Classification-Based Likelihood Functions for Bayesian Tracking

Chunhua Shen[1,2], Hongdong Li[1,2], Michael J. Brooks[3]
[1]ViSTA program, Canberra Research Laboratory, National ICT Australia
[2]Research School of Information Science & Engineering, Australian National University
[3]School of Computer Science, University of Adelaide

Abstract

The success of any Bayesian particle filtering based tracker relies heavily on the ability of the likelihood function to discriminate between the state that fits the image well and those that do not. This paper describes a general framework for learning probabilistic models of objects for exploiting these models for tracking objects in image sequences. We use a discriminative classifier to learn models of how they appear in images. In particular, we use a support vector machine (SVM) for training, which is able to extract useful non-linear information, and thus represent more complex characteristics of the tracked object and background. This is a particular advantage when tracking deformable objects and where appearance changes due to the unstable illumination and pose occur.

A by-product of the SVM training procedure is the classification function, with which the tracking problem is cast into a binary classification problem. An object detector directly using the classification function is then available. To make the tracker robust, an object detector that directly uses the classification function is combined into the tracker for object verification. This provides the capability for automatic initialisation and recovery from momentary tracking failures. We demonstrate improved robustness in image sequences.

1 Introduction

Robust appearance-based tracking of objects in image sequences has been extensively researched in recent years due to its wide range of potential applications, including, for example, video surveillance and human-machine interaction. One of the key problems affecting the performance of visual trackers has been the lack of a suitable appearance model. For a particle filtering based tracking algorithm, the equivalent problem is to find a sufficiently robust likelihood function. The likelihood function $\Pr(\boldsymbol{z}|\boldsymbol{x})$ computes a measure of how well a state hypothesis $\boldsymbol{x}$ fits the image observations. The first particle filtering tracker CONDENSATION uses bottom-up likelihood functions,[1] where only edge features is adopted [1]. Most of the other top-down approaches use a single static template image to construct a target representation based on density models. The reference is usually extracted from the first frame by hand and kept fixed. It is difficult to update this type of target model [2; 3; 4; 5; 6], and the target representation's fragility usually breaks these trackers over a long image sequence. A naive adaptation method replaces the template by the tracking results from the previous frames, but this can easily undo the tracker. Considerable effort has been expended to ease these difficulties. We believe that the key to finding a solution is to find the right representation. In order to accommodate appearance changes, the representation model should be learned from as many training examples as possible.

Fundamentally two methods, namely off-line and on-line learning, can be used for the training procedure. On-line learning means constantly updating the representation model during the course of tracking. Lim et al. [7] propose an incremental eigenvector update strategy to adapt the target representation model. A linear probabilistic principal component analysis model is used. The main disadvantage of the eigen-model is that it is not generic and is usually only suitable for characterising texture-rich objects. Jepson et al. [8] update a wavelet model with the expectation maximisation (EM) algorithm. A classification function is progressively learned using AdaBoost for visual detection and tracking in [9] and [10] respectively. Han and Davis [11] adopt pixel-wise Gaussian mixture models to represent the target model and sequentially update them. To date, however, less work has been reported on how to elegantly update *region-wise density* models in tracking. Vermaak et al. [6] use Monte Carlo EM to update a colour-histogram appearance model. Motion information is used to determine when

[1]With a "bottom-up" likelihood model, no pre-set reference model is needed for calculating the likelihood probability. In contrast, a "top-down" likelihood model defines a likelihood probability by measuring the distance between a hypotheses and the reference model.

to update the model in their method. Nevertheless it depends on the motion information that is not always reliably available.

The idea of off-line learning an appearance model for tracking is not new. Given an amount of training examples of the tracked object, a representation model is obtained by using clustering or subspace learning. For the task of finding multiple people, Ramanan and Forsyth [12] learn an appearance model by clustering the candidate feature vectors using mean-shift for each body segment. Black and Jepson [13] build an eigen-space representation using principle component analysis (PCA) for tracking rigid and articulated objects.

In contrast, classification[2] is a powerful bottom-up procedure: It is trained off-line and works on-line. Due to the training being typically built on very large amounts of training data, its performance is fairly promising even without on-line updating of the classifier/detector. It is often beneficial to combine a bottom-up process into a top-down tracker. The way the two components are combined is application dependent and plays a critically decisive role in the robustness and efficiency of the tracker. Inspired by image classification tasks with colour density features (*e.g.*, [14]) and real-time detection [15], we learn off-line a representation model from multiple training data. By considering tracking as a binary classification problem, a discriminative classification rule is learned to distinguish between the tracked object and background patterns. In this way a robust object representation model is obtained.

This proposal provides a basis for considering the design of enhanced particle filtering based trackers. Unlike the conventional principle of comparing the feature content of candidate regions to a reference model, we compute a classification probability with the learned classifier and embed it within a particle filtering framework. In this way, we obtain a hybrid generative/discriminative model which, as we will show, is more promising than the standard generative particle filters.

A by-product of the training is the classification function, with which the tracking problem is cast into a binary classification problem. An object detector directly using the classification function is then available. Combining a detector into the tracker makes the tracker more robust and provides the capabilities of automatic initialisation and recovery from momentary tracking failures.

We discuss the proposed likelihood functions in Section 2. Experiments on real video sequences are presented in Section 3, followed by concluding remarks.

[2]Object detection is typically a classification problem.

2 Method

To make this paper self-contained, we briefly review the technique of Bayesian particle filtering and support vector machine. We then summarise the proposed algorithm and show how the proposed likelihood functions lead to better tracking performance.

2.1 Particle Filtering

We work in a Bayesian framework using a generative approach. The goal is to estimate the posterior probability distribution $\Pr(\boldsymbol{x}_t|\boldsymbol{z}_{1:t})$ for the state $\boldsymbol{x}_t$ of the tracked target at time t given a sequence of image observations $\boldsymbol{z}_{1:t} = (\boldsymbol{z}_1, \dots \boldsymbol{z}_t)$. With the first-order Markovian assumptions that the state at time t is only dependent on the previous state while the observation is only dependent on the current state, a recursive Bayes formula can be derived and used for inference: $\Pr(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) \propto \Pr(\boldsymbol{z}_t|\boldsymbol{x}_t)\Pr(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1})$ where the prior is the previous posterior propagated across the temporal axis,

$$\Pr(\boldsymbol{x}_t|\boldsymbol{z}_{1:t-1}) = \int \Pr(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})\Pr(\boldsymbol{x}_{t-1}|z_{1:t-1})\,d\boldsymbol{x}_{t-1}.$$

When the dynamic and observation models are non-linear and/or non-Gaussian, the above posterior cannot be analytically computed and one has to resort to numerical approximations. In visual tracking problems, the dynamic model can be approximated by a linear model while the observation model is usually highly non-linear.

To handle the multi-modality of the posterior distribution, non-parametric approximate methods are used that represent distributions by a set of N samples or particles with associated normalised weights $\{(\boldsymbol{x}_t^i, w_t^i)\}_{i=1}^N$. This is the essential idea of particle filtering. The posterior is then formulated as $\Pr(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^N w_t^i \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^i)$, where $\delta(\cdot)$ is the Dirac function. The above integral is now tractable with this numerical approximation.

Assume we can sample the particles from an importance density $\Pr_f(\cdot)$, *i.e.*, $\boldsymbol{x}_t^i \sim \Pr_f(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^i, \boldsymbol{z}_{1:t})$, then each particle's weight is set to

$$w_t^i \propto \frac{\Pr(\boldsymbol{z}_t|\boldsymbol{x}_t^i)\Pr(\boldsymbol{x}_t^i|\boldsymbol{x}_{t-1}^i)}{\Pr_f|(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^i, \boldsymbol{z}_{1:t})}, \; i = 1, \dots N. \quad (1)$$

Before or after the importance sampling step, a selective re-sampling step is adopted to ensure the efficiency of the particles' evolution. Based on the discrete approximation of the posterior, different estimates of the state at time t can be devised. In our experiment, we use the Monte Carlo approximation of the expectation (minimum mean square error estimate) $\tilde{\boldsymbol{x}}_t = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_t \approx \mathrm{E}(\boldsymbol{x}_t|\boldsymbol{z}_{0:t})$ as the tracker output at time t.

- **Initialisation**: $t = 1$. Sample N particles $\{(\boldsymbol{x}_{t-1}^i, w_{t-1}^i)\}_{i=1}^N$ from the prior.
- **Re-sampling**: Re-sample to obtain N replacement particles $\{(\boldsymbol{x}_{t-1}^i, \frac{1}{N})\}_{i=1}^N$, according to the weights w_{t-1}^i.
- **Importance sampling**: For $n = 1, \ldots, N$, sample N particles $\boldsymbol{x}_t^i$ from the importance proposal $\Pr_f(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}^i, \boldsymbol{z}_{1:t})$, and evaluate the weights according to Equation (1). Normalise the weights.
- $t = t + 1$. Go to the **Re-sampling** step to process the next frame.

Figure 1. The standard particle filtering algorithm.

To summarise, we present the complete algorithm for a particle filter in Figure 1.

As mentioned, the dynamics model ($\Pr(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$) and the likelihood function ($\Pr(\boldsymbol{z}_t|\boldsymbol{x}_t)$) determine a particle filter's performance. We focus on learning a better likelihood function for visual tracking. In theory many classifiers can be used for this training purpose. In this work, we train a support vector machine (SVM) to distinguish the foreground and background. We show that a carefully selected kernel based non-linear SVM is powerful, yet compared with a conventional fixed-template approach, no computation overhead is introduced. This is particularly important for real-time application. Next we recall some important concepts of SVM.

2.2 Support Vector Machine

Support vector classifiers are developed from the theory of Structural Risk Minimisation [16]. SVMs implicitly map the data into a dot product space $\mathcal{F}$ via a (usually non-linear) map $\Phi(\cdot)$. Although typically $\mathcal{F}$ is high-dimensional, with the kernel trick, it is not necessary to explicitly work in that space. An SVM computes a hyperplane in the kernel space which separates the data in $\mathcal{F}$ by a large margin.

An SVM constructs a symmetric and positive definite kernel matrix (Gram matrix) which represents the similarities between all training datum points. Given M training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^M$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \{+1, -1\}$, the primal optimisation problem of 1-norm soft margin SVM is written as:

$$\begin{aligned} \min_{\boldsymbol{\xi}, \boldsymbol{w}, b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum\nolimits_{i=1}^M \xi_i, \\ \text{s.t.} \quad & y_i(\boldsymbol{w}^\top \Phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i, \xi_i \geq 0, \ \forall_i. \end{aligned} \tag{2}$$

Here $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^M$ is the slack variable set and the regularisation parameter C determines the trade-off between the SVM's generalisation capability and training error; b is the offset. In order to use kernel functions, (2) is usually converted into its dual:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum\nolimits_{i=1}^M \alpha_i - \frac{1}{2} \sum\nolimits_{i,j=1}^M \alpha_i \alpha_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j), \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \forall_i, \quad \sum\nolimits_{i=1}^M \alpha_i y_i = 0. \end{aligned} \tag{3}$$

Here $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle$ is the kernel matrix. The above optimisation problem can be efficiently solved by quadratic programming. The decision rule is then given by $\operatorname{sign}(f(\boldsymbol{x}))$ with

$$f(\boldsymbol{x}) = \sum\nolimits_{i=1}^M y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{4}$$

The output, $f(\boldsymbol{x})$, of an SVM is an un-calibrated value, not a probability of a class given an input. A posterior probability

$$\Pr(y = +1|\boldsymbol{x}) = \frac{1}{1 + \exp(Af(\boldsymbol{x}) + B)} \tag{5}$$

is estimated by fitting a sigmoid to a set of SVM-distances using the method proposed by Platt [17]. The parameters A, B in (5) are obtained using maximum likelihood estimation from a training set.

2.3 Probability Product Kernel

Measuring the similarity between image patches is of central importance in computer vision. In SVMs, the kernel $k(\cdot, \cdot)$ plays this role. Most commonly used kernels such as Gaussian and polynomial kernels are not defined on the space of probability distributions. Recently various probabilistic kernels have been introduced, including the Fisher kernel [18], TOP [19], Kullback-Leibler kernel [20] and probability product kernels (PPK) [21], to combine generative models into discriminative classifiers. A probabilistic kernel is defined by first fitting a probabilistic model $\mathbf{p}(\boldsymbol{x}_i)$ to each training vector $\boldsymbol{x}_i$. The kernel is then a measure of similarity between probability distributions. PPK is an example [21], with kernel given by

$$k_\rho^\star(\mathbf{q}(\boldsymbol{x}), \mathbf{p}(\boldsymbol{x})) = \int_{\mathcal{X}} \mathbf{q}(\boldsymbol{x})^\rho \mathbf{p}(\boldsymbol{x})^\rho \, d\boldsymbol{x} \tag{6}$$

where ρ is a constant. When $\rho = \frac{1}{2}$, PPK reduces to a special case, termed the Bhattacharyya kernel:

$$k_{\frac{1}{2}}^\star(\mathbf{q}(\boldsymbol{x}), \mathbf{p}(\boldsymbol{x})) = \int_{\mathcal{X}} \sqrt{\mathbf{q}(\boldsymbol{x})} \sqrt{\mathbf{p}(\boldsymbol{x})} \, d\boldsymbol{x}. \tag{7}$$

In the case of discrete histograms, *i.e.*, $\mathbf{q}(\boldsymbol{x}) = [q_1 \cdots q_m]^\top$ and $\mathbf{p}(\boldsymbol{x}) = [p_1 \cdots p_m]^\top$, (7) becomes

$$k_{\frac{1}{2}}^\star(\mathbf{q}(\boldsymbol{x}), \mathbf{p}(\boldsymbol{x})) = \sqrt{\mathbf{q}(\boldsymbol{x})}^\top \sqrt{\mathbf{p}(\boldsymbol{x})} = \sum_{u=1}^m \sqrt{q_u p_u}. \tag{8}$$

When $\rho = 1$, $k_1^\star(\cdot,\cdot)$ computes the expectation of one distribution over the other, and hence is termed the expected likelihood kernel [21]. In [22] its corresponding statistical affinity is used as similarity measurement for tracking. The Bhattacharyya kernel is adopted in this work because it has been empirically shown, at least for image classification, that the generalisation capability of expected likelihood kernel $k_1^\star(\cdot,\cdot)$ is weaker than the Bhattacharyya kernel. Meanwhile, non-linear probabilistic kernels including Bhattacharyya kernel, Kullback-Leibler kernel, Rényi kernel, *etc*., perform similarly [23]. Moreover, the Bhattacharyya kernel is simple and has no kernel parameter to tune.

The PPK has an interesting characteristic that the mapping function $\Phi(\cdot)$ is explicitly known: $\Phi(\mathbf{q}(\boldsymbol{x})) = \mathbf{q}(\boldsymbol{x})^\rho$. This is equivalent to directly setting $\boldsymbol{x} = \mathbf{q}(\boldsymbol{x})^\rho$ and the kernel $k_\rho^\star(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$. Consequently for discrete PPK based SVMs, in the test phase the computational complexity is independent of the number of support vectors. This is easily verified. For example, for histogram based image classification like [14], given a test image $\boldsymbol{x}$, the histogram vector $\mathbf{p}(\boldsymbol{x})$ is immediately available. In fact we can interpret discrete PPK based SVMs as *linear* SVMs in which the input vectors are $\mathbf{q}(\boldsymbol{x}_i)^\rho$—the features *non-linearly*[3] extracted from raw images $\boldsymbol{x}_i$. By contrast, [24] applied the reduced set method to reduce the number of support vectors for speeding up the classification phase. Applications which favour fast computation in the test phase, such as large scale image retrieval, might also benefit from this property of the discrete PPK.

2.4 The State Space and Dynamics and Likelihood Function

We want to track a patch of interest in the image plane. The patch is a rectangle in shape. In this work, we estimate the location of the target $\boldsymbol{d} = (x, y)$ in the image coordinate system and the scale s. We define the state at time t as $\boldsymbol{x}_t = (\boldsymbol{d}_t, \boldsymbol{d}_{t-1}, s_t, s_{t-1})$. A second-order auto-regressive dynamics model is used here: $\boldsymbol{x}_t = S_1\boldsymbol{x}_{t-1} + S_2\boldsymbol{x}_{t-2} + v_t, v_t \sim \mathcal{N}(0, \Sigma)$, where v_t is a multivariate normal distribution and the matrices S_1, S_2 define the deterministic component, and Σ the stochastic component. These matrices could be learned from a set of manually-labelled representative sequences. A fixed model is used here, which is composed of three independent constant velocity dynamics on x_t, y_t and s_t.

The second critical factor in a particle filter is the likelihood model. Typically particle filtering based trackers define a likelihood function by converting the distance between the current hypothesis $\boldsymbol{x}^i$ and the template $\boldsymbol{x}^*$ (under certain metric $D(\cdot,\cdot)$) to a weighted exponential probability $\Pr(\boldsymbol{z}^i|\boldsymbol{x}^i) \propto \exp(-\lambda D(\boldsymbol{x}^i, \boldsymbol{x}^*))$ The parameter λ must be manually tuned. See, for example, [5].

[3]When $\rho = 1$, it is linear.

As mentioned earlier, we learn likelihood functions directly from both positive and negative training data using SVMs. Specifically, the particle filter's likelihood function is defined by the classifier's posterior probability learned in Equation (5), That means,

$$\Pr(\boldsymbol{z}|\boldsymbol{x}) \propto \Pr(y = +1|\boldsymbol{x}). \qquad (9)$$

This likelihood function is then plugged into the Bayesian tracking algorithm described in Figure 1.

3 Experiments

It is often beneficial to incorporate a bottom-up classifier/detector into a tracker. In the tracking experiment, an SVM verification is performed after the particle filtering tracker finds a candidate location. If the SVM score is positive, we accept the result and switch to the next frame. Otherwise a negative score triggers a search in a wider region, using either a brute-force detector or AdaBoost detectors.

To make the representation more robust, we train a SVM model for the specific person we want to track. In the SVM training, we use only RGB colour channels as the features. All the negative training data are extracted from the background. The image size is 128×96. The size of the tracked object is about 18×18. In the first experiment, the tracked face moves quickly. Hence the displacement between neighbouring frames is large. The tracker is started manually. Armed with a detector/localiser, the proposed algorithm tracks the whole sequence successfully. Figure 2 summarises the tracking results.

The second experiment tests the tracker's robustness to partial occlusion. The tracked face moves behind the cubicle. Our proposed tracker can recover from momentary failure. When the object verification returns a negative score, the tracker applies a search around the area where it fails. The tracking results are presented in Figure 3. A limitation of the proposed algorithm, also of most trackers, is that the tracker tends to fail if the size of the target is too small. Motion information could be utilised to alleviate this problem.

4 Conclusion

To summarise, we have proposed a novel approach to particle filtering based tracking, which is more robust than conventional trackers. Specifically we use an SVM for training a robust likelihood model, which can represent more complex characteristics of the tracked object and background. It is advantageous when tracking deformable objects and where appearance changes due to the unstable illumination and pose occur.

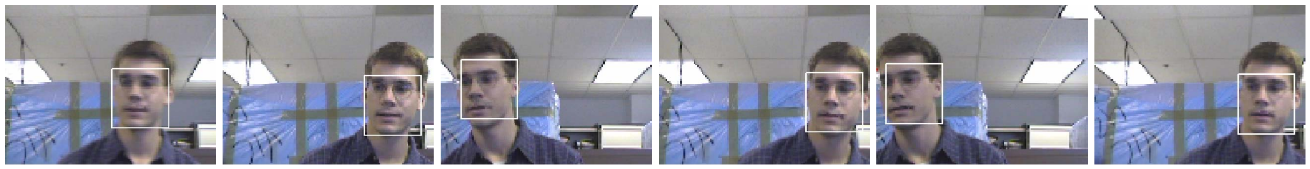

Figure 2. SVM tracking. Frames #1,#3,#7,#20,#24,#30 are shown. The target can be tracked through the whole sequence despite large displacements.

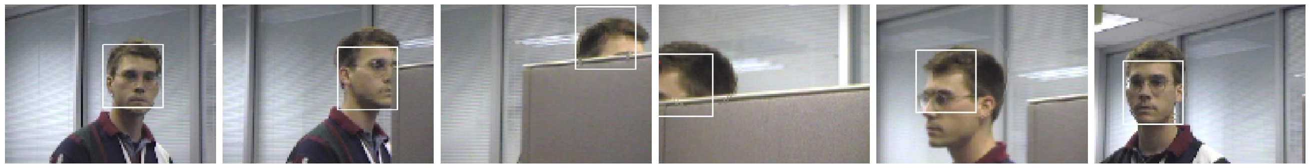

Figure 3. SVM tracking. Frames #4,#11,#18,#38,#42,#49 are shown. In many frames (*e.g.*, #18 and #38 in the figure) the target is partially occluded by the cubic. The SVM tracker survives in these difficult frames.

Future work will focus on the following possible avenues:

- As mentioned, other machine learning approaches such as relevance vector machines [25; 26] or Gaussian processes [27] might be employed to learn the representation model. It is interesting to compare the performances of different approaches;
- More discriminative features such as Gabor filtering responses will be used to make the tracker more robust;
- Continuous updating of the representation model can capture changes of the target appearance/backgrounds. As discussed in Section 1, previous work such as [8; 10; 7; 11] has demonstrated the importance of this on-line update during the course of tracking. The incremental SVM technique well meets this end [28; 29], which efficiently updates a trained SVM function whenever a sample is added to or removed from the training set.

Acknowledgements

National ICT Australia is funded through the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

[1] M. Isard and A. Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. In *European Conference on Computer Vision*, volume 1, pages 893–908, 1998.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, May 2003.

[3] A. Elgammal, R. Duraiswami, and L. S. Davis. Probabilistic tracking in joint feature-spatial spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 781–788, Madison, Wisconsin, 2003.

[4] C. Shen, M. J. Brooks, and A. van den Hengel. Fast global kernel density mode seeking with application to localisation and tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1516–1523, Beijing, China, October 2005.

[5] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, volume 2350 of *Lecture Notes in Computer Science*, pages 661–675, Copenhagen, Denmark, 2002.

[6] J. Vermaak, P. Pérez, M. Gangnet, and A. Blake. Towards improved observation models for visual tracking: Selective adaptation. In *European Conference on Computer Vision*, volume 2350 of *Lecture Notes in Computer Science*, pages 645–660, Copenhagen, Denmark, 2002.

[7] J. Lim, D. Ross, R.-S. Lin, and M.-H. Yang. Incremental learning for visual tracking. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2004.

[8] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 415–422, Kauai, 2001.

[9] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 696–701, San Diego, CA, 2005.

[10] S. Avidan. Ensemble tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, CA, 2005.

[11] B. Han and L. Davis. On-line density based appearance modeling for object tracking. In *IEEE International Conference on Computer Vision*, Beijing, China, 2005.

[12] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, Madison, Wisconsin, 2003.

[13] M. J. Black and A. D. Jepson. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. In *European Conference on Computer Vision*, volume 1, pages 329–342, 1996.

[14] O. Chapelle, P. Haffner, and V. Vapnik. SVMs for histogram based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[15] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[16] V. Vapnik. *The Nature of Statistical Learning Theory*. Spinger Verlag, 1995.

[17] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[18] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, 1998.

[19] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K.-R. Müller. A new discriminative kernel from probabilistic models. *Neural Computation*, 14(10): 2397–2414, 2002.

[20] P. J. Moreno, P. Ho, and N. Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2003.

[21] T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[22] C. Yang, R. Duraiswami, and L. Davis. Efficient spatial-feature tracking via the mean-shift and a new similarity measure. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 176–183, San Diego, CA, 2005.

[23] A. B. Chan, N. Vasconcelos, and P. J. Moreno. A family of probabilistic kernels based on information divergence. SVCL-TR 2004/01, University of California, San Diego, 2004. `http://www.svcl.ucsd.edu`.

[24] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[25] O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1292–1304, 2005.

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

[27] C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): 1342–1351, 1998.

[28] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, pages 409–415, 2000.

[29] G. Fung and O. L. Mangasarian. Incremental support vector machine classification. In *SIAM International Conference on Data Mining*, Arlington, VA, USA, 2002.