

Visual Tracking via Incremental Self-tuning Particle Filtering on the Affine Group

Min Li, Wei Chen, Kaiqi Huang, Tieniu Tan
National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences
{mli, wchen, kqhuang, tnt}@nlpr.ia.ac.cn

Abstract

We propose an incremental self-tuning particle filtering (ISPF) framework for visual tracking on the affine group. SIFT (Scale Invariant Feature Transform) like descriptors are used as basic features, and IPCA (Incremental Principle Component Analysis) is utilized to learn an adaptive appearance subspace for similarity measurement. ISPF tries to find the optimal target position in a step-by-step way: particles are incrementally drawn and intelligently tuned to their best states by an online LWPR (Local Weighted Projection Regression) pose estimator; searching is terminated if the maximum similarity of all tuned particles satisfies a target similarity distribution (TSD) modeled online or the permitted maximum number of particles is reached. Experimental results demonstrate that our ISPF can achieve great robustness and very high accuracy with only a very small number of random particles.

1. Introduction

Visual tracking has received great attention because of its critical role in many computer vision tasks, such as intelligent visual surveillance, human-computer interaction, augmented reality, and driver assistance.

Tracking strategy plays an important role in the efficiency and robustness of visual tracking. Though different tracking strategies may seem very different, they can be coarsely divided into two categories: deterministic models [9, 4, 7, 14, 2, 3, 19, 22] and stochastic models [10, 5, 15, 1, 21]. Deterministic model often estimates target's position directly from low-level features by a gradient descent search to a cost function or by online learning. There is a variety of such tracking algorithms. In template matching based methods [9, 4], the SSD (Sum of Squared Difference) between the candidate target and a fixed template is defined as the cost function and motion parameters are obtained by steepest descent optimization. In Kernel based methods [7, 14], histogram features are used as object representations and regularized by a mask which induces

spatially-smooth similarity functions suitable for gradient-based optimization. Then tracking is formulated as finding the local maxima of the object similarity function by mean-shift iterations. In online boosting based methods [2, 3], tracking is treated as a pixel-level classification problem between object and background. A strong classifier is trained online by Boosting and used to label pixels to give a confidence map, the peak of which is found using mean shift. In regression based methods [19, 22], object pose is directly estimated from low-level features by a learned regression mapping without any criterion function for state estimation. Although deterministic models are usually computationally efficient, they highly depend on low-level feature data, and thus easily become trapped in local minima or produce unpredictable results due to great appearance changes.

Comparing to deterministic model, stochastic model takes the measurement and model uncertainties into account during tracking. This strategy usually formulates tracking as a problem of Bayesian inference in state space. If the state posterior density is a Gaussian (or can be approximated as such), Kalman filter [5], EKF (Extended Kalman Filter) [5] or UKF (Unscented Kalman Filter) [15] can be used to find the optimal/suboptimal solution. However, most real tracking problems are usually nonlinear and non-Gaussian, and thus can not be solved by classical Kalman filter. Condensation [10], known also as SMC (Sequential Monte Carlo)[8] or particle filtering [6], is proposed for implementing a recursive Bayesian filter by MC simulations. With almost no strict assumption on the dynamical model and measurement model, condensation and its variants (now commonly called particle filtering) [1] are widely used in nonlinear non-Gaussian problems. The key idea of particle filtering is to represent the required posterior density function by a set of random samples with associated weights. Though particle filtering has a lower probability to be trapped in local maxima, the optimal importance function for sampling is often not available, so usually a very large number of particles (drawn from the prior dynamic model) are needed to approximate the posterior density.

We propose a robust and efficient tracking framework by incorporating an online pose estimator (deterministic model) into an incremental particle filtering framework (stochastic model), called incremental self-tuning particle filtering (ISPF). The main contributions of this paper include: 1) IPCA (incremental principle component analysis) [18] is utilized to learn an adaptive appearance subspace, based on which two similarity distributions TSD/BSD (target/background similarity distribution) are modeled and used as certain criterion in state optimization or model update. 2) LWPR (Local Weighted Projection Regression) [20] is introduced to learn an online pose estimator. Comparing to rigid regression based pose estimator [19], the LWPR pose estimator does not suffer from high-dimension feature space for it uses PLS (partial least squares)[17] to reduce feature dimension; furthermore, it does not assume the global regression function is linear because it can approximate nonlinear functions by piecewise linear models; moreover, LWPR pose estimator can be updated incrementally. 3) An incremental particle filtering framework is proposed based on 1) and 2). With the help of the learned pose estimator and feedback similarity scores, our ISPF can achieve great robustness and very high accuracy by using a very small number of particles.

The remainder of the paper is organized as follows. Section 2 gives an overview of the proposed framework. Section 3 introduces basic knowledge of the affine group. The subspace based appearance model and LWPR based pose estimator are described in Section 4 and Section 5 respectively. Then, the incremental particle filtering framework is proposed in Section 6. Experimental results and analysis are presented in Section 7. Finally, we draw our conclusions in Section 8.

2. Overview of the proposed framework

In particle filtering framework, tracking problem is casted as an inference task in a Markov model with hidden state variables [1]:

$$p(\mathbf{X}_t|\mathbf{O}_t) \propto p(\mathbf{o}_t|\mathbf{X}_t) \int p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{O}_{t-1})d\mathbf{X}_{t-1} \quad (1)$$

where \mathbf{X}_t describes the state variable of the tracked target at time t and $\mathbf{O}_t = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t\}$ is a set of observations. The tracking process is governed by the observation model $p(\mathbf{o}_t|\mathbf{X}_t)$ and the dynamic model $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ between two states. Here, \mathbf{X} is a 2D affine transformation matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{C} & \mathbf{e} \\ 0 & 1 \end{pmatrix} \quad (2)$$

where \mathbf{C} is a nonsingular 2×2 matrix and $\mathbf{e} \in \mathbb{R}^2$. All affine transformations form the affine group (matrix Lie group).

Figure 1 shows the flowchart of our tracking framework. At frame #0, we use the initial state \mathbf{X}_0 to initialize the IPCA based appearance model (AM), based on

which two distributions: TSD (Target Similarity Distribution) and BSD (Background Similarity Distribution) are obtained. The similarity here measures how similar a patch is to the learned appearance subspace. TSD is the similarity distribution of target patches (specified by optimal states which correctly contain the tracked target) and BSD is that of background patches. Both are approximated by a Gaussian and denoted by $\mathcal{N}_T(\mu_T, \sigma_T)$ and $\mathcal{N}_B(\mu_B, \sigma_B)$ respectively. Moreover, a pose estimator (PE) is learned by the LWPR algorithm with a pose training set generated by performing small variations around the initial state. At frame # t , the particle filtering process incrementally draws random samples on the affine group and uses the pose estimator to tune them to their best states. No more particles will be drawn if the maximum similarity score of all tuned particles is larger than $\mu_T - \sigma_T$ or the maximum number of particles is reached. Target's state $\hat{\mathbf{X}}_t$ is estimated by the sample with the maximum similarity score \hat{S}_t . If the final maximum similarity score is not too bad, *e.g.* $\hat{S}_t > \mu_B + \sigma_B$, the IPCA AM, BSD, TSD and PE will be updated based on the estimated state $\hat{\mathbf{X}}_t$.

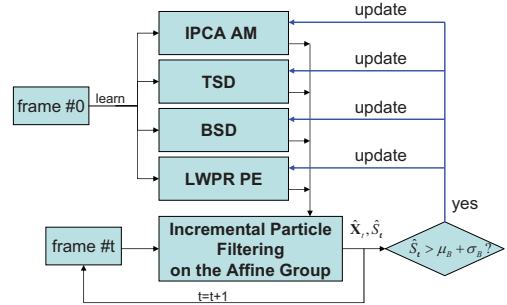


Figure 1. The flowchart of the proposed tracking framework. AM: appearance model, PE: pose estimator, TSD/BSD: target/background similarity distribution.

3. Affine Group

Affine group \mathbf{G} is a differentiable manifold [19], and the tangent space to the identity element \mathbf{I} of this group forms a Lie algebra \mathfrak{g} which has a matrix structure as following:

$$\mathbf{x} = \begin{pmatrix} x_1 & x_3 & x_5 \\ x_2 & x_4 & x_6 \\ 0 & 0 & 0 \end{pmatrix} \quad (3)$$

where $x_i \in \mathbb{R}$. This Lie algebra is equivalent to a 6 dimensional vector space.

The distance on the affine group manifold is measured by the length of geodesic which is the minimum length curve between two points. From the identity element point \mathbf{I} , given any vector $\mathbf{x} \in \mathfrak{g}$, a unique geodesic is determined. The exponential map, $\exp : \mathfrak{g} \rightarrow \mathbf{G}$ maps the vector \mathbf{x} to the point reached by this geodesic. The inverse mapping is given by $\log : \mathbf{G} \rightarrow \mathfrak{g}$. Assume $\mathbf{x} \in \mathfrak{g}$ and $\mathbf{X} \in \mathbf{G}$, computationally, $\exp(\mathbf{x}) = \sum_{i=0}^{\infty} \frac{\mathbf{x}^i}{i!}$, $\log(\mathbf{X}) = \sum_{i=1}^{\infty} \frac{(-1)^{i-1}}{i} (\mathbf{I} - \mathbf{X})^i$.

The geodesic distance between two group elements $\mathbf{X}_1, \mathbf{X}_2 \in \mathbf{G}$ can be defined as [19]:

$$\rho(\mathbf{X}_1, \mathbf{X}_2) = \|\log(\mathbf{X}_1^{-1}\mathbf{X}_2)\| \quad (4)$$

4. Subspace based appearance model

Target's appearance often changes over time due to challenges like illumination or pose changes during tracking. To make appearance model adaptive to these changes, Ross *et al.* [18] present an IPCA (incremental principle component analysis) algorithm to incrementally learn a low dimensional eigenspace representation of target's appearance. Given one or more additional training samples, this incremental learning algorithm can correctly update the eigenbasis as well as the mean.

Appearance Model We use a variant of SIFT descriptor [16] as basic feature. An image patch is divided into 6×6 regions, each of which is represented by a 8-bin gradient orientation histogram covering a full angle range $[0, 2\pi]$. The final descriptor is normalized by L2-norm. Then, IPCA is used to model target's appearance during tracking. Given the learned subspace \mathbf{U}_S , sample mean μ_S and a new observation \mathbf{o}_t specified by state \mathbf{X}_t (that is $\mathbf{o}_t = \mathbf{o}(\mathbf{X}_t)$), the distance between the observation and the appearance subspace is defined as:

$$Dis(\mathbf{o}_t) = \|\mathbf{o}_t - \mu_S - \mathbf{U}_S \mathbf{U}_S^T (\mathbf{o}_t - \mu_S)\|. \quad (5)$$

The observation model in Eq. (1) is naturally formed as:

$$p(\mathbf{o}_t | \mathbf{X}_t) = \exp(-Dis(\mathbf{o}_t)). \quad (6)$$

We define the similarity score (range $0 \sim 1$) between an observation and the appearance subspace as:

$$Sim(\mathbf{o}_t) = 1 - Dis(\mathbf{o}_t) / \sqrt{2} \quad (7)$$

where $\sqrt{2}$ is a normalization factor.

Similarity Distributions Given the initial state \mathbf{X}_0 , the initial TSD $\mathcal{N}_T(\mu_T, \sigma_T)$ is obtained by performing a set of very small perturbations around \mathbf{X}_0 , *e.g.* translating by 0.5 pixel or rotating by 0.5° . The initial BSD $\mathcal{N}_B(\mu_B, \sigma_B)$ can be obtained easily from random background patches. TSD is updated online by the estimated optimal state $\hat{\mathbf{X}}_t$ with a constant learning rate α_l (BSD is updated by background patches similarly):

$$\mu_T(t+1) = (1 - \alpha_l) \cdot \mu_T(t) + \alpha_l \cdot Sim(\hat{\mathbf{o}}_{t+1}) \quad (8)$$

$$\sigma_T(t+1) = ((1 - \alpha_l) \cdot \sigma_T^2(t) + \alpha_l \cdot (Sim(\hat{\mathbf{o}}_{t+1}) - \mu_T(t))^2)^{0.5} \quad (9)$$

where $\hat{\mathbf{o}}_{t+1} = \mathbf{o}(\hat{\mathbf{X}}_{t+1})$.

5. Pose Estimator

Pose estimator is widely used in regression-based tracking which aims to predict target's pose directly from the observation vector by using a regression function. In [19], pose estimator is trained online by rigid regression. This method has several limitations: 1) suffers from high-dimension feature space due to no feature selection for regression; 2) global linear relation between feature space and

the objective space is assumed; 3) the learned regression function is not incrementally updated. To avoid these limitations, we propose a new pose estimator based on LWPR (Local Weighted Projection Regression) [20].

5.1. Introduction to LWPR

LWPR [20] approximates nonlinear function by piecewise linear models. It automatically determines the number of local models K , the *super-plane* parameter β_k in each model, and the *region of validity* (called RF, receptive field), parameterized as a distance metric \mathbf{D}_k in a Gaussian kernel:

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{o} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{o} - \mathbf{c}_k)\right) \quad (10)$$

where \mathbf{o} is an observation, \mathbf{c}_k is the center of RF k , w_k is the activation strength of the observation to RF k . The output \hat{y} of LWPR regressor is the weighted mean of predictions \hat{y}_k produced by each local model:

$$\hat{y} = \left(\sum_{k=1}^K w_k \hat{y}_k\right) / \left(\sum_{k=1}^K w_k\right) \quad (11)$$

Since high-dimension data often has locally low-dimension distribution, LWPR uses PLS (partial least squares) [17] to reduce feature dimension and learn the local regression models. Algorithm 1 shows the weighed PLS regression algorithm. $\mathbf{O}=(\mathbf{o}_1, \dots, \mathbf{o}_n)^T$ are input observations with weighted mean $\bar{\mathbf{o}}$, $\mathbf{y}=(y_1, \dots, y_n)^T$ are responses with weighted mean \bar{y} , $\mathbf{W}=\text{diag}\{w_1, \dots, w_n\}$ are the weights (activation strength) of each data point. PLS recursively uses the maximal correlation direction \mathbf{u}_i between the residual input \mathbf{O}_{res} and output \mathbf{y}_{res} as the projection, then performs single variable regression (obtain β_i) along this projection based on the residuals. When a new sample is available, LWPR incrementally updates local models according to the activation strength and a forgetting factor λ [20].

The distance metric \mathbf{D} (subscript k is omitted) for each model can be learned by stochastic gradient descent in a penalized leave-one-out cross validation cost function [20]:

$$J = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M w_i (y_i - y_{i,-i})^2 + \frac{\gamma}{d} \sum_{i,j=1}^d D_{ij}^2 \quad (12)$$

where M is the number of training samples, d is the data dimension, γ is a penalty factor. See [20] for details.

Given a new training sample (\mathbf{o}, y) , if no RF is activated by more than w_{gen} , a new receptive field will be created with $R = 2$, $\mathbf{c} = \mathbf{o}$, $\mathbf{D} = \mathbf{D}_{def}$, where R is the initial number of projections in PLS, \mathbf{D}_{def} is the initial distance metric in Eq. (10). The number of projections R will be increased if the ratio of the regression error of the last projection to that of the last but two is smaller than a threshold (*e.g.* 0.75).

5.2. Online Learning for Pose Estimator

As shown in Figure 2, given a random sample with an initial state, we aim to use a pose estimator to incrementally adjust the sample's state to its best state according to

Algorithm 1 Weighted Partial Least Squares Regression

- **Learning** with training data: $\mathbf{O}, \mathbf{y}, \mathbf{W}$
 1. Initialize: $\beta_0 = \bar{y}, \mathbf{o}_0 = \bar{\mathbf{o}}, \mathbf{O}_{res} = \mathbf{O} - \bar{\mathbf{o}}^T, \mathbf{y}_{res} = \mathbf{y} - \bar{y}$
 2. **for** $i = 1$ **to** R **do**
 - (a) $\mathbf{u}_i = \mathbf{O}_{res}^T \mathbf{W} \mathbf{y}_{res}$
 - (b) $\beta_i = \mathbf{z}_i^T \mathbf{W} \mathbf{y}_{res} / (\mathbf{z}_i^T \mathbf{W} \mathbf{z}_i)$, where $\mathbf{z}_i = \mathbf{O}_{res} \mathbf{u}_i$.
 - (c) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{z}_i \beta_i, \mathbf{O}_{res} = \mathbf{O}_{res} - \mathbf{z}_i \mathbf{p}_i^T$, where $\mathbf{p}_i = \mathbf{O}_{res}^T \mathbf{W} \mathbf{z}_i / (\mathbf{z}_i^T \mathbf{W} \mathbf{z}_i)$.
 - **Predicting** with querying data \mathbf{o}_q
 1. Initialize: $y_q = \beta_0, \mathbf{o}_q = \mathbf{o}_q - \mathbf{o}_0$.
 2. **for** $i = 1$ **to** R **do**
 - (a) $y_q = y_q + \beta_i s_i$ where $s_i = \mathbf{u}_i^T \mathbf{o}_q$
 - (b) $\mathbf{o}_q = \mathbf{o}_q - s_i \mathbf{p}_i$
 3. Output: y_q .
-

feedback similarity scores. The problem is: given a training set $(\mathbf{O}, \Delta\mathcal{X})$, where $\Delta\mathcal{X} = \{\Delta\mathbf{X}_i\}_{i=1}^M$ is a set of state variations around the correct target state \mathbf{X} , $\mathbf{O} = \{\mathbf{o}_i\}_{i=1}^M$ is a set of corresponding observations, how to learn a good regression function $f : \mathbb{R}^d \rightarrow \mathbf{G}$ mapping from feature space to the affine group? It is difficult to learn such a regression function that can directly map a vector space to a manifold. An alternative method is to learn $f = \exp \circ g : \mathbb{R}^d \rightarrow \mathfrak{g} \rightarrow \mathbf{G}$, where regression g is performed on Lie algebra \mathfrak{g} (vector space), then mapped to affine group \mathbf{G} by the map \exp . This requires that the distance measurement of two elements $\mathbf{X}_1, \mathbf{X}_2$ on \mathbf{G} can be approximated by the distance of their counterparts $\mathbf{x}_1, \mathbf{x}_2$ on \mathfrak{g} . In other words, Eq. (4) can be approximated as [19]:

$$\rho(\mathbf{X}_1, \mathbf{X}_2) = \|\log(\mathbf{X}_1^{-1} \mathbf{X}_2)\| \approx \|\mathbf{x}_2 - \mathbf{x}_1\| \quad (13)$$

where $\mathbf{x}_1 = \log(\mathbf{X}_1), \mathbf{x}_2 = \log(\mathbf{X}_2)$. This approximation is of no problem to our training set, since each state variation $\Delta\mathbf{X}_i$ just lies on a small neighborhood of the identity element \mathbf{I} .

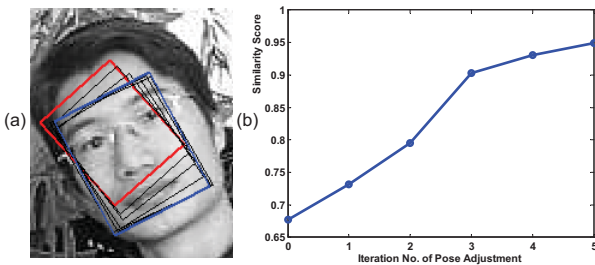


Figure 2. An example of pose adjustment given by our pose estimator (the target is a face). (a) the initial state (red quadrangle), intermediate states (black quadrangles) and final state (blue quadrangle). (b) the similarity score in each iteration.

We use LWPR to learn function $g : \mathbb{R}^d \rightarrow \mathfrak{g}$. Since the objective space \mathfrak{g} is equivalent to a 6-dimensional vector space, we have to learn six regression functions with one for each dimension. Denote $g = lwpr = (lwpr_1, \dots, lwpr_6)$,

Algorithm 2 Incremental learning of pose estimator

- Input:** number of samples M , current target state \mathbf{X} , previous pose estimator $lwpr$
- Process:**
1. draw $\{\Delta\mathbf{x}_i\}_{i=1}^M$ from Uniform distribution \mathcal{U}_g on \mathfrak{g}
 2. compute $\{\mathbf{o}_i = \mathbf{o}(\mathbf{X}\Delta\mathbf{X}_i)\}$, where $\Delta\mathbf{X}_i = \exp(\Delta\mathbf{x}_i)$
 3. **for** $i = 1$ **to** M **do**
 - update $lwpr$ with sample $(\mathbf{o}_i, \Delta\mathbf{x}_i)$
- Output:** $lwpr$
-

Algorithm 3 Self-tuning process of a random sample

- Input:** the state \mathbf{X} , learned pose estimator $lwpr$, maximum number of refining times N_{ref} and the threshold of minimum similarity increment T_{SI}
- Process:**
1. Initialize: $\mathbf{X}_0 = \mathbf{X}, S_0 = Sim(\mathbf{o}(\mathbf{X})), i=0, \Delta S = \infty$.
 2. **while** $i < N_{ref}$ **and** $\Delta S > T_{SI}$ **do**
 - (a) $i=i+1$;
 - (b) estimate pose: $\Delta\mathbf{X}_i = \exp(lwpr(\mathbf{o}(\mathbf{X}_{i-1})))$;
 - (c) compute current state: $\mathbf{X}_i = \mathbf{X}_{i-1} \Delta\mathbf{X}_i^{-1}$;
 - (d) compute current similarity: $S_i = Sim(\mathbf{o}(\mathbf{X}_i))$;
 - (e) compute similarity increment: $\Delta S = S_i - S_{i-1}$;
- Output:** $\mathbf{X} = \mathbf{X}_j |_{S_j = \max\{S_j\}}$
-

given the initial state \mathbf{X}_0 and the number of training samples M_0 at frame #0, the initial $lwpr$ can be learned by Algorithm 2. \mathcal{U}_g is a 6-dimensional uniform distribution on \mathfrak{g} such that: $\forall \mathbf{x} \sim \mathcal{U}_g$, its element (see Eq. 3) ranges are $x_{1:4} \in (-\alpha, \alpha), x_5 \in (-\alpha W_T, \alpha W_T), x_6 \in (-\alpha H_T, \alpha H_T)$, where α is a small constant, (W_T, H_T) is the current size of the tracked target. Note that the elements x_5 and x_6 in \mathbf{x} correspond to x,y translation respectively. Usually we set $M_0=400, \alpha=0.10$, and during tracking we use $\Delta M = 4$ samples per frame to update the pose estimator. Figure 3 shows how to generate the training samples for pose estimator.

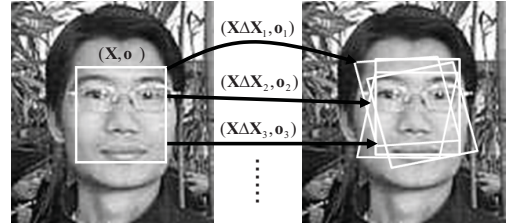


Figure 3. An illustration of how to generate training samples for pose estimator. \mathbf{X} is the current target state with the observation \mathbf{o} , $\Delta\mathbf{X}_i$ is the state variation and \mathbf{o}_i is an observation specified by new state $\mathbf{X}\Delta\mathbf{X}_i$.

The pose tuning process of a random sample with initial state \mathbf{X} is shown in Algorithm 3. Tuning will stop if the maximum number of refining times N_{ref} is reached or the similarity increment between two adjacent iterations is smaller than a threshold T_{SI} . Usually, we set $N_{ref}=4, T_{SI}=0.01$.

6. Incremental Self-tuning Particle Filtering

As in [13] (This paper proposes an optimal importance function derived from an analytical appearance measurement function for sampling particles on the affine group, however, it still works under traditional particle filtering framework), we assume the state transition function is a first-order autoregressive (AR) process evolving on affine group :

$$\mathbf{X}_t = \mathbf{X}_{t-1} \cdot \exp(\mathbf{A}_{t-1}) \exp(d\Omega_t) \quad (14)$$

$$\mathbf{A}_{t-1} = a \log(\mathbf{X}_{t-2}^{-1} \mathbf{X}_{t-1}) \quad (15)$$

where a is the AR process parameter and $d\Omega_t = \sum_{i=1}^6 \omega_{t,i} \mathbf{E}_i$ is a gaussian noise on \mathfrak{g} with $\omega_t = \{\omega_{t,1}, \dots, \omega_{t,6}\}$ sampled from a Gaussian distribution $\mathcal{N}_g(0, \Sigma_g)$. $\Sigma_g = \text{diag}\{\sigma_s^2, \sigma_\alpha^2, \sigma_\beta^2, \sigma_\phi^2, \sigma_x^2, \sigma_y^2\}$, and \mathbf{E}_i is the basis element of \mathfrak{g} chosen as:

$$\begin{aligned} \mathbf{E}_1 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \\ \mathbf{E}_4 &= \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_5 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{E}_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned} \quad (16)$$

The main geometrical transformation modes correspond to $\mathbf{E}_{1:6}$ are respectively scale, aspect ratio, rotation, skew angle, x translation and y translation [13].

Traditional particle filtering uses a large number of particles to find the best estimate of target state, especially when the state dimension is high. By using a pose estimator and the modeled similarity distributions of target/background patches, we propose an incremental self-tuning particle filtering (ISPF) framework which makes sparse sampling possible. Algorithm 4 shows the main steps. The initial number of particles ΔN_0 for each frame is usually 5 to 10 percent of the maximum number of particles N_{total} . Initial particles are propagated by the state transition function, and then tuned to their best states by the learned LWPR pose estimator. If the maximum similarity score of tuned particles $\tilde{\mathcal{X}}_t^k$ ($k=0,1,\dots$) does not satisfy TSD (*e.g.* $S_{max}^k < \mu_T - \sigma_T$), another ΔN_{k+1} particles $\Delta \mathcal{X}_t^{k+1}$ will be added by resampling from $\tilde{\mathcal{X}}_t^k$. These newly added particles are propagated by a Gaussian noise model (see step 5.(d)), then tuned to their best states by the learned LWPR pose estimator. The resampling process in step 5(c) means drawing ΔN_k particles (with replacement) from the discrete distribution $(\tilde{\mathcal{X}}_t^{k-1}, \tilde{W}_t^{k-1})$, where $\tilde{W}_t^{k-1} = \{\tilde{w}_{t,i}^{k-1}\}_{i=1}^n$ is the normalized weights of the particles in $\tilde{\mathcal{X}}_t^{k-1}$, and $\tilde{w}_{t,i}^{k-1}$ is determined by the observation model Eq. (6). The resampling process makes the incremental particles concentrate on the tuned particles with large weights. Since incremental particles are propagated and tuned at each iteration, the ISPF framework can intelligently find the optimal target position in a step-by-step way, which is beneficial for searching in large search space (*e.g.* caused by fast motion). If the similarity score of the final estimated state (see Step 6) is not

too bad (*e.g.* larger than $\mu_B + \sigma_B$), all online models (the IPCA AM, LWPR PE, TSD and BSD) are updated.

Algorithm 4 Incremental Self-tuning Particle Filtering

Input: maximum number of particles N_{total} , initial number of particles ΔN_0 , increment factor α_{inc} , previous particle states $\mathcal{X}_{t-1} = \{\mathbf{X}_{t-1,i}\}_{i=1}^{\Delta N_0}$, $\hat{\mathbf{X}}_{t-2}$ and $\hat{\mathbf{X}}_{t-1}$

Process:

1. Calculate velocity \mathbf{A}_{t-1} by Eq. (15) with $\hat{\mathbf{X}}_{t-2}$ and $\hat{\mathbf{X}}_{t-1}$.
2. Propagate \mathcal{X}_{t-1} to new states $\mathcal{X}_t^0 = \{\mathbf{X}_{t,i}^0\}_{i=1}^{\Delta N_0}$ by state transition Eq. (14).
3. Tune \mathcal{X}_t^0 to their best states $\tilde{\mathcal{X}}_t^0 = \{\tilde{\mathbf{X}}_{t,i}^0\}_{i=1}^{\Delta N_0}$ by Algorithm 3 and compute $S_{max}^0 = \max\{Sim(\mathbf{o}(\tilde{\mathbf{X}}_{t,i}^0))\}_{i=1}^{\Delta N_0}$.
4. $n = \Delta N_0$, $k = 0$.
5. **while** $S_{max}^k < \mu_T - \sigma_T$ **and** $n < N_{total}$ **do**
 - (a) $k = k + 1$;
 - (b) $\Delta N_k = \min\{\alpha_{inc} \Delta N_{k-1}, N_{total} - n\}$;
 - (c) Generate $\Delta \mathcal{X}_t^k = \{\mathbf{X}_{t,j}^k\}_{j=1}^{\Delta N_k}$ by resampling from $\tilde{\mathcal{X}}_t^{k-1}$;
 - (d) For all j , $\mathbf{X}_{t,j}^k = \mathbf{X}_{t,j}^k \exp(d\Omega_{t,j}^k)$ with $d\Omega_{t,j}^k \sim \mathcal{N}_g$;
 - (e) Tune $\Delta \mathcal{X}_t^k$ to their best states $\Delta \tilde{\mathcal{X}}_t^k$ by Algorithm 3;
 - (f) $n = n + \Delta N_k$;
 - (g) $\tilde{\mathcal{X}}_t^k = \{\tilde{\mathcal{X}}_t^{k-1}, \Delta \tilde{\mathcal{X}}_t^k\} = \{\tilde{\mathbf{X}}_{t,i}^k\}_{i=1}^n$;
 - (h) $S_{max}^k = \max\{Sim(\mathbf{o}(\tilde{\mathbf{X}}_{t,i}^k))\}_{i=1}^n$;
6. $\hat{\mathbf{X}}_t = \tilde{\mathbf{X}}_{t,i^*}^k |_{Sim(\mathbf{o}(\tilde{\mathbf{X}}_{t,i^*}^k)) = S_{max}^k}$, $\hat{S}_t = S_{max}^k$
7. **if** $\hat{S}_t > \mu_B + \sigma_B$ **then**
 - update IPCA AM, LWPR PE, TSD and BSD with $\hat{\mathbf{X}}_t$
8. Generate $\mathcal{X}_t = \{\mathbf{X}_{t,i}\}_{i=1}^{\Delta N_0}$ for the next frame by resampling from $\tilde{\mathcal{X}}_t^k$.

Output: $\hat{\mathbf{X}}_t$

7. Experimental Results

Five sequences are used to test the robustness and efficiency of our tracking algorithm. They are the "Trellis" (Figure 4), "Sylvester" (Figure 7 1st column), "Dudek" (Figure 8), "Cubic" (Figure 7 2nd column) and "Head-shoulder" (Figure 7 3rd column) sequences, with the first three from Ross's datasets [18]. Some important parameters are set as follows: maximum number of particles $N_{total} = 100$, number of initial particles $\Delta N_0 = 10$, particle increment factor $\alpha_{inc} = 1.5$, gaussian noise covariance matrix $\Sigma_g = \text{diag}\{0.03^2, 0.001^2, 0.03^2, 0.001^2, 5^2, 5^2\}$, initial number of projections in PLS $R = 2$, initial RF size $\mathbf{D}_{def} = 2.5 \cdot \text{diag}\{1, \dots, 1\}$, RF born threshold $w_{gen} = 0.25$. Image patch is usually resized to 36×36 . The MATLAB implementation of our method can run at about 2 ~ 5 frames/sec on a PC with a P4 2.4 GHz CPU.

As comparison, the experimental results of another two state-of-the-art tracking algorithms are also given. One is proposed by Tuzel *et al.* [19], using a rigid regression (RR) based pose estimator to track target. We call it RR method. The other is the IPCA based particle filtering method proposed by Ross *et al.* [18], we call it IPCAPF.

Goodness-of-Fit of PLS Since PLS is the basic regression model in the LWPR pose estimator, we study the re-

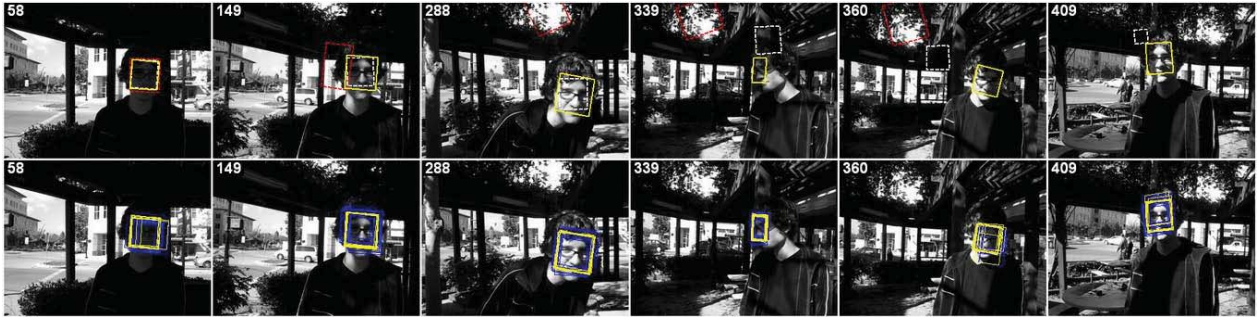


Figure 4. First row: the tracking results of the three methods: our ISPF (yellow solid), IPCAPF (white dot) and RR (red dot) in the "Trellis" sequence. Second row: randomly sampled particles (blue) and particles (yellow) provided by ISPF.

gression ability of PLS on our tracking data and plot some curves of "percentage of variance explained" vs. "number of projections" in Figure 5. The used training data is the initial pose training sets at the first frame of sequence "Trellis", "Sylvester" and "Dudek". The term "percentage of variance explained" [12] indicates how much better a regression function predicts the dependent variable than just using the mean value of the dependent variable. It is computed as "1-(variance using predicted value/ variance using mean)". As shown in Figure 5, with only 2 projections, more than 90 percent of variance can be explained by PLS regression. However, target's appearance may often change during tracking, the real regression function between the observation and state space may be complicated and changing. Fortunately, LWPR can automatically add new RFs (linear models) and new projections if necessary. In the following experiments, we also find that during tracking, usually no more than 4 RFs and in each RF usually no more than 4 projections are used for each regression function. So the LWPR pose estimator actually works every efficiently.

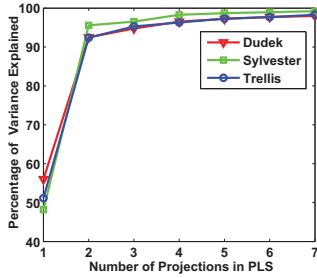


Figure 5. Regression ability of PLS on our tracking data

Robustness Evaluation Four sequences are used to test the robustness of the three methods (IPCAPF, RR, and ISPF) to illumination/pose changes and fast motion. They are "Trellis", "Sylvester", "Cubic" and "Head-shoulder" sequence respectively.

The tracked target in the "Trellis" sequence is a face of a man who is walking underneath a trellis covered by vines. As shown in Figure 4, the lighting conditions vary drastically and cast shadows often change the face significantly. The target also experiences sever pose changes. The RR

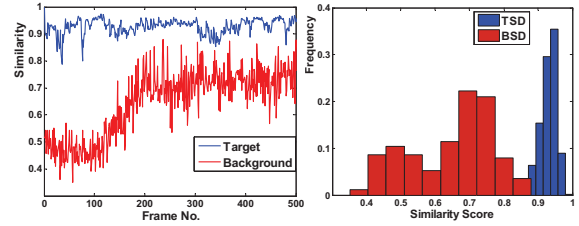


Figure 6. Left: Similarity curves of target patches (specified by estimated state \hat{X}_t) and background patches (randomly sampled from background regions) in the "Trellis" sequence. Right: corresponding Target/Background Similarity Distributions.

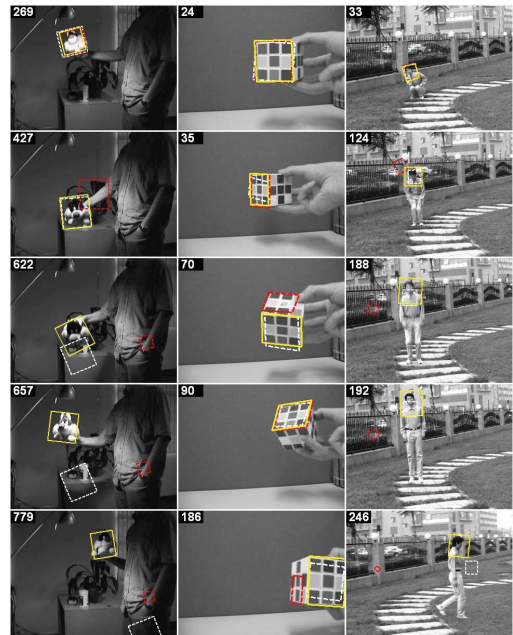


Figure 7. Tracking results of the three methods: our ISPF (yellow solid), IPCAPF (white dashed) and RR (red dashed) in the "Sylvester" (1st Column), "Cubic" (2nd Column) and "Head-shoulder" (3rd Column) sequence.

tracker is trapped in the cluttered background at about frame #149 due to large illumination change, and the IPCAPF tracker fails to track the target at about frame #339 due to severe pose change. However, our ISPF method tracks the target very accurately and robustly throughout this sequence. Figure 6 shows the TSD/BSD over the total "Trellis" se-

quence. It is reasonable for us to use a time varying Gaussian to model TSD/BSD, though the BSD experiences two major Gaussians in two different periods of time. As shown in Figure 4 2nd row, most random particles can be correctly tuned to target’s real position by the learned LWPR pose estimator. The number of particles used by the ISPF tracker in each frame is shown in Figure 10. The mean number per frame is only 12.4. While IPCAPF uses 600 particles/frame. Therefore, our ISPF tracker works every efficiently.

The tracked target in the "Sylvester" sequence is a doll undergoing great illumination changes and severe pose changes, shown in Figure 7 1st column. At about frame #427, the RR method fails to track the target because of large motion and pose changes, and at about frame #622, the IPCAPF tracker totally loses the target due to severe pose changes. However, our ISPF tracker tracks the target very robustly. In the "Cubic" sequence, shown in the Figure 7 2nd column, the target is a cubic facet undergoing great pose changes (#35, #90) and partial occlusion (#186). The tracking results of the RR method are unstable, for the RR tracker sometimes jumps from one facet to another (#70, #186). The IPCAPF method shows certain robustness in this sequence, but the tracking accuracy is not as good as ours (#35, #186). In the "Head-shoulder" sequence (Figure 7 3rd column), the target is the head-shoulder part of a man who jumps frequently causing fast motion. The RR tracker loses target at about frame #124 because of cluttered background, and the IPCAPF tracker begins to lose target at about frame #188 due to fast motion (caused by jumping). It is totally trapped by cluttered background at about frame #246 because of large pose changes. However, our ISPF tracker shows good robustness to these challenges. The mean numbers of particles used per frame in the "Sylvester", "Cubic" and "Head-shoulder" sequences are 12.8, 5.9 and 13.0 respectively. The number of particles used in each frame is shown in Figure 10. Note that the IPCAPF tracker always uses 600 particles per frame.

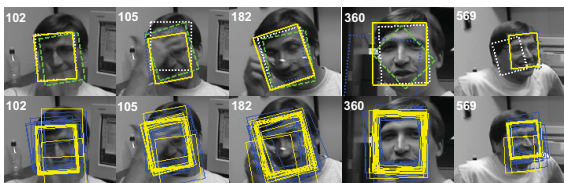


Figure 8. First row: tracking results of the four methods: ISPF (yellow solid), IPCAPF (white dotted), LWPR (blue dotted), and RR (green dashed) in the "Dudek" sequence. Second row: randomly sampled particles (blue solid) and tuned particles (yellow solid) provided by our ISPF method. (images are cropped for better show).

Quantitative Evaluation To quantitatively evaluate our algorithm, the "Dudek" sequence from Ross’s dataset (initially created by Jepson *et al.* [11]) is used. There are seven manually labeled ground-truth feature points in each frame.

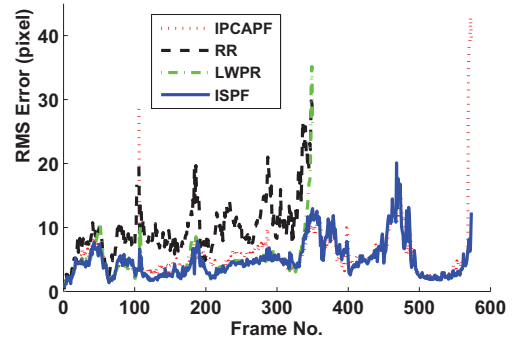


Figure 9. RMS curves of four tracking algorithms in the "Dudek" sequence

Table 1. Comparison of some statistics

	IPCAPF	RR	LWPR	ISPF
\overline{RMS}_{350}	4.95	9.89	4.69	4.22
\overline{RMS}_{total}	5.6	NULL	NULL	4.95
$\overline{N}_{particles}$	4000	NULL	NULL	14.9

Considering there existing large motions and pose changes simultaneously sometimes in this sequence, we set the maximum number of refining times N_{ref} of our ISPF method to 8. To make a quantitative comparison between the RR pose estimator [19] and our LWPR pose estimator, we implement a tracker which only uses the LWPR pose estimator to track target (called LWPR tracker). During tracking, seven validation feature points, corresponding to the benchmark feature points, are obtained according to the target’s affine parameters at each frame. The RMS (root mean square) errors of the four trackers (IPCAPF, RR, LWPR, ISPF) between the estimated locations of the feature points and the ground truth are shown in Figure 9. The RMS errors of the LWPR and RR tracker after frame #350 are not shown because they both lose the target at that time due to large target motions and pose changes. Table 1 shows some statistics of the four trackers. \overline{RMS}_{350} is the mean RMS error in the first 350 frames, \overline{RMS}_{total} is the mean RMS error over total sequence and $\overline{N}_{particles}$ is the mean number of particles used per frame. It can be seen that our ISPF method achieves the best tracking accuracy, the RR method gives the worst tracking accuracy. LWPR tracker works much better than the RR/IPCAPF tracker before frame #350. However, the robustness of both the LWPR and RR tracker is inferior to that of the ISPF/IPCAPF tracker.

The representative tracking results over some frames are shown in Figure 8 1st row. The RR tracker is obviously influenced by a small partial occlusion (see #102), while LWPR tracker shows much better robustness. This is because the used PLS regression selects some best projection directions for regression, not all low-level features. The second row of Figure 8 shows the randomly sampled particles and tuned particles in some frames provided by our ISPF method. Most random particles are tuned to the correct target state, so our ISPF tracker always maintains a set of very

effective particles.

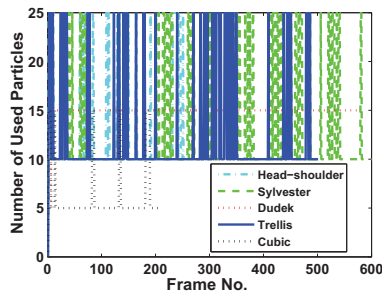


Figure 10. Number of particles used in every frame by the ISPF method in the five test sequences.

Discussion From experimental results we can obtain some useful findings: 1) The LWPR pose estimator can tremendously reduce feature dimension for regression, and it also achieves much better tracking accuracy than the RR (rigid regression) pose estimator. 2) Pure regression based tracking methods (both the LWPR method and RR method) are often not robust because they often provide unexpected results when target's appearance changes greatly due to large pose changes, fast motion or clutter background. 3) With the help of the LWPR pose estimator and an online subspace based similarity measurement, our ISPF framework can achieve better robustness and accuracy than the pure stochastic method (traditional particle filtering) and pure deterministic method (regression based tracking) while using only a very small number of particles.

8. Conclusions

We have proposed an incremental self-tuning particle filtering (ISPF) for visual tracking. Its success lies in the factor that individual modules form an organic whole to enhance each other. To find a short way to reach the optimal state in high dimensional state space, incremental particle filtering (PF) process utilizes a pose estimator to guide random samples to move towards the correct directions; the pose estimator gathers good training data based on the reliable state estimation provided by the PF process. The two parts are well combined by a reliable subspace based similarity feedback, which supervises both the pose tuning process and the Bayesian state optimization process. This sophisticated but well-organized PF process can find the optimal state with a very small number of particles almost regardless of the dimension of state space. Experimental results show that ISPF is more robust and accurate than two state-of-the-art tracking frameworks (pure stochastic or deterministic method) while works efficiently.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant No. 60736018, 60723005), National Hi-Tech Research and Development Program of China (2009AA01Z318), National Science Foundation (60605014, 60875021).

References

- [1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Journal of IEEE Trans. on Signal Processing*, 50(2):174–188, 2002.
- [2] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271, 2007.
- [3] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. *Proc. of CVPR'09*, 2009.
- [4] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004.
- [5] Y. Bar-Shalom and T. Fortmann. Tracking and data association. Academic Press, New York, 1988.
- [6] J. Carpenter, P. Clifford, and P. Fearnhead. Improved particle filter for nonlinear problems. *Proc. of Inst. Elect. Eng., Radar, Sonar, Navig*, 1999.
- [7] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564–577, 2003.
- [8] A. Doucet, J. F. G. de Freitas, and N. J. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.
- [9] G. Hager and P. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. *Proc. of CVPR'96*, 1996.
- [10] M. Isard and A. Blake. Condensation: conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.
- [11] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *PAMI*, 25(10):1296–1311, 2003.
- [12] J. T. Kent. Information gain and a general measure of correlation. *Biometrika*, 70(1):163–173, 1983.
- [13] J. Kwon, K. M. Lee, and F. C. Park. Visual tracking via geometric particle filtering on the affine group with optimal importance functions. *Proc. of CVPR'09*, 2009.
- [14] I. Leichter, M. Lindenbaum, and E. Rivlin. Tracking by affine kernel transformations using color and boundary cues. *PAMI*, 31(1):164–171, 2009.
- [15] P. Li, T. Zhang, and B. Ma. Unscented kalman filter for visual curve tracking. *Image and Vision Computing*, 22(21):157–164, 2004.
- [16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
- [17] R. Rosipal and N. Krämer. Overview and recent advances in partial least squares. *Subspace, Latent Structure and Feature Selection*, pages 34–51, 2006.
- [18] D. A. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1):125–141, 2008.
- [19] O. Tuzel, F. Porikli, and P. Meer. Learning on lie groups for invariant detection and tracking. *Proc. of CVPR'08*, 2008.
- [20] S. Vijayakumar, A. D. Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.
- [21] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu. Sequential particle swarm optimization for visual tracking. *Proc. of CVPR'08*, 2008.
- [22] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *PAMI*, 31(4):677–692, 2009.