Large-Scale Convolutional HMMs for Real-Time Video Tracking

(MPLab TR 2003.04)

Javier R. Movellan, John Hershey, Josh Susskind
Machine Perception Laboratory, Institute for Neural Computation
University of California San Diego, La Jolla, CA 92093

Abstract

Bayesian filtering provides a principled approach for a variety of problems in machine perception and robotics. Current filtering methods work with analog hypothesis spaces and find approximate solutions to the resulting non-linear filtering problem using Monte-Carlo approximations (i.e., particle filters) or linear approximations (e.g., extended Kalman filter). Instead, in this paper we propose digitizing the hypothesis space into a large number, $n \approx 100{,}000$, of discrete hypotheses. Thus the approach becomes equivalent to standard hidden Markov models (HMM) except for the fact that we use a very large number of states. One reason this approach has not been tried in the past is that the standard forward filtering equations for discrete HMMs require order n^2 operations per time step and thus rapidly become prohibitive. In our model, however, the states are arranged in two-dimensional topologies, with location-independent dynamics. With this arrangement predictive distributions can be computed via convolutions. In addition, the computation of log-likelihood ratios can also be performed via convolutions. We describe algorithms that solve the filtering equations, performing this convolution for a special class of transition kernels in order n operations per time step. This allows exact solution of filtering problems in real time with tens of thousands of discrete hypotheses. We found this number of hypotheses sufficient for object tracking problems. We also propose principled methods to adapt the model parameters in non-stationary environments and to detect and recover from tracking errors.

1 Introduction

Bayesian filtering refers to the problem of making inferences about the values taken by random variables at time t based on a sequence of observations up to that time. In computer vision the observed variables are typically image sequences and the unobserved variables are analog in nature (e.g, pose and deformation parameters of an object). For this reason continuous state filtering approaches are the preferred choice. While exact analytical solutions exist for continuous filtering problems the needed assumptions (Gaussianity and Linearity) are too restrictive. Thus Monte-Carlo approximations (i.e., particle filters) have become the method of choice in computer vision and much work is being devoted to making these approximations as efficient as possible [2; 1; 9]. We informally refer to these approaches as "smarticle filters" for their emphasis on improving the accuracy of a small number of particles using clever techniques.

In speech recognition *hidden Markov models* (HMMs) are used to model the dynamics of speech. The observable data of interest are phoneme-like segments of speech and thus can be represented well with a small number of discrete states or *hypotheses* ($n_h \approx 50$). Furthermore the models are typically constrained for state transitions to have left-to-right constraints. The small number of states and state transitions allows the filtering problem to be solved exactly in real time. Unfortunately the filtering equations in the discrete case require order n_h^2 operations per time step and thus do not scale well for large, densely connected hypothesis spaces. This is arguably the main reason why HMMs are not as popular in computer vision as they are in speech recognition.

While in general the discrete filtering equations scale order n_h^2, in most computer vision problems there is spatio-temporal structure that can be used to develop faster inference algorithms. In this paper we present a new algorithm that works in order n_h operations. This makes it possible to solve inference problems in real time with tens of thousands of hypotheses. We informally refer to the approach as "dumbicle filtering" because it relies on a massive number of particles rather than a few clever particles. In fact we exhaustively populate a continuous hypothesis space with a large number of discrete states and solve the resulting inference problem exactly. We describe the algorithm in the context of 2D tracking problems. Extensions are discussed in Section 7.

2 A Generative Model for 2D Tracking

We identify random variables with capital letters, and specific values taken by those variables with small letters. When possible we use shorthand notation and identify probability functions by their arguments. For example, $p(h_t)$ is shorthand for $p_{H_t}(h_t)$, the probability (or probability density) that the random variable H_t takes the specific value h_t. We use subscripted columns to designate sequences. For example $y_{1:t} = y_1 \cdots y_t$.

Finally we reserve Greek letters for parameters.

We model the image generation process as follows (see Figure 1): First a parameter λ_t is chosen by a process described in Section 4. This parameter determines the location-dependent probability distribution of image features in the background $b_i(\cdot \mid \lambda_t)$, and the probability distribution of image features in the object of interest $o(\cdot \mid \lambda_t)$. The image features can be any function of a local patch of pixels, and can represent for instance texture, color, or motion or object categories (see [6]). Without loss of generality we formulate the model here using the color of individual pixels as the feature of interest [1].

The pixels rendered by the object are inside a rectangle of fixed aspect ratio $h_t = (x_t, s_t)$ centered at x_t, with scale parameter s_t. Generalizations to arbitrary rotations and non-rectangular hypotheses are easy (see Section 7) once this case is understood. The rectangle containing the object pixels is chosen with probability $p(x_t s_t \mid x_{t-1} s_{t-1}) = p(s_t \mid s_{t-1}) p(x_t \mid x_{t-1} s_{t-1})$.

Once h_t is known, we know which pixels are rendered by the background and which are rendered by the object of interest. For each pixel location u in the background, a color $y_t(u)$ is chosen with probability $b_i(y_t(u) \mid \lambda_t)$. For each pixel v in the object, a color $y_t(v)$ is chosen with probability $o(y_t(v) \mid \lambda_t)$

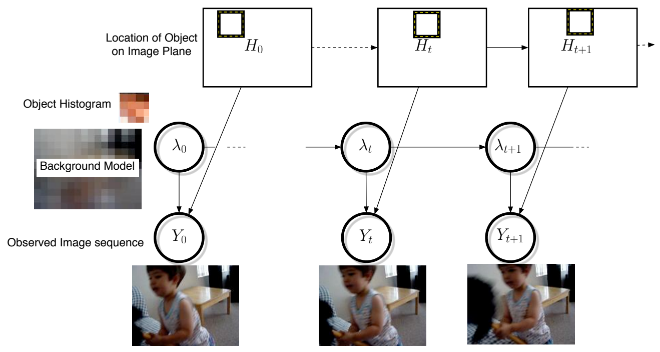


Figure 1: *The hidden variable H determines which pixels belong to the object and which belong to the background. The object pixels are rendered independently from an object histogram. The background pixels are rendered independently from a space variant background histogram model.*

Image Likelihood Let y_t represent the image observed at time t and $y_t(u)$ the value taken by the pixel at location $u \in \mathbb{R}^2$ in that image. From the description of the model above, it follows that:

$$\log p(y_t \mid x_t s_t \lambda_t) = \log \prod_{u \in h_t} o(y_t(u) \mid \lambda_t) + \log \prod_{u \notin h_t} b_i(y_t(u) \mid \lambda_t) \tag{1}$$

$$= \sum_{u \in h_t} \log \frac{o(y_t(u) \mid \lambda_t)}{b_i(y_t(u) \mid \lambda_t)} + Z(y_t, \lambda_t) \tag{2}$$

where

$$Z(y_t, \lambda_t) = \sum_u \log b_i(y_t(u) \mid \lambda_t) \tag{3}$$

The log-likelihood of a hypothesis h_t is a constant Z plus the sum of the log-likelihood ratios of all the pixels within that hypothesis.

Filtering Distribution Let $y_{1:t} = (y_1 \cdots y_t)$ represent an observed image sequence up to time t. Our goal is to compute the filtering distribution, i.e., the posterior distribution of h_t given $y_1 \cdots y_t$. Using the standard HMM update equations we have that the posterior probability of a hypothesis h_t is proportional to the product of the probability of the current image given the hypothesis times the predictive probability of each hypothesis given the past image sequence:

$$p(h_t \mid y_{1:t} \lambda_{1:t}) = \frac{p(y_{1:t-1} \mid \lambda_{1:t})}{p(y_{1:t} \mid \lambda_{1:t})} p(y_t \mid h_t \lambda_{1:t}) p(h_t \mid y_{1:t-1} \lambda_{1:t-1}) \tag{4}$$

$$p(h_t \mid y_{1:t-1} \lambda_{1:t-1}) = \sum_{s_{t-1}} p(s_t \mid s_{t-1}) \sum_{x_{t-1}} p(x_t \mid x_{t-1} s_{t-1}) p(h_{t-1} \mid y_{1:t-1} \lambda_{1:t-1}) \tag{5}$$

where $h_t = (x_t, s_t)$, $h_{t-1} = (x_{t-1}, s_{t-1})$. For each scale, we let the transition distribution $p(x_t \mid x_{t-1} s_t)$ be rectangular uniform and shift invariant. This allows using cumulative probability maps to compute the predictive probability of each hypothesis with four operations per hypothesis.

Minimum Risk Estimation In many applications we need to choose a single hypothesis per time step. In such cases it is reasonable to choose the hypothesis that minimizes the posterior risk, i.e., the expected average of an error function

$$\hat{h}_t = \operatorname*{argmin}_h E(\rho(H_t; h) \mid y_{1:t} \; \lambda_{1:t}) \tag{6}$$

where ρ is an error function that measures the mismatch between two hypotheses. We experimented with two types of error functions: (1) The correct hypotheses get zero error and incorrect hypotheses get error 1; (2) An error function that measures average distance between corresponding object landmarks in two hypotheses. The first error function is minimized by choosing the hypothesis with maximum posterior probability (MAP). The Appendix shows the minimum posterior estimate for the second error function.

[1]Here color refers to an rgb value and thus it may include intensity, not just hue and saturation.

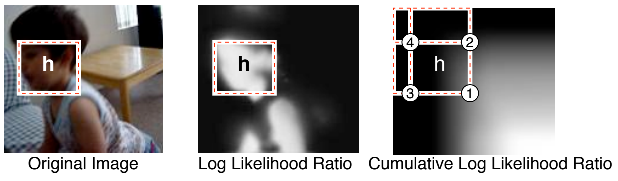


Figure 2: *The log-likelihood of a hypothesis is the sum of the log-likelihood ratios of each pixel. This can be computed in 4 operations using cumulative log-likelihood ratio maps: $l(h) = L(1) - L(2) - L(3) + L(4)$, where $L(i)$ is the cumulative log likelihood ratio evaluated at pixel i.*

3 Computational Complexity

To compute the filtering distribution at time $t+1$ first we need to to compute the predictive distribution at time $t+1$. This is the distribution of hypothesis at time $t+1$ based on the observed images up to time t. The predictive distribution contains all the information about the hypotheses prior to the observation of the image at time $t+1$. It is obtained by propagating the filtering distribution at time $t+1$ via the state transition function $p(h_{t+1} \mid h_t)$. Following Bayes rule we then need to compute the likelihood of the image at time $t+1$ for each possible hypothesis and multiply the predictive probability (prior) times the likelihood of each hypothesis. In this section we describe methods to achive the desired results in order $n_h + n_p$ operations, making it possible to work with a very large number of hypothesis in real time.

Double Integral Likelihood Ratio Maps A brute force approach for computing the likelihood-ratios would require order $n_p \times n_h$ sums, where n_p is the number of pixels on the image and n_h the number of hypotheses. In practice we can compute the log-likelihood ratio of all the hypothesis using $n_p + 4n_h$ sums. First for each pixel location $x = (x_1, x_2)^T$ we compute the likelihood ratio of the value taken by that pixel

$$l(x) = \log \frac{o(y_t(x) \mid \lambda_t)}{b_x(y_t(x) \mid \lambda_t)} \tag{7}$$

This can be done using table-lookups for the likelihood-ratio function. Then we compute the double integral log-likelihood ratio map L:

$$L(x) = \sum_{u_1=0}^{x_1} \sum_{u_2=0}^{x_2} l(x) \tag{8}$$

Once L is known, the probability of each hypothesis can be computed in 4 operations (see Figure 2).

Double Derivative Predictive Maps For the general case the problem of computing the predictive probability from the filtering probability takes n_h^2 operations

$$p(x_{t+1} s_{t+1} \mid y_{1:t} \lambda_{1:t}) = \sum_{x_t, s_t} p(x_t s_t \mid y_{1:t} \lambda_{1:t}) \, p(x_{t+1} s_{t+1} \mid x_t s_t) \tag{9}$$

$$= \sum_{s_t} p(s_{t+1} \mid s_t) \sum_{x_t} p(x_t s_t \mid y_{1:t} \lambda_{1:t}) p(x_{t+1} \mid x_t s_t) \tag{10}$$

If the transition probabilities are shift invariant this amounts to a convolution operation for each of the scales, with cost of order $n_s \times n_p \log n_p$, where n_s is the number of scales under consideration. Here we propose a new approach that allows updating in $n_s \times (n_p + 4n_x)$ operations. The method relies on propagation of probability derivatives. Once the probability derivative map is ready, the actual probabilities are obtained by integration. Let the double derivative of a probability mass p be as follows follows

$$\nabla_x^2 p(x) = \sum_{i,j \in \{-1,1\}} p(x + u_{ij}) \tag{11}$$

where $u_{ij} = (i,j)^T$. Thus

$$\nabla_x^2 p(x_{t+1}, s_{t+1} \mid y_{1:t} \lambda_{1:t}) = \sum_{s_t} p(s_{t+1} \mid s_t) \sum_{x_t} p(x_t s_t \mid y_{1:t} \lambda_{1:t}) \nabla_x^2 p(x_{t+1} \mid x_t s_t) \tag{12}$$

and since $p(x_{t+1} \mid x_t s_t)$ is a square centered at x_t and with height $2s_t$ it follows that

$$\nabla_x^2 p(x_{t+1} \mid x_t s_t) = \begin{cases} 1 & \text{if } x_{t+1} = x_t \pm (1,1)^T \\ -1 & \text{if } x_{t+1} = x_t \pm (-1,1)^T \\ 0 & \text{else} \end{cases} \tag{13}$$

Table 1 shows the cost of an iteration of the filtering algorithm. By use of cumulative log-likelihood ratio maps and rectangular transition probabilities, the cost is order $n_h + n_p$.

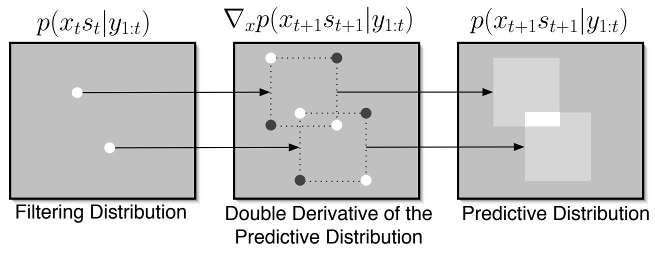
Thus, to construct the gradient predictive probability map we just need to send four numbers per hypothesis (one for each of the corners of the square centered at the center of the hypothesis). Once the gradient map is built, the predictive probability can be obtained by integrating the map, which costs n_p operations per map and obtaining the value of the integral map at x_{t+1} for scale s_t (see Figure 3

$$p(x_{t+1} s_{t+1} \mid y_{1:t}) = \sum_{i=1}^{x_{t+1}(1)} \sum_{j=1}^{x_{t+1}(2)} \nabla_x^2 p(x_{t+1} s_{t+1} \mid y_{1:t}) \tag{14}$$

The standard forward filtering recurrence for HMMs requires order n_h^2 operations per time step, where n_h is the number of hypotheses. The use of cumulative probability maps reduces it to $8n_h$ algebraic operations

Task	Sum/Diffs	Prods/Ratios	Exps	LLRs	If
CLR	n_p			n_p	
LI	$4n_h$		n_h		
PD	$4n_h + n_p$				
UFD		n_h			
NFD	n_h	n_h			
MAP					n_h
MRS	n_h	1			
MRL	n_h	$n_h + 1$			

Table 1: *n_h: Number of hypothesis; n_p: Number of pixels; LLR: Log-likelihood ratio of a single pixel; If: Logical "if" operation; CLR: Cumulative Likelihood Ratio Map; LI: Likelihoods; PD: Predictive Distribution; UFD: Unnormalized Filtering Distribution; NFD: Normalized Filtering Distribution; MAP: Maximum Posterior Hypothesis; MRS: Minimum Risk Scale; MRL: Minimum Risk Location. Log-likelihood ratios (LLR) and exponentials can be implemented via look-up tables.*



$p(x_t s_t|y_{1:t})$
$\nabla_x p(x_{t+1} s_{t+1}|y_{1:t})$
$p(x_{t+1} s_{t+1}|y_{1:t})$
Filtering Distribution
Double Derivative of the Predictive Distribution
Predictive Distribution

Figure 3: *The double derivative method for computing the predictive distribution. For each hypothesis from the filtering distribution we add 4 numbers (2 positive and 2 negative) at the corners of the transition probability kernel for that hypothesis. The sum of all these numbers is the double derivative map of the predictive distribution. The double integral of this map gives us the desired predictive distribution.*

and n_h exponentials. This allows filtering problems with about 100,000 hypotheses to run in real time on a state of the art PC. In practice this provides more than enough resolution for difficult Tracking problems (see Section 5).

4 Unknown, Non-stationary Model Parameters

The appearance of the background and the object of interest may change due to changes in illumination, camera movement, or the movement of objects in and out of the image plane. Thus we need a scheme to adaptively change the model parameters.

Let $\mathcal{M}$ represent the set of possible image generation models. When the model is unknown and non-stationary, optimal inference calls for marginalizing across all possible image sequence models

$$p(h_t \mid y_{1:t}) = \sum_{\lambda_{1:t}} p(\lambda_{1:t} \mid y_{1:t}) p(h_t \mid y_{1:t} \lambda_{1:t}) \qquad (15)$$

where $p(\lambda_{1:t} \mid y_{1:t}$ is the parameter adaptation term. In practice we need to approximate this by: finding reasonable estimates $\hat{y}_{1:t}$ and letting $p(\lambda_{1:t} \mid y_{1:t}) \approx \delta(y_{1:t}, \hat{y}_{1:t})$, i.e.,

$$p(h_t \mid y_{1:t}) \approx p(h_t \mid y_{1:t} \hat{\lambda}_{1:t}) \qquad (16)$$

This approximation while efficient is risky and thus methods are needed to detect when the approximation is not working well and to recover from error.

In our current implementation we rely on an auxiliary object detector whose main role is to help with parameter estimation and error recovery of the primary object tracker. The secondary detector is set to have very small number of false alarms, thus when it detects the object of interest we can safely assume that the object was there. The disadvantage of the auxiliary detector is higher computational cost than the primary detector and the fact that it can only detect the object of interest in a particular pose. In our experiments the auxiliary detector is a Viola & Jones [10] style detector of frontal/upright faces described in [6].

Non Stationary Environments We model changes in illumination, camera movement and background movement as a continuous time Poisson jump process: The background and object models are constant except for specific jump points that occur at unknown random times. The time between jump points is independent of previous jump points and is governed by an exponential density function with parameter θ. At jump points new model parameters are chosen from a distribution of known mean.

Let $T_1, T_2, \cdots$ represent the unknown times at which the object and background model changed (the jump times). Let $S_1, S_2, \cdots$ be the unknown object and background parameters chosen at jump times $T_1, T_2, \cdots$. Since these values are independent samples from a continuous random vector it follows that $P(S_i = S_j) = 0$ if $i \neq j$. Let λ_t represent the unknown color and background models at time t, i.e.,

$$\lambda_t = S_{T_i} \text{ for } T_i \leq t < T_{i+1} \qquad (17)$$

Suppose by time t the auxiliary detector has found the object of interest at times $\tau_1 < \tau_2 < \cdots < \tau_n \leq t$. By doing so it provided samples pixels from the background and from the object. Lee $x_i = (h_{\tau_i}, y_{\tau_i})$ represent the information provided by the auxiliary detector at time τ_i. Our goal is to use this information to obtain estimates of λ_t. One reasonable estimate is the posterior mean of λ_t given $x_1 \cdots x_n$. Let A_{n+1} be the event that at least one jump occurred after τ_n. Thus

$$P(A_{n+1}) = p(\lambda_t \neq \lambda_{\tau_n}) = e^{-\theta(t-\tau_n)} \qquad (18)$$

For $j = 2, \cdots n$ let A_j represent the event that the last jump occurred between τ_{j-1} andτ_j. Thus

$$A_j = \cap_{i=j}^{n} \{\lambda_t = \lambda_{\tau_i}\} \cap_{i=1}^{j-1} \{\lambda_t \neq \lambda_{\tau_i}\} \qquad (19)$$

i.e., the probability that at least a jump occurred between τ_{j-1} and no jump occurred afterwards:

$$P(A_j) = (1 - e^{-\theta(\tau_j - \tau_{j-1})})e^{-\theta(t-\tau_j)} \tag{20}$$
$$= e^{-\theta(t-\tau_j)} - e^{-\theta(t-\tau_{j-1})} \tag{21}$$

Finally let A_1 be the event that the last jump occurred prior to τ_1. Thus

$$A_1 = \cap_{i=1}^n \{\lambda_t = \lambda_{\tau_i}\} \tag{22}$$

$$P(A_1) = e^{-\theta(t-\tau_1)}, \ \sum_{j=1}^{n+1} P(A_j) = 1 \tag{23}$$

Thus

$$\hat{\lambda}_t = E(\lambda_t | x_{1:n}) = P(A_{n+1})E(\lambda_t) + \sum_{j=1}^{n} P(A_j)E(\lambda_t | x_{j:n}\ A_j) \tag{24}$$

where $E(\lambda_t \mid x_{j:i}\ A_j) = E(\lambda_t \mid y_{\tau_1}h_{\tau_1} \cdots y_{\tau_i}h_{\tau_i})$ are the object and background histograms obtained by segmenting the images $y_{\tau_1} \cdots y_{\tau_i}$ into object and background as determined by $h_{\tau_1} \cdots h_{\tau_i}$ clumping all the object pixels together and all the background pixels from the same location together. After some algebra it can be shown that $\hat{\lambda}_t$ consists on weighted frequency counts of colors found in the object and the background locations, where the weight of a pixel decays exponentially with the length of time since the pixel was collected.

Error Detection and Recovery The auxiliary object detector may be slow or may run with low priority; thus it may provide information with some delay. Suppose at time t the auxiliary detector tells us that at time $t - \Delta$ the correct hypothesis was $h_{t-\Delta}$. We also have information that at that time the tracker chose $\hat{h}_{t-\Delta}$. Ideally we should go back in time and propagate forward the new information. However due to the fact that we adapt the model parameters λ based on our knowledge about $h_{t-\Delta}$ this would require buffering the distribution $p(h_{t-\Delta} \mid y_1 : y_{t-\Delta}\hat{\lambda}_{t-\Delta})$ and the image sequence $y_{t-\Delta:t}$. If this information is lost by time t, we are presented with the problem of combining two experts whose opinions are derived from different information sources: (1) The auxiliary detector provides us with $p(h_t \mid h_{t-\Delta})$, which does not make any assumptions about λ. However if Δ is large, $p(h_t \mid h_{t-\Delta})$ will be almost flat, and thus uninformative. (2) The main tracker provides us with $\hat{p}(h_t \mid y_{1:t}\hat{\lambda}_{1:t})$, which relies on the assumption that $\lambda_{t:t}$ is a good estimate of the actual object and background models. A reasonable approach is to choose the minimum risk expert. The risk for expert 1 is

$$R_1 = \min_h E\left(\rho(H_t, h) \mid h_{t-\Delta}\right) \tag{25}$$

The risk for expert 2 is

$$R_2 = E\left(\rho(H_t, \hat{H}_t) \mid h_{t-\Delta}, \hat{h}_{t-\Delta}, \Delta\right) \tag{26}$$

If $R_1 < R_2$ we discard the current distribution and restart the system with the distribution $p(h_{t+\Delta} \mid h_t)$ proposed by the auxiliary detector. In practice we model R_2 using some reasonable heuristic (28) or by using a labeled dataset in which we estimate how the error of the system changes as a function of time Δ and the starting error $\rho(h_{t-\Delta}, \hat{h}_{t-\Delta})$.

5 Simulations

A video tracking simulation was performed on a dataset comprised of five minutes of video. Footage was collected from three subjects. Each subject performed two action sequences consisting of rapid camera movements, in plane translations, rotations, and hand/arm occlusions. The goal was to simulate the very difficult tracking conditions typically found in pet robots. During the sequence the lighting was changed by adding two blue illumination sources. The resulting video footage was converted to 160x120 color images. The simulation was developed and tested on a 3.0 GHz Pentium 4 computer.

The model is specified by the initial distribution of H, the transition kernel $p(h_{t+1} \mid h_t)$, the average time $1/\theta$ between parameter jumps, and the prior distribution for object and background models, the error function ρ and the risk estimation method. In the experiments presented below we used the following architecture: The initial distribution for H was uniform across hypotheses, the transition kernel was uniform rectangular with width and height equal to 1/2 the scale of the parent hypothesis. The average time between model jumps was set at 5 seconds (i.e., $\theta = 0.2$ seconds. The face color model for the tracker was implemented as a twenty-bin histogram. The prior distribution for the background and object models was assumed to be flat. The prior histogram model for faces was based on the model published in Jones [5]. For risk estimation we used the following

$$R_1 = \min_h E(\rho(H_t, h) \mid h_{t-\Delta}) = \max_{h_t} p(h_t \mid h_{t-\Delta}) \tag{27}$$

$$E\left(\rho(H_t, \hat{H}_t) \mid h_{t-\Delta}, \hat{h}_{t-\Delta}, \Delta\right) \approx p(h_{t-\Delta} | y_{1:t-\Delta}\lambda_{1:t-\Delta}) \tag{28}$$

The system could run in real time with a space of 100,000 hypotheses.

6 Previous Work

The use of double integral functions to measure rectangular sets is well known in measure theory, probability theory, and statistics. This approach is also used in computer animation for fast rendering of rectangular objects. Viola and Jones [10] were first to use this method in computer vision problems for computing the output of rectangular feature detectors. Stochastic filtering approaches to tracking have been popular for more than a decade in the computer vision community. Most approaches nowadays find approximate solutions to the filtering problem using Monte-Carlo methods

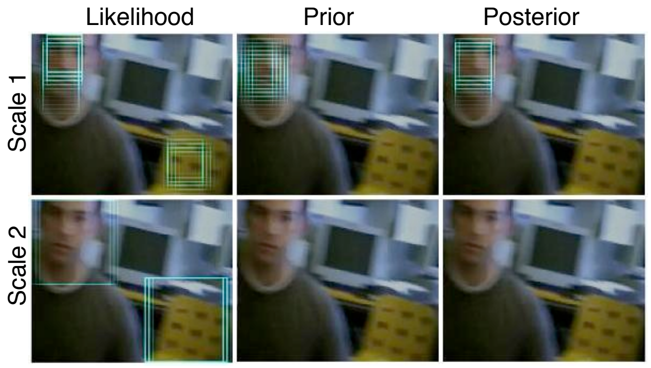


Figure 4: *The two rows of images represent hypotheses at two different scales. The left column represents the most likely hypotheses. The center image represents the prior distribution based on the previous image, the right-sice column shows the posterior distribution of hypohteses.*

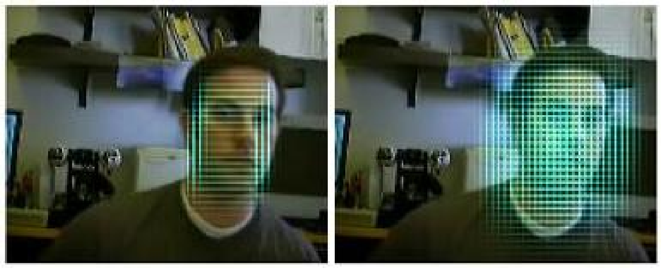


Figure 5: *An example of uncertainty propagation. The left side shows the most probable hypotheses at time t, i.e., the filtering distribution. The image on the left shows the predictive distribution for time $t+1$, i.e, the prior distribution for the nexst time step.*

(particle filters). Monte-Carlo approaches to filtering were first described in [3] and introduced to the computer vision community by [4]. Current work in this field has focused on the development of intelligent sampling methods to find good approximations with very few particles [1; 9; 8].

As far as we know the current document is the first one to point out that the double integral method can be used to compute likelihood-ratio maps. The use of double derivative maps to compute predictive probabilities was also unknown to us before we produced this document.

7 Extensions

While the specific architecture presented here is limited to upright rectangular hypothesis, additive log-likelihood functions, and rectangular transition probability kernels, extensions are possible that preserve the computational complexity of the method while providing great generality.

Complex geometries can be obtained by producing double cumulative functions at several orientations. The transition kernels or the features underlying the likelihood computation do not need to be uniform. For example, one can apply double cumulative sums recursively (cumulative sums of cumulative sums ...) to obtain Gaussian-like transition kernels and Gabor-like features. More complex likelihood-ratio functions are also possible, using such kernels as object windowing functions.

In this paper we treat objects a single blobs. Multi-part objects can also be tracked using as many likelihood-ratio maps as object parts. Coarse histogram matching can also be done using one likelihood-ratio map per histogram bin. In a different paper we showed how discriminative methods, like Gentle-Boost can be used to learn complex likelihood-ratio maps that can also be computed very efficiently [6].

8 A General Architecture for Machine Perception and Robotics

We described methods for solving the stochastic filtering problem in order n_h operations. This allows us to work with tens of thousands of hidden states, and solve large-scale non-linear filtering problems exactly in real time. While we focused on visual tracking problems, the methods proposed here can be used in a wide variety of real-time machine perception and robotics problems.

The algorithms presented in this paper exploit the fact that in many computer vision problems the computation of likelihoods and the propagation of probabilities are convolutional. The methods presented here solve these convolutions in order n_h operations. Frequency domain methods could also be used, which would work in order $n_p \log n_p$ operations.

HMMs are already the architecture of choice for speech recognition problems. Working with a similar architecture in vision facilitates approaching problems that require combination of acoustic and visual information (e.g., audio-visual tracking, audiovisual speech recognition). Particle filter approximations are being used in robotics for real-time inference and control problems[7; 8]. The architecture presented here may allow these problems to be solved more efficiently.

The proposed architecture shows a surprising resemblance to the functional architecture of visual cortex: A set of topographical organized hyper-columns, where each hyper-column has scaled and rotated replicas of the same detectors. The hyper-columns are interconnected by lateral connections (see Figure 6). The short distance lateral connections in the G-flow filtering algorithm take care of propagation of probability maps. These maps represent the posterior distribution of hypotheses given an observed video sequence. We can now efficiently simulate such systems on a grand scale.

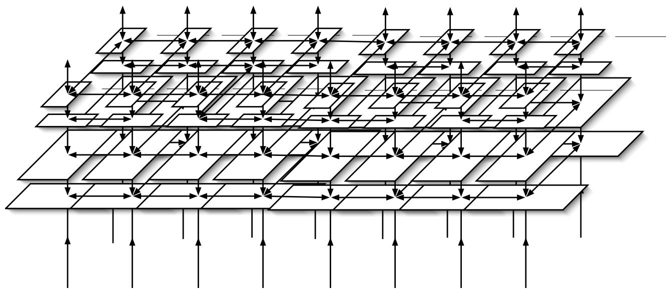


Figure 6: *The optimal inference algorithm can be implemented using a topographically organized set of columns, where each column computes the likelihood ratio of rotated and scaled versions of an image patch.*

9 Appendix: Minimum Risk Estimation

For generality we allow different object landmarks to have different weights, by using a normalized relevance map w. Let u represent a point on the image plane. Its standardized location with respect to the hypothesis x, s is $z = (u - x)/s$. The weight of this point is given by $w(z)$. Let μ_x, σ_x^2 represent the mean and variance of the hypothesis x, s with respect to the relevance map w, i.e.

$$\mu_x = \int u \, w(\frac{u-x}{s}) du \tag{29}$$

$$\sigma_x^2 = \int (u - \mu_x)^2 \, w(\frac{u-x}{s}) du \tag{30}$$

Now consider a different hypothesis x', s'. According to this hypothesis the landmark u from hypothesis x, s is located at $\frac{s'}{s}(u-x)+x'$. The scaled average distance from equivalent landmarks follows:

$$\rho^2(x, s; x', s') = \frac{1}{\sigma_x^2} \int \|u - \frac{s'}{s}(u-x) + x'\|^2 \, w(\frac{u-x}{s}) du \tag{31}$$

This error function has an intuitive interpretation as the expected distance between corresponding landmarks in the two hypotheses: For example if $\rho^2 = 0.25$ the average error is in the order of 0.5 times the scale of the standard object. After some simple derivations it can be shown that

$$\rho^2(x, s; x', s') = \left(\frac{s_i - s_i'}{s}\right)^2 + \left(\frac{\mu_x - x'}{\sigma_x}\right)^2 \tag{32}$$

For simplicity we can choose a relevance map such that $\mu_x = x$ and $\sigma_x = s$, in which case

$$\rho^2(x, s; x', s') = \left(\frac{s_i - s_i'}{s}\right)^2 + \left(\frac{x - x'}{s}\right)^2 \tag{33}$$

The error between two hypothesis (x, s) and (x', s') is simply the sum of the squared scaled difference of locations plus the squared scaled difference of the scales between the two hypotheses.

The minimum risk hypothesis for this error function can be found by differentiating the posterior risk with respect to x, s, setting it to zero and solving the resulting equation. The results are as follows:

$$\hat{s} = \frac{1}{E(1/S \mid y_{1:t} \; \lambda_{1:t})}; \quad \hat{x} = \frac{E(X/S \mid y_{1:t} \; \lambda_{1:t})}{E(1/S \mid y_{1:t} \; \lambda_{1:t})} \tag{34}$$

Acknowledgements

This research was funded by a UC Discovery grant, NSF IIS-0086107, and NSF IIS-0220141.

References

[1] C. Andrieu and A. Doucet N. de Freitas. Rao-blackwellised particle filtering via data augmentation. In *Advances in Neural Information Processing Systems*, number 13. MIT Press, Cambridge, Massachusetts, 2001.

[2] Hans Burkhardt and Bernd Neumann, editors. *ICONDENSATION: Unifying Low-Level and High-Level Tracking in a Stochastic Framework.*, volume 1406 of *Lecture Notes in Computer Science*. Springer, 1998.

[3] J. E. Handschin and Mayne D. Q. Monte-carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547–559, 1969.

[4] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conf. Computer Vision*, volume 58, pages 343–356, Cambridge, UK, 1996.

[5] James M. Rehg Michael J. Jones. Statistical color models with application to skin detection. *IEEE Computer Vision and Pattern Recognition*, 1:1274–1280, 1999.

[6] J. R. Movellan, B. Fortenberry, and I. Fasel. A generative framework for real-time object detection. *UCSD MPLab Technical Report 2003.02*, 2003.

[7] Sebastian Thrun. Particle filters in robotics.

[8] Sebastian Thrun, John Langford, and Vandi Verma. Risk sensitive particle filters. In *Neural Information Processing Systems (NIPS)*, 2001.

[9] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan. The unscented particle filter. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, number 12. MIT Press, Cambridge, Massachusetts, 2000.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001.