On-Line Learning of the Transition Model for Recursive Bayesian Estimation

Samuele Salti
DEIS, University of Bologna
V.le Risorgimento, 2 40136 Bologna, Italy
samuele.salti@unibo.it

Luigi Di Stefano
DEIS, University of Bologna
V.le Risorgimento, 2 40136 Bologna, Italy
luigi.distefano@unibo.it

Abstract

Recursive Bayesian Estimation (RBE) is a widespread solution for visual tracking as well as for applications in other domains requiring hidden state estimation. Although theoretically sound and unquestionably powerful, from a practical point of view RBE suffers from the assumption of complete a priori knowledge of the transition model, that is typically unknown. The use of wrong a priori transition model may lead to large estimation errors or even to divergence. This work proposes to prevent these problems, in case of fully observable systems, learning the transition model on-line via Support Vector Regression. An application of this general framework is proposed in the context of linear/Gaussian systems and shown to be superior to a standard, non adaptive solution.

1. Introduction

The problem of hidden state estimation from noisy measurements is transversal to several disciplines. Recursive Bayesian Estimation (RBE) is the tool typically used to tackle this problem.

One of the main limitations to its use is the requirement to *a priori* specify a process dynamic the user believes the system to generally follow, also referred to as *transition model* . In most cases this model is unknown and is empirically selected among a restricted set of standard ones. This approximate tuning of a recursive Bayesian filter may seriously degrade its performances, that could be optimal (*e.g.*, Kalman filter) or sub-optimal (*e.g.*, particle filter) in case of correct system identification.

In this paper, we propose to learn the transition model on-line without any intervention of the user via Support Vector Machines. We provide first a conceptual solution suitable for the general formulation of RBE and we discuss some points that are valid for every specific instantiation of this general model. Then, we address the linear Gaussian case and provide a detailed solution for the Kalman Filter, dubbed Support Vector Kalman (SVK).

The combination of the strengths of the Kalman filter and the SVM for visual tracking applications has appeared already in [6]. SVM was used to provide an adaptive measurement process to a standard Kalman filter with fixed transition model. The focus was on getting good measurements even in presence of highly similar targets, rather than on improving tracking performances learning on-line the motion traits of the object of interest. More closely related to our work are the efforts produced on the derivation of Adaptive Kalman Filters, that have been studied almost since the introduction of this filtering technique. In fact, our SVK can be seen as a new approach to build an Adaptive Kalman Filter. The main idea behind adaptive filtering schemes is that the basic source of uncertainty is due to the unknown noise statistics, and the proposed solution is to estimate them online from the observed data. One of the most comprehensive contributions is [9]. It reviews proposed approaches and classify them according to four categories: Bayesian Estimation (BE), Maximum Likelihood Estimation (MLE), Correlation Methods (CM) and Covariance-Matching Techniques (CMT). Methods in the first category imply integration over a large dimensional space and can be solved only with special assumptions on the PDF of the noise parameters. MLE requires the solution of a non-linear equation that, in turns, is solvable only under the assumptions that the system is time invariant and completely observable and the filter has reached a steady state. Under these assumptions, however, only a time invariant estimation of the parameters of the noise PDF can be obtained. Correlation Methods, too, are applicable only to time invariant and completely observable systems. Finally, Covariance-Matching Techniques can estimate either process or measurement noise parameters and turn out to provide good and time-varying approximations for the measurement noise when the process noise is known. In [10], an improved correlation method is proposed, but the requirement on the stationarity of the system is not dropped. In the context of visual tracking, [16] presents the application of an Adaptive Kalman Filter. The process and measurement errors are modified in every frame taking into account the degree of occlusion of the target:

greater occlusion corresponds to greater value of measurement noise and vice versa. The two noises always sum up to one. In the extreme case of total occlusion, measurement noise is set to infinity and process noise to 0. In [17], the term Adaptive refers to an adaptive forgetting factor, that is used to trade off the contribution to the covariance estimate for the current time step of the covariance estimate for the previous time step and the process noise. This is done in order to improve the responsiveness of the filter in case of abrupt state changes. With respect to all these proposals, our SVK makes less assumptions on the system, the only one being its complete observability. Moreover, unlike BE, MLE and CM techniques SVK provides a time-varying noise statistics estimation. Unlike [16], SVK is not specifically conceived for visual tracking and, hence, generally applicable. Finally, it is worth pointing out that, unlike all reviewed approaches, SVK is adaptive in a broader sense, for it identifies on-line not only the process noise statistics but also the transition matrix.

2. Recursive Bayesian Estimation

The problem of hidden state estimation from noisy measures in discrete-time systems is typically solved via Recursive Bayesian Estimation. In this framework, following the notation from [1], the system is completely specified by a first order Markov model compound of

- a law of evolution of the state,

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \nu_k) \tag{1}$$

  where $k \in \mathbb{N}$ is the discrete time variable, $\mathbf{x}_k$ is the state at time k, ν_k is an i.i.d. process noise sequence and f_k is a possibly non-linear function relating the state at time k with the previous one.

- a measurement process,

$$\mathbf{z}_k = h_k(\mathbf{x}_k, \eta_k) \tag{2}$$

  where $\mathbf{z}_k$ is the noisy measurement at time k, η_k is an i.i.d. measurement noise sequence and h_k is a possibly non-linear function relating the measurement at time k with the current state.

- an initial state $\mathbf{x}_0$, that in the context of Bayesian reasoning is known via its PDF $p(\mathbf{x}_0)$.

From a Bayesian point of view, the problem of estimating the state translates into the problem of estimating a degree of belief in its possible values, *i.e.* its PDF, given all the available information, *i.e.* the initial state and all the measurements up to a given moment. To reason probabilistically, first of all two PDFs must be derived from the known laws of state space dynamics and measurement: from f_k one can obtain the transition model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ and from h_k one can get the observation likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$. Then, the solution is seeked recursively: given the PDF of the state for time $k-1$ and the availability of a new measurement, a new estimate for the PDF at time k is computed. A general but conceptual solution can be obtained in two steps: prediction via the Chapman-Kolmogorov equation and update using the current measure $\mathbf{z}_k$ (details can be found in [1]). This conceptual solution is analytically solvable only in few cases. A notable case is when the state dynamics and the measurement equations are linear and the noises are Gaussian. In this situation, the optimal solution is provided by the Kalman filter [7].

2.1. On-line learning of the motion model

One of the main disadvantages of RBE is the need to *a priori* specify the transition model. The difficulty of identifying a proper transition model for a specific application typically leads to an empirical and suboptimal tuning of the estimator parameters. A widespread solution to this problem is to learn the transition model off-line from training sequences. Besides the availability of these training sequences, that, depending on the application context, may be easily as well as nearly impossible to have, the major shortcoming of this solution is that it does not allow for taking advantage of the opportunity to change the transition model trough time, although this would be beneficial and neither the conceptual solution nor the solving algorithms require it to be fixed in time.

In this paper, we propose to overcome the difficulties and the shortcomings due to empirical set-up of the transition model by learning it *on-line* . If the state is completely observable, *i.e.* the h_k function just adds measure noise on the state, as it is the case in most practical applications, the transition model is directly related to the dynamics exhibited by the measures. Hence, it is possible to exploit their temporal evolution in order to *learn* the function f_k, and, implicitly, the PDF $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. That is, we can avoid to define $p(\mathbf{x}_k|\mathbf{x}_{k-1})$, and instead use in its place a learned PDF $\tilde{p}_{\mathbf{z}_{1:k-1}}(\mathbf{x}_k|\mathbf{x}_{k-1})$, derived from a learned $\tilde{f}_{\mathbf{z}_{1:k-1}}$. Here, $\tilde{p}_{\mathbf{z}_{1:k-1}}$ formally indicates that the PDF is learned using as training data the relationships between all the consecutive measures from 1 to $k-1$.

Furthermore, we propose to learn the motion model using Support Vector Machine in ϵ-regression mode (SVR). SVMs are well known and effective tools in pattern recognition based on the statistical learning theory developed by Vapnik and Chervonenkis. Their use as regressors is probably less popular but even in this field they obtained excellent performances [15].

2.2. SVMs in ϵ-regression mode

To introduce SVMs as regressors, and in particular in ϵ-regression mode, let us have a quick look at the regression of a linear model given a series of data $(\mathbf{x}_i, y_i)$. In ϵ-regression mode the SVR tries to estimate a function of $\mathbf{x}$ that is far from training data y_i at most ϵ and is at the same time as flat as possible. In the linear case, the model to regress is

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{3}$$

and the solution with minimal complexity is given by the solution of the following convex optimization problem

$$\begin{gathered} min \tfrac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ \begin{cases} y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \leq \quad \epsilon + \xi_i \\ y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - b & \geq \quad -\epsilon - \xi_i^* \end{cases} \end{gathered} \tag{4}$$

The constant C is an algorithm parameter and weights the deviations from the model greater than ϵ. The problem is then usually solved using its dual form, that is easier and extensible to estimation on non-linear functions ([15]).

2.3. SVRs for transition model identification

In the context of RBE, given the first order Markovian assumption, one is left with two options to regress f_k: a) to learn it from measures, that is to provide to the SVR as training data at time k the tuples $\langle \hat{\mathbf{x}}_1, \mathbf{z}_2 \rangle, \ldots, \langle \hat{\mathbf{x}}_{k-2}, \mathbf{z}_{k-1} \rangle$, where $\hat{\mathbf{x}}_k$ stands for the state vector estimate obtained from the recursive Bayesian filter at time k; b) to learn if from states, that is to provide to the SVR as training data at time k the tuples $\langle \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle, \ldots, \langle \hat{\mathbf{x}}_{k-2}, \hat{\mathbf{x}}_{k-1} \rangle$. Going into more details, since the SVR can only regress functions $f : \mathbb{R}^n \to \mathbb{R}$, if the state vector has dimension n, n SVRs are used, and each one is fed with tuples of the form $\langle \hat{\mathbf{x}}_{k-2}, \mathbf{o}^i_{k-1} \rangle$, where the superscript i indicates the i-th component of a vector and $\mathbf{o}$ represents either $\hat{\mathbf{x}}$ or $\mathbf{z}$. Generally speaking, to learn the model from filtered states may induce the learning filter to repeatedly confirm itself, *i.e.* to learn the transition model that itself is imposing on the data. While this effect may guarantee a certain level of smoothness of the output, if this loop degenerates the filter may trust too much the learned model and diverge completely from the real state of the system, ignoring subsequent measures. On the other hand, learning form measures avoids this risk and results in a more responsive filter; yet, for the same reasons, this gives a filter definitely more sensitive to noise, whose effects on the jitter of the output or on the quality of the learned transition model cannot easily be mitigated. Therefore, we advocate the use of the learning from states strategy and will introduce a specific mechanism to avoid degeneracy of the learning loop.

Another choice that may affect performances is the temporal window used to select states (or measures) for training. Clearly, it does not make sense to use all the transitions of states since the beginning of the observations to learn the transition model for the current time slot, or, at least, it does not make sense during regression to equally weight the hint each of them provides. A solution that may be used to address this problem is dynamic SVR for time series regression [3]. While we believe that this may be beneficial, and we plan to experiment with this solution in the near future, so far we have relied on a simpler solution, namely a sliding window of fixed length, to prevent the too old samples of motion from polluting the current estimate.

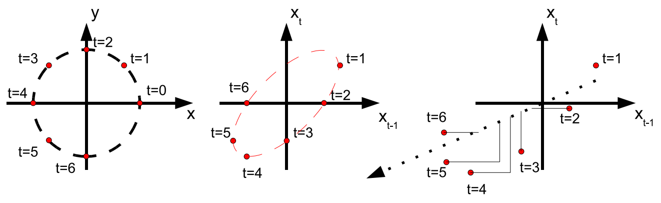

Figure 1. An example showing the importance of the inclusion of the temporal variable among those used for regression.

Finally, the influence of the time variable must be considered during regression. To understand this, consider the circular motion on the unit circle depicted in the leftmost chart of Fig.1. Assuming for clarity of the graphical explanation the state vector to be composed only by the x position of the point, some of the samples from which the SVR has to regress the transition model of this point are depicted in the second chart. As can be seen, without taking into account the evolution of the state through time, even with a perfect regression (represented by the dotted line in the second chart), it is impossible to have a correct prediction of the state at time t, given the state at time $t-1$: for example, at time $t = 4$ and $t = 6$ the previous state, x_{t-1}, is equal for the two positions, but the output of the regression should be different, namely $x_4 = -1$ and $x_6 = 0$. This situation can be disambiguated adding time as an input variable to the function to be regressed, as shown by the last chart. Therefore, in the end the tuples in input to every SVR are of the form $\langle 2, \hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2 \rangle, ..., \langle k-1, \hat{\mathbf{x}}_{k-2}, \hat{\mathbf{x}}_{k-1} \rangle$.

In the following section we address in detail the linear-Gaussian case, when the Kalman filter is the optimal solution, and show how our framework can be instantiated to successfully and advantageously learn the transition matrix on-line.

3. Support Vector Kalman

In the case of linear process and measurement functions, of Gaussian zero-mean noise and of Gaussian PDF for the initial state, all the subsequent PDFs of the state are (multivariate) Gaussians as well. Therefore, they are completely specified by their mean vector, that is usually considered also the estimation of the state, and their covariance matrix. The RBE framework for this case becomes:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \nu_k, E\left[\nu_k \nu_k^T\right] = \mathbf{Q}_k \tag{5}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \eta_k, E\left[\eta_k \eta_k^T\right] = \mathbf{R}_k. \tag{6}$$

Since among the hypothesis of the Kalman filter is the linearity of f_k, two consequences immediately arise: we must use a linear kernel, *i.e.* the SVR formulation introduced in 2.2; and, we must modify it in order to regress a linear function. In fact, the standard function learned by an SVR is (3), *i.e.* an affine mapping. As discussed in [12], a linear mapping can be learned without harming the general theory underneath SVM algorithms. Moreover, a solving algorithm for the linear mapping was also proposed in the paper that introduced the standard and widespread solution for the affine case, *i.e.* the Sequential Minimal Optimization (SMO) algorithm [11].

Using this set-up for the SVRs, it is possible, given the training data in the considered temporal window, to obtain an estimate of F_k. In fact, each weights vector $\mathbf{w}_k$ learned by an SVR at time k is almost directly one of the row of the estimate $\hat{F}_k$ of F_k. This almost refers to a last but not least aspect to be considered in order to regress a truly linear function: the problem of normalization. Typical implementations of SVMs require the input and output to be normalized within the range $[0, 1]$ or $[-1, +1]$. While this normalization may be thought of as neutral as far as the SVR output is concerned, it has subtle consequences when the weights vector of the SVR is used as a dynamic parameter for a Kalman filter. To illustrate this, let us consider a simple example where a mapping from a scalar x to y is learned, and the variables are normalized to the range $[-1, +1]$. Then

$$\tilde{x} = \frac{2x - x_{max} - x_{min}}{x_{max} - x_{min}} \quad \tilde{y} = \frac{2y - y_{max} - y_{min}}{y_{max} - y_{min}}, \tag{7}$$

where the superscript $\tilde{}$ denotes the normalized variables and x_{max}, x_{min} are the maximum and minimum value of the measured variable within the considered temporal window. Hence, the function of x that gives the unnormalized y is

$$\tilde{y} = w\tilde{x} \;\Rightarrow\; y = ax + b,\; a = \frac{(2(y_{max} - y_{min})w)}{(x_{max} - x_{min})}$$
$$b = y_{max} + y_{min} - \frac{(y_{max} - y_{min})(x_{max} + x_{min})w}{x_{max} - x_{min}} \tag{8}$$

i.e., again an affine mapping. Therefore, using the unnormalized coefficient a as an entry of the transition matrix $\hat{F}_k$ results in poor prediction, since the constant term is not taken into account. In order to obtain a linear mapping, that fits directly into the transition matrix of a Kalman filter, a two steps normalization must be carried out. Given a sequence of training data, a first normalization is applied,

$$\hat{x} = x - \frac{x_{max} + x_{min}}{2} \quad \hat{y} = y - \frac{y_{max} + y_{min}}{2}. \tag{9}$$

These are the data on which the Kalman filter has to work. In other words, at every time step, the output of the previous time step must be renormalized if its value changes the minimum or maximum within the temporal window. This is equivalent to a translation of the origin of the state space and does not affect the Kalman algorithm itself. No normalization is required for the covariance matrix. After this normalization, the data can be scaled in the range $[-1, +1]$, as required by the SVR, according to

$$\tilde{x} = \frac{2}{\hat{x}_{max} - \hat{x}_{min}}\hat{x} \quad \tilde{y} = \frac{2}{\hat{y}_{max} - \hat{y}_{min}}\hat{y}, \tag{10}$$

where the mining of the subscripts is the same as in (7). Using this two steps normalization, the unnormalized function of the Kalman data is

$$\tilde{y} = w\tilde{x} \quad \Rightarrow \quad \hat{y} = \frac{(\hat{y}_{max} - \hat{y}_{min})}{(\hat{x}_{max} - \hat{x}_{min})} w\hat{x}, \tag{11}$$

i.e. the required linear mapping.

A final remark is worth pointing out. The noise model assumed by an SVR is Gaussian, with mean and covariance being random variables whose distributions depend on C and ϵ [13]. The mean, in particular, is uniformly distributed between $-\epsilon$ and ϵ. Therefore, the SVR noise model is a superset of that assumed by the Kalman filter, *i.e.* a zero-mean Gaussian. In other words, the SVR is a theoretically sound regressor to apply in all the set-ups when the Kalman is the optimal filter.

3.1. Adaptive process noise model

As we have seen in the Introduction, the classical definition of an adaptive Kalman filter is more concerned with dynamic adjustment of $\mathbf{Q}_k$ than with the adaptation of the transition model [10, 17]. Our proposal makes it easy to learn on-line the value of $\mathbf{F}_k$, but provides also a sound and efficient way to dynamically adjust the value of the process noise. The value of $\mathbf{Q}_k$, in fact, is crucial for the performances of the Kalman filter. In particular, the ratio between the uncertainties on the transition model and on the measures tunes the filter to be either more responsive but more affected by noise or smoother but with a greater latency in reacting to sharp changes in the dynamics of the system.

Within our framework, a probabilistic interpretation of the output of the SVR allows to dynamically quantify the degree of belief on the regressed motion model, and, consequently, the value of $\mathbf{Q}_k$. Some works have been done already on a probabilistic interpretation of the output of a

SVR [5, 4, 8]. All of them look for error bars on the prediction, *i.e.* the variance of the prediction. Therefore they are all suitable for estimating a Gaussian covariance matrix for the regression output. We chose to use [8] since it is the simplest method and turned out also the most effective in the comparison proposed in [8].

Given a training set, this method performs k-fold cross validation on it and considers the histogram of the residuals, *i.e.* the difference between the known function value at $\mathbf{x}_i$ and the value of the function regressed using only the training data not in the $\mathbf{x}_i$ fold. Then it fits a Gaussian or a Laplace PDF to the histogram, using a robust statistical test to select between the two PDFs. In our implementation, in accordance with the hypothesis of the Kalman filter, we avoid the test and always fit a Gaussian, *i.e.* we estimate the covariance as the mean squared residual. We also keep $\mathbf{Q}_k$ diagonal for simplicity. Hence, every SVR provides only the value of the diagonal entry of its row of $\mathbf{Q}_k$. As discussed before, however, learning from states is prone to degeneration of the learning loop into a self confirming filter unaffected by measures. To avoid this, we prevent the covariance of every SVR to fall down a predetermined percentage of the corresponding entry of $\mathbf{R}$. This has experimentally proved to be effectively enough to avoid the coalescence of the filter while at the same time preserving its ability to dynamically adapt the values of $\mathbf{Q}$.

Finally, this method of update of the process noise allows for an intuitive interpretation of C. Since it weights the deviations from the regressed function greater than ϵ, it is directly related with the smoothness of the Support Vector Kalman output. In fact, if C is kept high, errors will be highly penalized, and the regressed function will tend to overfit the data, leading to greater residuals during the cross validation and to a bigger uncertainty on the transition model. This will result in a more noisy output of the Kalman estimation. If, instead, C is kept low, the SVR will generalize better and the residuals during the cross validation will be less significant. The resulting tighter covariances will produce a smoother estimate by the Kalman filter.

4. Experimental results

We provide first two simulations concerning a simple 1D estimation problem (*i.e.*, a point moving along a line). In the first experiment, the motion is kept within the assumptions required by the Kalman filter, in particular there is a linear relationship between consecutive states. In the second one, a case of non-linear motion is considered. Finally, we provide experimental results concerning tracking of the 3D position and orientation of a moving camera for real-time video augmentation.

4.1. Simulation of linear motion

In both simulations, comparisons have been carried out versus three Kalman filters adopting different motion models: drift (Kalman DR), constant velocity (Kalman CV) and constant acceleration (Kalman CA). Two different tunings were considered for each Kalman filter: a more responsive one, when $\mathbf{Q}$ has been set equal to $10^{-2}\mathbf{R}$; and a smoother one, with $\mathbf{Q} = 10^{-4}\mathbf{R}$. As far as SVK is concerned, it was fed with noisy measures of the position and the velocity of the point, therefore regressing a 2×2 model matrix. The only rough tuning regards C, which is taken equal to 2^{-10} in this simulation and to 2 in the non-linear case: intuitively, an easier sequence allows for using a smoother filter.

During the linear motion sequence, motion is switched every 160 samples among a constant acceleration, a constant position and a constant velocity law. Therefore, each Kalman filter has a slice of time when the real motion performed by the point is exactly that described by its transition matrix. Results on the whole sequence are reported in Fig.2 and Tab.1. As for simulation parameters, $\mathbf{R}$ has been kept constant in time and equal to $100 * \mathbf{I}$, with $\mathbf{I}$ denoting the identity matrix; constant acceleration was $30.0\ m/s^2$, constant velocity was $1000\ m/s$ and Δt was 0.5. Gaussian noise with covariance matrix $\mathbf{R}$ was added to the data to produce noisy measurements for the filters.

As shown by the first column of Tab.1, our proposal achieves the best Root Mean Squared Error (RMSE) on the whole sequence. This is an encouraging proof of the benefits that on-line learning of the transition model can produce on the hidden state estimate. This is also shown by the two charts in the first row of Fig.2. At the scale of the charts, the estimation of our filter is indistinguishable from the real state of the system, whereas the delay of Kalman DR and the overshots/undershots of Kalman CA and Kalman CV in presence of sharp changes of motion are clearly visible.

Going into more details, we separately analyze each of the three different parts of motion. Here, we discuss not only the performance on the whole interval associated with each motion law, but, also, those achieved in the final part of each interval (*i.e.*, the last 80 samples). In fact, final samples allow to evaluate the accuracy of the steady state of the estimators, filtering out the impact of possible delays due to the filter degree of responsiveness.

During the constant acceleration interval, Kalman CA performs best, both with the responsive and the smooth tuning. This is reasonable, since theoretically it is the optimal filter for this specific part of motion. What is important is that our filter performs slightly worse than Kalman CA, but definitely better than Kalman CV and Kalman DR (2-nd column of tab.1). This is also demonstrated by the first chart on the second row of Fig.2, which, for better visualization, displays only absolute errors less than 50. Only our filter stays in the visualized range, apart from the optimal one.

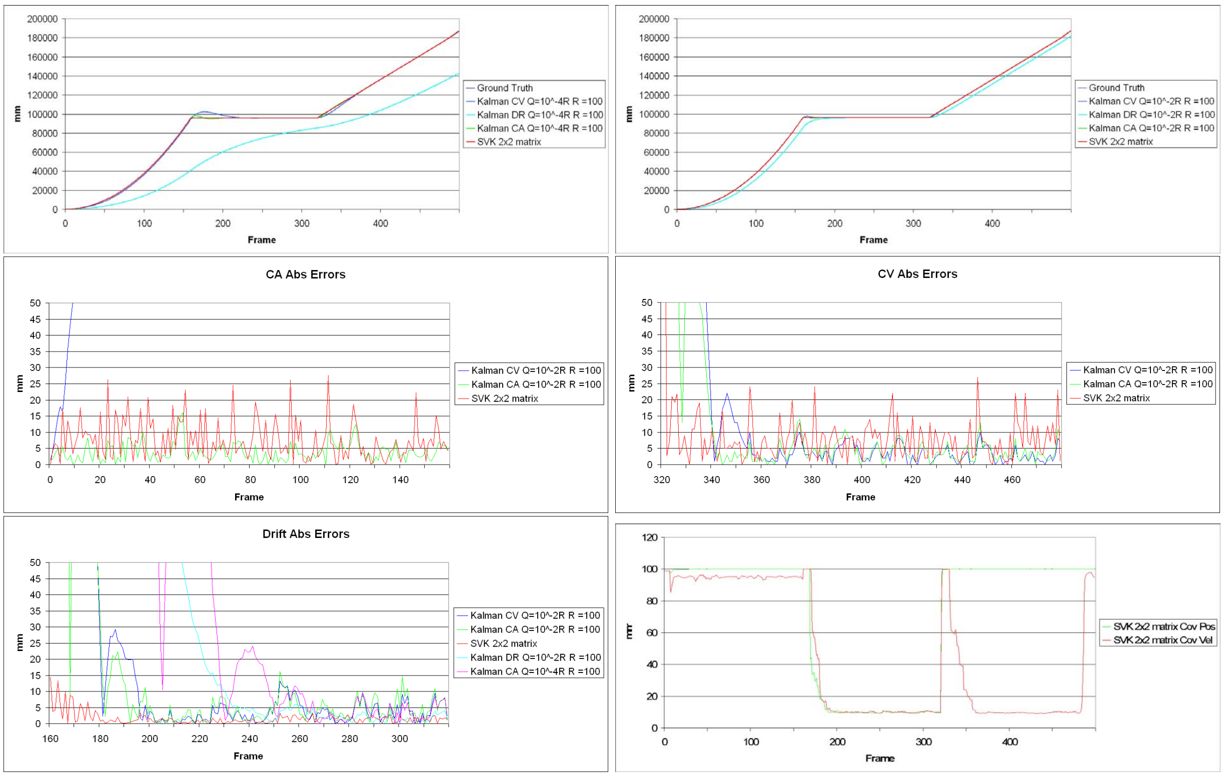

Figure 2. The charts in the first row show the evolution of the different filters against ground truth data in case of linear motion: the one on the left compares SVK to Kalman filters tuned for smoothness, that on the right to Kalman filters tuned for responsiveness. The three subsequent charts report absolute errors for, respectively, the constant acceleration, the constant velocity and the constant position intervals. Finally, last chart shows the covariances on state variables provided by SVK throughout the whole sequence.

When considering only the steady state part (5-th column of tab.1) the analysis does not change, partly because this interval is the very first one and, hence, there are no delays to recover, and partly because the Kalman CV and DR do not have the proper transition matrix for this part and, thus, cannot recover from errors.

During the constant velocity part, SVK has the best overall RMSE (3-rd column of tab.1). This is due to the delay accumulated by Kalman CV, theoretically the optimal filter, during the previous intervals. Therefore, we can highlight one of the major advantages brought in by SKV: in case of sharp changes of the motion law, dynamical update of parameters renders SVK even more accurate than the optimal filter due to its higher responsiveness. This is confirmed by the second chart on the last row of Fig.2, showing the progress of the position and the velocity covariances of SVK. It can be seen that, immediately after the change of motion from constant position to constant velocity at sample 320, both covariances significantly increase, somehow "detecting" such a change, thanks to the adaptive process noise modeling embodied into our filter. The resulting lower confidence in the predictions automatically turns the filter from smoothness to responsiveness, preventing the overshots/undershots exhibited by standard Kalman filters. After few samples the covariance on the velocity decreases again, proving that SVK has confidently learned the new model. Considering only the steady state (6-th column of tab.1) Kalman CV is, as expected, the best one. Unlike the CA interval, however, only the responsive tuning performs well since the smoother Kalman CV has accumulated too much delay to recover. This difference is due to the intrinsically higher smoothness of the CV model with respect to the CA one. Kalman CA, with both tunings, is the second best and this is also predictable since a constant velocity motion may be seen as a special case of a constant acceleration one. Again, the most important finding is that SVK is by far closer to the optimal filters than to those adopting a wrong motion model. And again, visualizing only errors less than 50, it is the only one visible in the corresponding chart of Fig.2, apart from the optimal ones.

Finally, due to the delay accumulated by the other filters, SVK turns out the best estimator also in the constant position interval (4-th column of Tab.1). As far as the steady state is concerned, all the filters exhibit a good RMSE apart from the very smooth ones, namely CV and DR tuned towards smoothness, for the latter do not recover from delays even after 80 samples. Unlike the other motion intervals, SVK keeps on being the best, even when the steady state

Filter	Whole	CA	CV	Drift	CA*	CV*	Drift*	R=1000
SVK 2x2 Model	22.41	9.79	38.02	35.41	8.91	9.63	1.67	43.36
Kalman CA $Q = 10^{-2}R$	76.62	4.83	51.3	125.87	4.59	4.55	6.06	79.65
Kalman CA $Q = 10^{-4}R$	357.45	4.26	242.19	581.52	3.72	4.04	7.87	357.69
Kalman CV $Q = 10^{-2}R$	227.38	100.12	155.13	355.71	104.84	3.74	5.31	228.08
Kalman CV $Q = 10^{-4}R$	1680.37	1213.78	1160.73	2439.37	1416.30	49.82	109.30	1681.04
Kalman DR $Q = 10^{-2}R$	4498.51	6015.22	4536.67	1793.30	8056.45	4757.75	2.77	4500.00
Kalman DR $Q = 10^{-4}R$	29698.38	25771.38	31583.97	29279.53	35763.45	37809.42	16743.08	29699.11

Table 1. Comparison of RMSE on linear motion: first column reports the RMSEs on the whole sequence; then, partial RMSEs on each piece of motion are given as well as RMSEs concerning only the final part of each interval (marked with *), when the filter may have reached the steady state; finally, RMSEs on the whole sequence for higher measurement noise are reported in the last column

only is considered. A reason for this is provided again by the chart of covariances. During the constant position part, the SVR is able to regress a very good transition matrix and both the uncertainties are kept really low compared to the values in $\mathbf{R}$. Therefore, the filter is highly smooth, as can be seen in the chart of absolute errors, and this keeps the RMSE low also in the last part.

Our proposal is robust to higher measurement noise, too. As an example, we report in the last row of Tab.1 the RMSEs for the same simulation, but with $\mathbf{R} = 1000\mathbf{I}$. Even in this case SVK turns out to be the overall best thanks to its adaptive behavior. Considerations similar to previous ones apply to the three different parts of motion, whose data cannot be reported in this publication due to space limits.

To summarize, simulations with linear motion laws show that the proposed SVR-based approach to on-line learning of the transition model is an effective solution for the tracking problem when the assumption of stationary transition matrix cannot hold because the tracked system undergoes significant changes in its motion traits.

4.2. Simulation of non-linear motion

Given its ability to dynamically adapt the transition matrix, we expect SVK to be superior to a standard Kalman filter also in the case of non-linear motion. Hence, to assess its merits we have run simulations with a motion compound of two different sinusoidal parts linked by a constant position interval. The motion law of the two sinusoidal parts is as follows:

$$\begin{aligned} x_1(t) &= 300t + 300\sin(2\pi t) + 300\cos(2\pi t), \quad (12) \\ x_2(t) &= 300t - 300\sin(2\pi t) - 300\cos(2\pi t). \quad (13) \end{aligned}$$

Aggregate results are shown in Fig.3 and Tab.2 for the same levels of measurement noise as in 4.1. Our filter proves again to be the overall best.

4.3. 3D camera tracking

In this experiment, we track the 3D position of a moving camera in order to augment the video content, taking as measurement the output of a standard pose estimation algorithm [14] fed with point correspondences established matching invariant local features [2]. Results consist of videos available on our website [1] and as supplementary material. Video `ARnormalFPS.avi` shows side-by-side the augmentation resulting from the use of Kalman CA and our SVK. Both filters have been tuned to be as responsive as in 4.2 and measurement noise covariances has been adjusted to match the range of the input data. The video shows a fast change of motion of the camera, the purpose of filters being to keep the virtual object spatially aligned with the reference position, denoted for easier interpretation of results by a white sheet of paper. We can see that both filters exhibit a delay following the sharp motion change, but SVK is subject to a smaller translation error, recovers much faster and, unlike Kalman CA, without any overshot. Since it might be difficult to analyze the different behaviors of filters at normal speed, video `ARslowFPS.avi` plays the same sequence as video `ARnormalFPS.avi` at a lower frame rate.

5. Conclusions and Future Work

A new approach to build an adaptive recursive Bayesian estimation framework has been presented, both from a conceptual point of view and in terms of its practical instantiation in the case of linear transition and measurement models and Gaussian noise. Its performances are definitely encouraging: the proposed SVK filter has been shown to outperform a standard Kalman solution, while, at the same time, requiring less parameters to be arbitrarily (and possibly wrongly) tuned. As for the linear/Gaussian scenario, we plan to evaluate the proposed approach against comparable solutions for adaptive Kalman filtering (*i.e.* Covariance Matching Techniques and [17]). More broadly speaking, we see this work as a step towards a general and parameters free tracking system. Endowing this vision, future work will be directed to: a) the instantiation of our proposal also in the case of non linear and non Gaussian tracking, in particular adapting it in order to be beneficially used also with parti-

[1]`http://www.vision.deis.unibo.it`

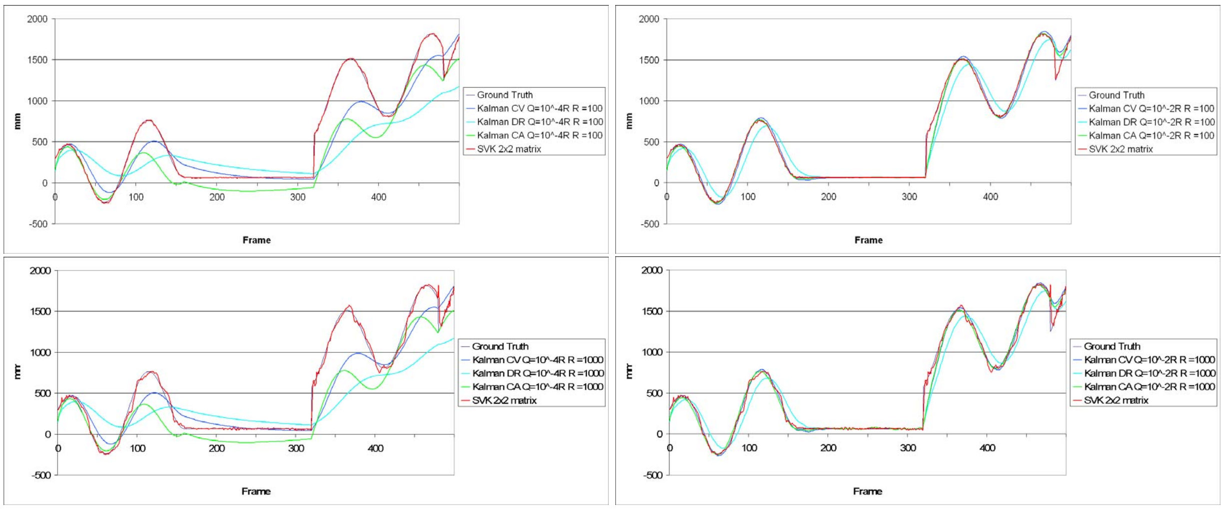

Figure 3. Simulation dealing with non-linear motion. Charts on the left compares SVK to Kalman filters tuned for smoothness, those on the right to Kalman filters tuned for responsiveness. The top row reports experiments with $\mathbf{R} = 100 * \mathbf{I}$, the last row with $\mathbf{R} = 1000 * \mathbf{I}$. With the adopted scale, in the first two charts the estimation of our filter is almost indistinguishable from the ground truth.

R = 100	Whole	R=1000	Whole
SVK 2x2 Model	20.61	SVK 2x2 Model	47.98
Kalman CA resp.	61.92	Kalman CA resp.	62.32
Kalman CA smooth	308.32	Kalman CA smooth	308.66
Kalman CV resp.	72.69	Kalman CV resp.	72.95
Kalman CV smooth	248.30	Kalman CV smooth	248.46
Kalman DR resp.	143.63	Kalman DR resp.	144.87
Kalman DR smooth	434.83	Kalman DR smooth	435.20

Table 2. Comparison of RMSE on non-linear motion.

cle filters and b) to the insertion of algorithms for automatic on-line selection of best SVR parameters.

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] L. Cao and Q. Gu. Dynamic Support Vector Machines for non-stationary time series forecasting. *Intelligent Data Analysis*, 6:67–83, 2002.

[4] W. Chu, S. Keerthi, and C. J. Ong. Bayesian Support Vector Regression using a unified loss function. *Neural Networks, IEEE Transactions on*, 15(1):29–44, Jan. 2004.

[5] J. Gao, S. Gunn, C. Harris, and M. Brown. A probabilistic framework for SVM regression and error bar estimation. *Machine Learning*, 46:71–89(19), 2 January 2002.

[6] A. Garg, I. Cohen, and T. S. Huang. Adaptive learning algorithm for SVM applied to feature tracking. In *Information Intelligence and Systems 1999, International Conference on*, 1999.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8] C.-J. Lin and R. C. Weng. Simple probabilistic predictions for Support Vector Regression. Technical report, Department of Computer Science, National Taiwan University,, 2004.

[9] R. Mehra. Approaches to adaptive filtering. *Automatic Control, IEEE Transactions on*, 17(5):693–698, Oct 1972.

[10] M. Oussalah and J. De Schutter. Adaptive Kalman filter for noise identification. In *25th International Conference on Noise and Vibration Engineering*, 2000.

[11] J. C. Platt. Fast training of Support Vector Machines using Sequential Minimal Optimization. *Advances in kernel methods: support vector learning*, pages 185–208, 1999.

[12] T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. b. *CBCL Paper 198/AI Memo 2001-011*, 2001.

[13] M. Pontil, S. Murkerjee, and F. Girosi. On the noise model of Support Vector Machine Regression. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1998.

[14] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 28(12):2024–2030, Dec. 2006.

[15] A. J. Smola and B. S. Olkopf. A tutorial on Support Vector Regression. Technical report, Statistics and Computing, 1998.

[16] S.-K. Weng, C.-M. Kuo, and S.-K. Tu. Video object tracking using adaptive Kalman filter. *J. Vis. Comun. Image Represent.*, 17(6):1190–1208, 2006.

[17] Y. Zhang, H. Hu, and H. Zhou. Study on adaptive Kalman filtering algorithms in human movement tracking. In *Information Acquisition, 2005 IEEE International Conference on*, pages 5 pp.–, June-3 July 2005.