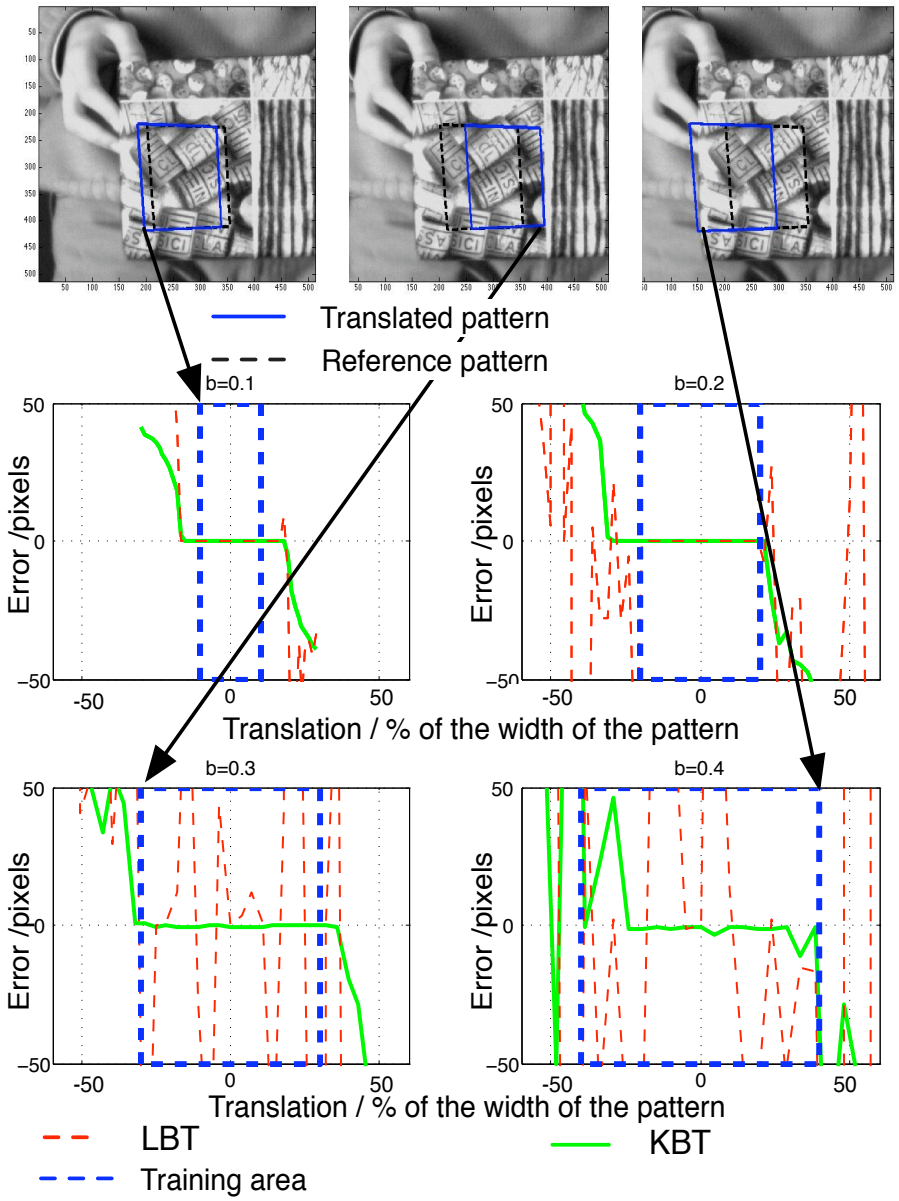

Figure 2: Comparison of the LBT and the KBT in terms of accuracy, against horizontal displacement magnitude, and for four different values of the training parameter b.

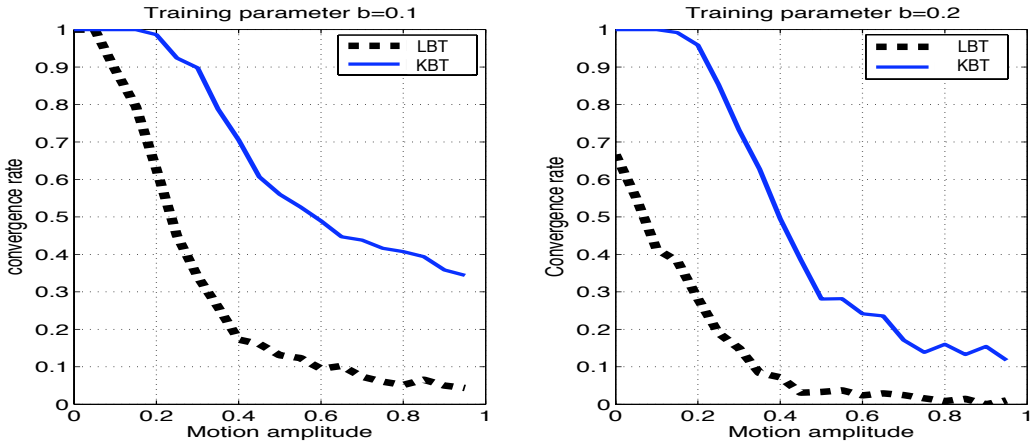

Figure 3: Comparison of the LBT and the KBT method in terms of convergence frequency, against the displacement magnitude

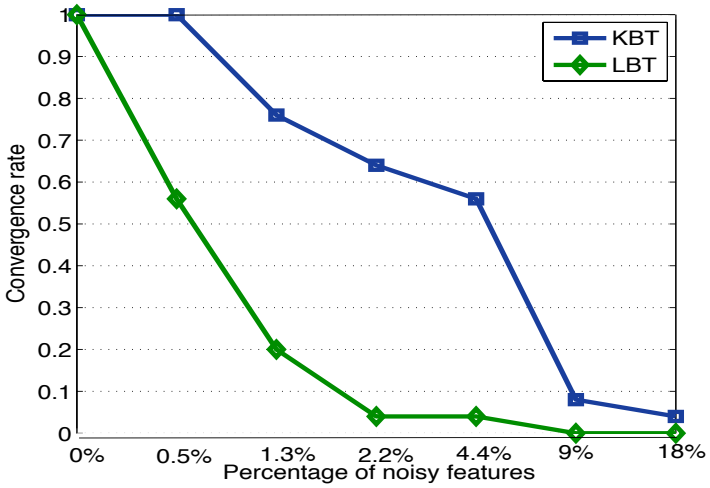

Figure 4: Comparison of the LBT based tracker and the KBT method against noisy observations.

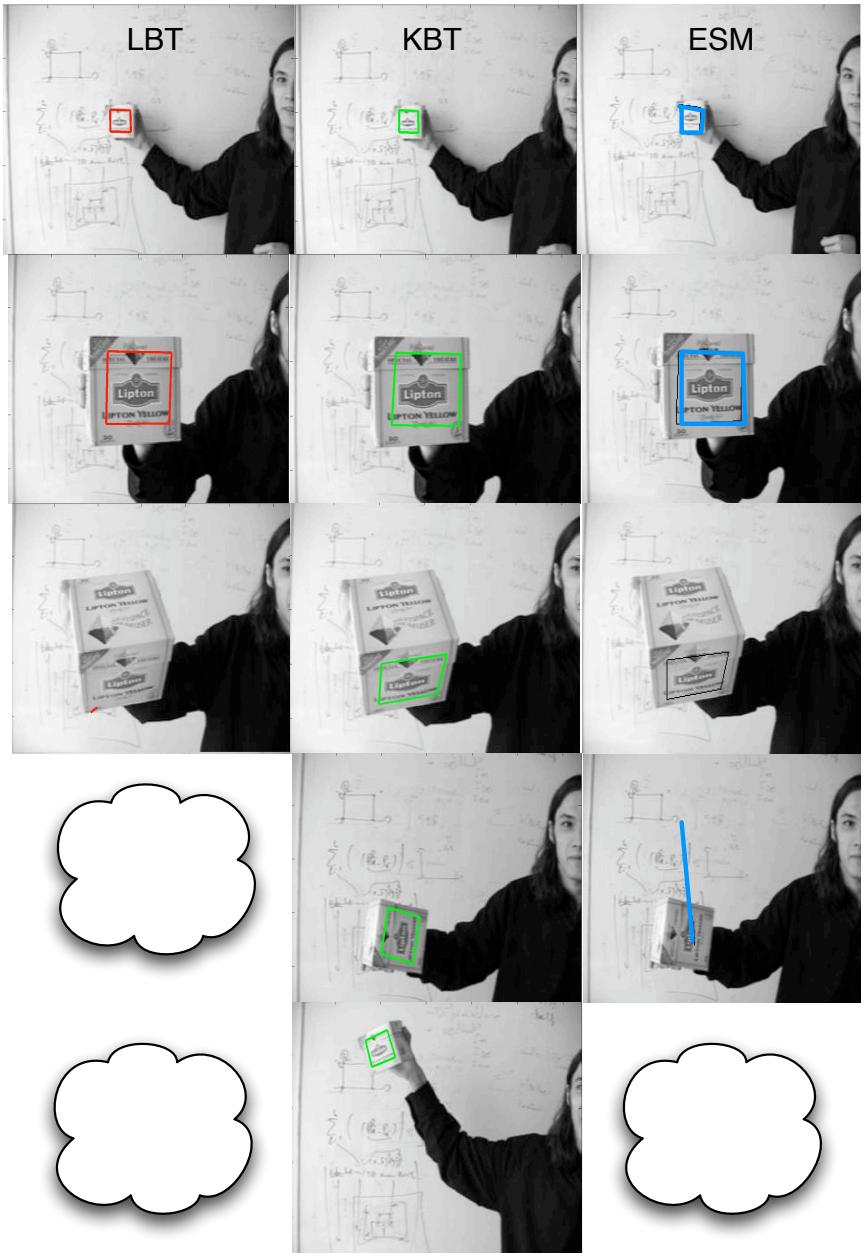

Figure 5: Comparison of three tracking algorithms, on a real video sequence: LBT for the left column, KBT for the median column and ESM for the right column. Default parameters have been used for ESM. Morevover 1000 samples have been generated for the training step of KHT and KBT and 100 basis functions have been used for KBT.

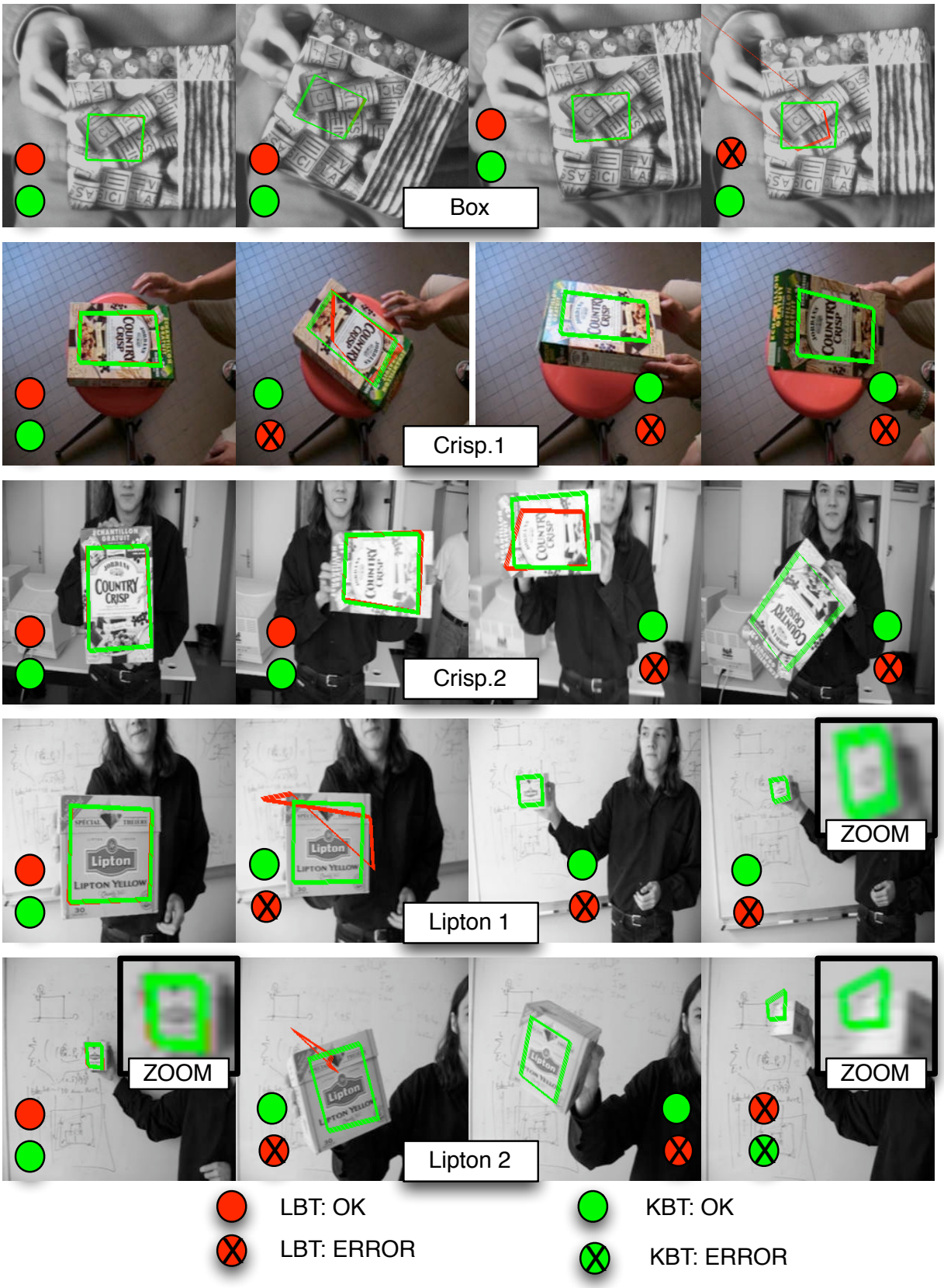

Figure 6: Comparison of the LBT and the KBT for five sequences related to rear car tracking.

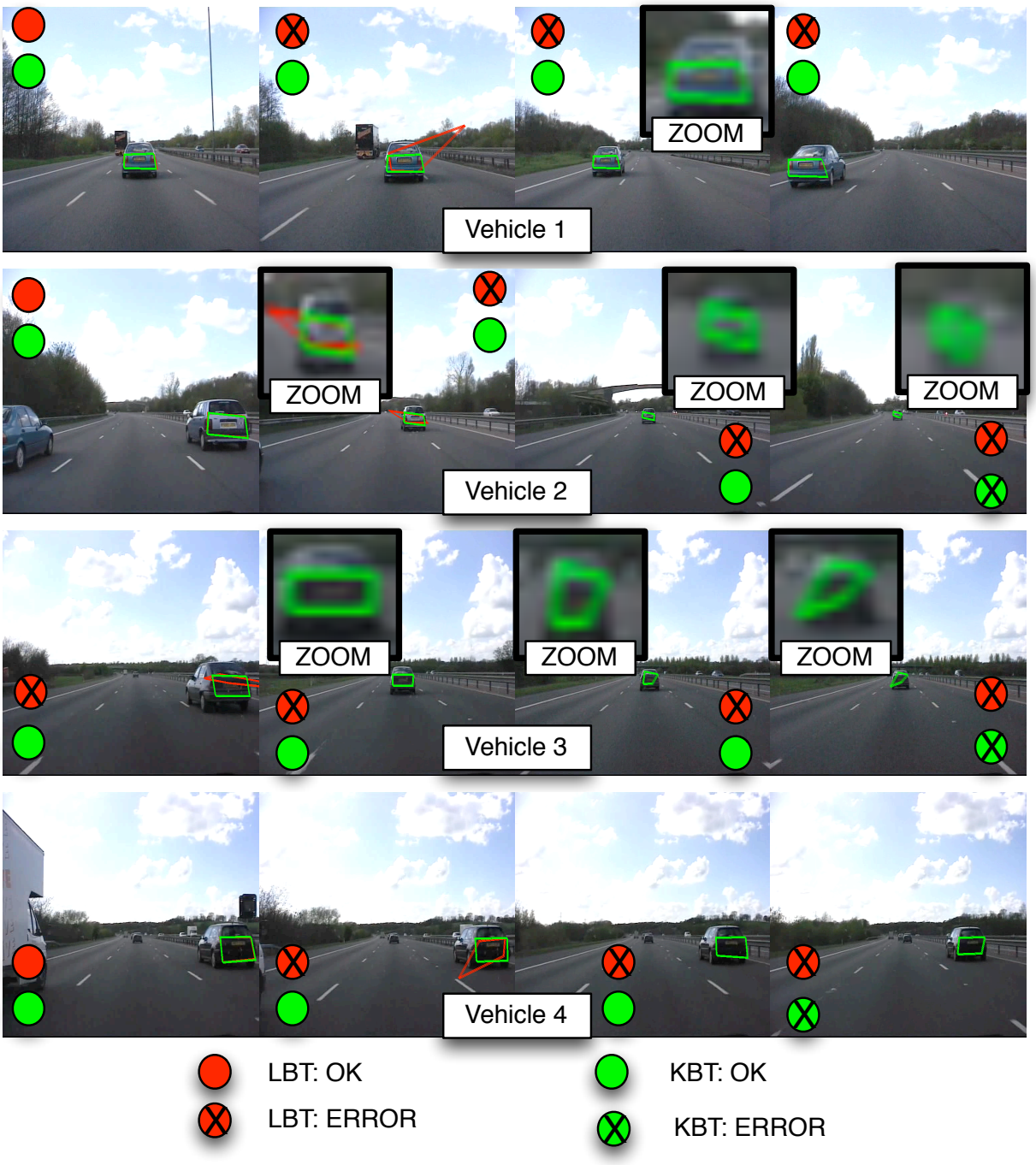

Figure 7: Comparison of the LBT and the KBT for five sequences related to rear car tracking.