Improving Monte-Carlo Bayesian Filtering by Means of Local Search: Theory and Applications in Visual Tracking

Péter Torma
Eötvös Loránd University
Mindmaker Ltd.
e-mail: tyus@mindmaker.hu

Supervisor: Csaba Szepesvári
Mindmaker Ltd.
H-1121 Budapest,
Konkoly Thege Miklós út 29-33.

15th of January, 2001

Abstract

With computational power doubling every 1.5 year, non-linear filters are gaining importance and popularity. This facilitates the emergence of new ideas in control theory, artificial intelligence, image processing, and more specifically in object tracking. A widely accepted broad framework for studying non-linear filtering is that of the framework of discrete-time Hidden Markov Models (HMM).

In this thesis we will consider non-linear approximate Bayesian filtering applied to the state estimation problem of continuous state-space HMMs. Bayesian filtering is the process of calculating the posterior given a model and a prior. In approximate Bayesian filtering only approximate knowledge of the HMM generating the observations is available to the observer. One problem that we will consider in this thesis is the Bounded Input Bounded Output (BIBO) stability of the certainty-equivalence Bayesian filter, i.e., if the error of the response of the certainty-equivalence Bayesian filter can be bounded by the approximation error of the model, uniformly in time. The filter's stability properties shall be related to the ergodicity properties coefficient of the Markov process corresponding to the filter-dynamics. The stability analysis will also be extended to the practical case when the filter response is calculated approximately e.g. by means of a sequential Monte-Carlo algorithm. Interestingly, no similar stability analysis existed in the literature prior this work. Convergence, stability and finite-sample size bounds are derived.

An algorithm, called LS-N-IPS, extending sequential Monte Carlo Bayesian filtering is proposed and is shown to be more efficient than the baseline algorithm when the observation density function is sufficiently "peaky". The algorithm adds local search to the baseline algorithm: in each time step the prediction of the approximate model is refined in a local search procedure that utilizes the most recent observation. It is shown by means of a heuristic argument and some numerical examples that the algorithm is indeed superior than the baseline algorithm.

Experiments are performed on various visual-tracking problems. In one set of experiments an artificial object with a homogeneous color is tracked, whilst in another set the tracking of the hand of a human being is considered. The state space of tracking is chosen to be the multi-dimensional manifold encoding the position, orientation, scale and the velocity of the object to be tracked. The observation model is based on a spline model of the object's shape combined with a color-model, the observation density being modeled in a somewhat ad-hoc way assuming quasi-independence of parts of the contour being far away of each other, and independence of the object's color and contour. Dynamics is modeled

by means of a MIMO AR(2) model.

Experiments are performed in a natural, cluttered indoor scene. Tracking performance in terms of total tracking time, recovery time after occlusion, and behavior in the presence of ambient illumination changes were studied. Results show a significant improvement as compared to the behavior of the baseline filtering algorithms.

In summary, in this thesis:

1. We prove a new robust stability theorem concerning particle filters.

2. We discuss the limitations and possible extensions of the proven theorem.

3. We propose a new particle filtering algorithm (LS-N-IPS) and show that it is significantly more efficient than its closest relatives provided that the observations are "reliable".

4. We propose an implementation of the newly proposed particle filter for visual object tracking based on a combination of Gaussian color, and B-spline contour models. We derive an algorithm for approximating the matrix that transforms support points into control points in an efficient way.

5. We show that the proposed algorithm significantly outperforms baseline algorithms in tracking visual objects in natural indoor environments. In particular, we demonstrate near real-time tracking performance without the use of (unreliable) color information.

Contents

Chapter 1

Introduction

1.1 The Filtering Problem

Let us consider the discrete time stochastic system

$$X_{t+1} = f(X_t) + V_t, \tag{1.1}$$
$$Y_t = g(X_t) + W_t, \tag{1.2}$$

where $t = 0, 1, 2, \ldots$ denotes the time and V_t, W_t are martingale difference series such that the observation density $p(Y_t = y|X_t = x)$ exists. $X_t \in \mathcal{X}$ is called the *state* of the system at time t, $\mathcal{X}$ is called the state-space, $Y_t \in \mathcal{Y}$ is called the *observation* at time t, $\mathcal{Y}$ is the observation space. In this report $\mathcal{Y}$ and $\mathcal{X}$ will be taken to be measurable Polish-spaces with the Borel sigma algebra, unless otherwise noted. $f : \mathcal{X} \to \mathcal{X}$, a measurable mapping, is called the *dynamics* of the system and $g : \mathcal{X} \to \mathcal{Y}$, another measurable mapping, is called the *observation model*. The pair (f, g) is called the *model* of the system.

The filtering problem consists of the estimation of the posterior distribution of X_t given the past observations $Y_{0:t} = Y_0, \ldots, Y_t$. The posterior at time step t will be denoted by π_t, (supressing the dependence on the observations $Y_{0:t}$ and the model f, g) :

$$\pi_t(A) = P(X_t \in A|Y_{0:t}),$$

where $A \subset \mathcal{X}$ is any measurable subset of $\mathcal{X}$.

The filtering problem has an analytic solution that can be obtained by the repeated application of applying the Bayes-theorem:

$$\pi_{t+1} = \frac{G_{Y_{t+1}} F \pi_t}{(G_{Y_{t+1}} F \pi_t)(\mathcal{X})}, \tag{1.3}$$

where $F, G_y : M(\mathcal{X}) \to M(\mathcal{X})$ are defined by

$$(F\pi)(A) = \int K(x, A) d\pi(x), \tag{1.4}$$
$$(G_y \mu)(A) = \int_A g(y|x) d\mu(x). \tag{1.5}$$

Here $K(x, A)$ is the (transition) kernel associated with (1.1) and $g(y|x)$ is the observation density: $g(y|x) = p(Y_t = y|X_t = x)$.[1]

Unfortunatly, the solution of (1.3) cannot be computed analytically except in a few exceptional cases when e.g. the model is simple, like in the case of Kalman filters, or when the state and the observation spaces are finite and small. In practice, these simplifications are often invalid and may lead to large errors in the estimates. Therefore a large body of current work in the filtering literature is devoted to the approximate solution of this equations.

1.2 Visual Tracking as a Filtering Problem

Visual tracking of objects can be cast as a filtering problem as follows: In a typical tracking problem we are interested in the position, pose, and velocity of the object to be tracked, in the outer 3D space, i.e.: the state X_t will correspond in this case to the concatenation of the position, pose and the velocity information. The observation, Y_t shall correspond to the observed image at time step t. Equation (1.2) tells us that this image can be obtained as a function of the state of the system, plus some noise.

Knowledge of the posterior distribution, π_t, shall allow us to derive all kind of properties of the state of the system. We can, for example, compute the expected value of the state, $E[X_t|Y_{0:t}]$. Given π_t one can also compute higher order moments of it, e.g. the variance, etc.

What assumptions do we make when we cast the tracking problem as a filtering problem? Although, the autonomous state evolution model itself can be demanding on its own right (most objects to be tracked do interact with other objects - a fact not captured by Equation (1.1)), the validity of the observation model puts a larger constraint on the system. If, the "background" is not static, but if there are non-random (e.g. periodic) elements of it then there will be no function g such that the observation equation will be satisfied.

Typically we are going to work with linear dynamics and non-linear observation models. The highly non-linear nature of the observation model prevents us from using Kalman-filtering and simple extensions of it. In particular, in visual tracking the posterior is often multi-modal. Hence the need for advanced methods.

In visual tracking, the most complicated model is the observation model itself. In this report, we will work with contour (shape) and color observation models. The contour model we use will be similar to that of Blake and Isard [1] in that we will use splines. However, we devoted a great deal of work to overcome one problem with the Blake and Isard approach, namely that they define the splines by means of their control points. Splines defined by their control points provide a natural and mathematically elegant parameterization of spline curves, but has serious deficiencies as first class shape models. For example, they are not guaranteed to be loop free and their smoothness is largely uncontrolled. This is

[1] In the literature Equation (1.3) is sometimes called the Zakai equation, or the Feynman-Kac formula for the posterior. Equation (1.3) also arises in biological studies, e.g. in the theory of genetic algorithms.

not problematic as long as the shape to be tracked is convex. Since we want the model to be able to track e.g. human hands with opened fingers, we need a better model of shapes.

Therefore, we developed a shape-model that uses interpolating splines defined by their support-points as its starting point and which also guarantees that the generated spline will be loop-free. For the sake of keeping the efficiency of the algorithms, we propose an algorithm that is capable of approximating the splines with a small error and is computationally efficient.

1.3 Filtering with Approximate Models

Where do the dynamics and the observation models come from? In certain cases, we have some cues about the dynamics of the filter to be tracked (e.g. we know that the state evolution is subject to Newton's laws), or we have some cues about the observation. Still, in most of the cases we do not know the exact models, and thus we must first identify the system.

No matter how we do that, in fact, the best we can hope at the end of the identification process is to have an upper bound, or tolerance on the error of approximation of the derived models. We must then use approximate models instead of the true (unknown) ones in the filtering process.

The question then arises about the effect of the approximation error. Can this effect the filter in a bad way? Or is it always the case that approximation errors are harmless from the point of view of filtering error?

We are going to use the properties of the Hilbert-projection metrics to show that if the original system dynamics is "nice" enough then the approximation errors do not accumulate and are indeed harmless. More exactly, we are going to bound the filtering error in terms of the approximation error of the models. Looking at the error-equations, if the filtering error is viewed as the input and the approximation error as the output then our result can be regarded as a bounded-input bounded-output (BIBO) stability result.

The theory that will be developed here is far from being complete: we know of classes of systems about which we believe that their filtering equation is BIBO-stable but which lie outside of the realm of our theorems. In pursue for the full characterisation of systems having a BIBO-stable filtering equation, we give a number of counter-examples when the filtering equation will not be BIBO-stable.

1.4 N-IPS: Approximate Filtering by Sequential Monte-Carlo Methods

As it was noted above, the exact solution of the filter evolution equation (1.3) is not feasible in general. One approach to overcome this problem is to use sequential Monte-Carlo methods to sample from the posterior. The underlying idea of sequential Monte-Carlo methods is that if one has an unbiased weighted sample $(X_t^{(i)}, w_t^{(i)}), i = 1, 2, \ldots, N$ from

the posterior π_t (i.e., $E[\sum_{i=1}^N f(X_t^{(i)})w_t^{(i)}] = \int f(x)d\pi_t(x)$) then an unbiased sample from π_{t+1} can be generated by the following two-step method:

1. **Prediction:** $X_{t+1}^{(i)} = f(X_t^{(i)}) + V_t^{(i)}$, $i = 1, 2, \ldots, N$,
2. **Evaluation:** $w_{t+1}^{(i)} \propto w_t^{(i)} g(Y_{t+1}|X_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$.

In the evaluation step the weights are normalized so that they will sum to 1. Elements of the sample $(X_t^{(i)})$ are traditionally called particles and the filtering method is called particle filtering. Here $V_t^{(i)}$ are random variables having the same law as V_t.

Although, over finite intervals this method might work well provided that the number of particles, N, is large enough, it will not work over unbounded intervals of time (i.e., when one does not know a bound on the filtering interval) since, in order to generate a bounded tracking error, the number of particles should scale, in general, with the length of the interval. Therefore this simple method is not practical in general. The problem of this method is that given any finite sample size the weights $w_t^{(i)}$ will degenerate: $w_t^{(i)}$ will converge to 0 except for one index, for which $w_t^{(i)}$ will converge to 1.

In order to prevent this degeneration, a resampling step was introduced into the above procedure. In its simplest form the resampled particle filter (or the "interacting particle system" or N-IPS model[2] [12]) assumes the form:

1. **Prediction:** $\tilde{X}_{t+1}^{(i)} = f(X_t^{(i)}) + V_t^{(i)}$, $i = 1, 2, \ldots, N$,
2. **Evaluation:** $w_{t+1}^{(i)} \propto g(Y_{t+1}|X_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$,
3. **Resampling:** Sample $j_{t+1}^{(i)}$ from $(w_{t+1}^{(1)}, w_{t+1}^{(2)}, \ldots, w_{t+1}^{(N)})$ and let $X_{t+1}^{(i)} = \tilde{X}_{t+1}^{(j_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

$X_0^{(1)}, \ldots, X_0^{(N)}$ are assumed to be generated according to a common law, $\pi_0^N = \pi_0$.

A basic result of del-Moral, which will be quoted in its exact form in Section 2.2, concerns the uniform convergence of this filter. The first observation, on which that result is based is that the posterior of $(X_t^{(1)}, \ldots, X_t^{(N)})$,

$$\mu_t^N(A) = P((X_t^{(1)}, \ldots, X_t^{(N)}) \in A|Y_{0:t})$$

follows closely the form of Equation (1.3). Let

$$\mu_0^N(A_1 \times A_2 \times \ldots \times A_N) = \prod_{i=1}^N \pi_0^N(A_i)$$

and

$$\mu_t^N(A_1 \times A_2 \times \ldots \times A_N) = \prod_{i=1}^N \pi_t^N(A_i),$$

[2]As noted by del Moral, the model also incorporates certain genetic algorithms and other randomized optimization methods.

where

$$\pi_t^N(A) = \frac{1}{N}\sum_{i=1^N} \pi_{X_t^{(i)}}(A),$$

and where π_x denotes the counting measure that assigns a unit mass to sets containing x: $\pi_x(A) = \chi_A(x)$ (χ_A denotes the characteristic function of set A). By simple calculations one then gets

$$\pi_{t+1}^N(A) = \frac{\int g(Y_{t+1}|x)K(x,A)d\pi_t^N(x)}{\int g(Y_{t+1}|x)d\pi_t^N(x)},$$

i.e., π_t^N satisfies the filter evolution equation (1.3).

The BIBO result proved by del Moral [12] can roughly be stated as

Theorem 1.4.1. *When the functions $g(\cdot|x)$ and the Markov transition kernels $K(x,\cdot)$ are "sufficiently regular" there exists an exponent $\alpha > 0$ and an index $N_0 > 0$ such that for any bounded measurable function h*

$$\sup_{t\geq 0} E(\,|\int h d\pi_t^N - \int h d\pi_t \,|\, Y_{0:t}\,) = \left(\frac{\|h\|_\infty}{N^\alpha}\right),$$

where $N \geq N_0$ and π_t is the posterior computed by (1.3) and π_t^N is the empirical distribution corresponding to the particle filter.

This theorem assumes perfect knowledge of the system to be filtered. As we argued above, this is an unrealistic assumption. A result of primary interest concerns N-IPS where in the algorithm approximate models are used. We shall prove the following theorem:

Theorem 1.4.2. *Assume that the functions $g(\cdot|x)$ and the Markov transition kernels $K(x,\cdot)$ are "sufficiently regular" and they are approximated with some models $\hat{g}$ and $\hat{K}$, respectively, with tolerance β (in some distance measure measure to be defined later). Then there exists an exponent $\alpha > 0$ and a constant $0 < \tau < 1$ depending only on K such that for any bounded measurable function h*

$$\sup_{t\geq 0} E(|\int h d\hat{\pi}_t^N - \int h d\pi_t \,|\, Y_{0:t}) = \frac{5\exp(2\hat{\gamma}')}{N^{\hat{\alpha}/2}} + \frac{4\beta}{\log(3)(1-\tau)}$$

where $N \geq 1$ and where $\hat{\pi}_t^N$ is the empirical distribution that corresponds to the particles when the approximate model $(\hat{g}, \hat{K})$ is used in the course of updating the particle positions.

1.5 LS-N-IPS: A Sequential Monte-Carlo Method Combined with Local Search

On sequential computers the running-time of Monte-Carlo methods scales linearly with the number of particles[3] In visual tracking the computationally most expensive operation is

[3]Note that Monte-Carlo methods are well parallelizable algorithms in general.

the calculation of the observation densities $g(Y_t|X_t^{(i)})$ since this involves substantial image processing operations. Therefore, the reduction of the number of particles is of great practical interest (i.e., the increase of the efficiency of the filter). Since the quality of the filter depends on the number of particles (see the results of the previous section) it is far from being trivial if such a reduction can be achieved.

Here we propose a method which, according to our experiments under a wide range of conditions performs significantly better than the basic N-IPS algorithm if the number of particles is (relatively) small and is equivalent to the N-IPS algorithm in terms of performance if the number of particles is larger.

The idea of the method is to reduce the time spent on the calculations of observation likelihoods corresponding to particles with a small observation likelihood. Particles that have small observation likelihoods are not likely to "survive" in the resampling time and thus every millisecond spent on the computation of their observation likelihood is likely to be wasted. One can imagine two approaches here, either spend less time on the computation of small observation likelihoods or bias the evolution of the particles such that no particles will have small observation likelihoods. In this work, we pursue this approach, giving up the unbiasedness of the estimate of the posterior. Therefore our method can be interpreted in the usual bias-variance framework: the method reduces the variance at the price of introducing some bias.

The method we propose updates the positions of the particles according to the equation

$$X_{t+1}^{(i)} = S_\lambda(f(X_t^{(i)}) + V_t^{(i)}), Y_{t+1}),$$

where S_λ typically performs local-search in a neighborhood of size λ of $Z_{t+1}^{(i)} = f(X_t^{(i)}) + V_t^{(i)}$ such that $g(Y_t|X_{t+1}^{(i)})$ will be larger than $g(Y_t|Z_{t+1}^{(i)})$. We shall call this method the *local search N-IPS* method, or LS-N-IPS.

As a simple example, aiming to show the improved performance of LS-N-IPS consider the system defined by $X_{t+1} = V_{t+1}, Y_t = X_t + W_t$, where $\mathrm{var}(W_t) << \mathrm{var}(V_t)$. If $S_\lambda(x, y) = y$ (assuming that $g(y|y) = \mathrm{argmax}_x g(y|x)$) then the local search renders all particles to Y, i.e., $X_t^{(i)} = Y$ and thus the estimate of the position of X_t is $\int x d\pi_t^N = Y_t$. Here (and in what follows) π_t^N denotes the estimated posterior corresponding to the particle system $\{X_t^{(i)}\}_{i=1}^N$ of size N.[4] On the other hand the estimate of the N-IPS model is given by

$$\overline{X}_t = \sum_{i=1}^{N} \frac{g(Y_t|V_t^{(i)})}{\sum_{j=1}^N g(Y_t|V_t^{(j)})} V_t^{(i)}.$$

Clearly, under a wide range of conditions $E(|Y_t - X_t|^2 \,|\, X_t) << E(|\overline{X}_t - X_t|^2 \,|\, X_t)$ and in this sense, the estimate of LS-N-IPS is better than that of the N-IPS model. If we assume that V_t and W_t are Gaussian with respective covariances Q and R then the posterior can be computed analitically (using the Kalman-filter euqations), yielding $\overline{X}_t^* = Q(Q + R)^{-1} Y_t$. Therefore X_t^* is close to Y_t provided that $\mathrm{var}(W_t) << \mathrm{var}(V_t)$.

[4] $\pi_t^N(A) = (1/N) \sum_{i=1}^N \chi_A(X_t^{(i)})$, where χ_A is the characteristic function of the measurable set A.

Another interpretation of the observed increased efficiency of LS-N-IPS model comes from considering the effects of using approximate models. As it was noted already, in practice one must use an approximate model of the dynamics. Now imagine that the error of this model is higher than the variance of the noise of the observations. By the same argument as above, LS-N-IPS will help in this case. The local search will act as effectively reducing the error of approximation of the model of the dynamics.

Depending on the implementation of the local search operator S_λ, LS-N-IPS might unwantedly reduce the effective sample size This is not desirable if e.g. the observation allows alternative hypotheses about the state, i.e., when the posterior is multi-modal. A typical implementation would climb locally the function $g(Y_t|\cdot)$. This can be implemented in a number of ways. The parameter $\lambda \geq 0$ controls how much we "trust" the observation model ($\lambda = 0$ meaning that we do not trust in it at all). In later sections we will provide an example for the implementation of S_λ for a visual tracking problem.

LS-N-IPS can also be thought of as a variance reduction technique. In the literature, many variance reduction techniques have been developed, here we mention only the few most relevant ones.

The first method we consider combines importance sampling (IS) and sequential Monte-Carlo filtering. This method is called Sequential Importance Sampling with Resampling (SIR): the proposed states $\tilde{X}_{t+1}^{(i)}$ are sampled from an appropriately chosen proposal distribution that may also depend on the most recent observation Y_{t+1}. The computation of the weights is changed to compensate for the bias introduced by sampling from the proposal instead of the dynamics. The problem with this method is that the proposal is usually quite hard to design - indeed, a large body of work in the current literature is devoted to the design of proposals. The proposal has to satisfy two competing needs: (i) sampling from the proposal should be computationally cheap and (ii) the observation likelihoods should increase on average (hence the variance reduction effect). Sampling from the dynamics is typically very cheap - in visual tracking problems it does not involve any image processing steps. If the proposal is dependant on the most recent observation then sampling becomes a problem. Typically sampling must be accomplished by a general purpose, e.g. a Monte-Carlo Markov-Chain (MCMC) procedure which might require the evaluation of $g(Y_{t+1}|\cdot)$ at several points. Further, the running time of MCMC procedures is random - a disadvantage in tasks where predictable, real-time performance is required.

Another method, closely related to the LS-N-IPS method is the auxiliary variable method (AVM) of Pitt and Shephard [14]. AVM can be thought of as a special case of SIR. It builds on two ideas. The first idea is to let the proposal be of the non-parametric mixture form

$$\sum_{i=1}^{N} \frac{g(Y_{t+1}|\mu_{t+1}^{(i)})}{\sum_{j=1}^{N} g(Y_{t+1}|\mu_{t+1}^{(i)})} f(\cdot|X_t^{(i)}).$$

Here f is the density corresponding to the dynamics of the model (with a slight abuse of notation): $f(x_{t+1}|x_t) = p(X_{t+1} = x_{t+1}|X_t = x_t)$ and $\mu_{t+1}^{(i)}$ is an auxiliary variable: typically $f(X_t^{(i)})$ or the modulus of $f(\cdots|X_t^{(i)})$.

The second idea is to "over-sample" from this proposal, i.e., sample $R >> N$ samples

from it, instead of just sampling N samples. This ensures a more precise representation of the proposal, increasing the overall filtering accuracy. Note that sampling from this proposal does not require an MCMC method provided one can sample from the dynamics of the model.

The last step of AVM is to compute the importance weights of the R samples and finally to resample from these samples to get N samples.

The disadvantage of this method is that in general it requires the calculation of $O(R+N)$ observation likelihoods - a clear disadvantage when real-time performance is required.

Chapter 2

Theoretical Analysis

2.1 A Result on the Robustness of the Bayesian Filter

In this section first definition and basic features are discussed of Hilbert's projective metric, then contraction properties are stated, and finally these results are used for a robustness proof of the Bayesian filter.

Def 1. *(Hilbert Projective Metric) Let $\mu, \nu \in \mathcal{M}(\mathcal{X})$. The Hilbert projective distance between them is defined by*

$$h(\mu,\nu) = \ln \frac{\sup_{A\in\mathcal{M}(\mathcal{X})} \frac{\mu(A)}{\nu(A)}}{\inf_{A'\in\mathcal{M}(\mathcal{X})} \frac{\mu(A')}{\nu(A')}} = \sup_{A\in\mathcal{M}(\mathcal{X})} \ln \frac{\mu(A)}{\nu(A)} + \sup_{A'\in\mathcal{M}(\mathcal{X})} \ln \frac{\mu(A')}{\nu(A')} = \sup_{A,A'\in\mathcal{X}} \ln \frac{\mu(A)}{\nu(A)} \frac{\mu(A')}{\nu(A')}$$

One of the most important properties of Hilbert metric from our point of view is that of scale variance, i.e. for $\mu, \nu \in \mathcal{M}(\mathcal{X})$ and $\alpha, \beta > 0$, $h(\alpha\mu, \beta\nu) = h(\mu, \nu)$. Thus if μ and ν are posteriors arising from the filtering problem, for Hilbert projective metric it makes no difference whether they are normalized or not, and this fact will be convinient for us when proving robustness result.

Note 1. *If the probability measures μ and ν admit density, i.e.: $\int_A f(x)dx = \int_A d\mu(x)$ and $\int_A g(x)dx = \int_A d\nu(x)$, then Hilbert's projective metric can be defined as*

$$h(\mu,\nu) = h(f,g) = \sup_{x,y\in\mathcal{X}} \ln \frac{f(x)}{g(x)} \frac{f(y)}{g(y)}$$

In this case for any positive function u: $h(fu, gu) = h(f, g)$.

Another important property of Hilbert metric which makes it a very useful tool in stability analysis is the following contraction due to Birkhoff. Let K be an integral operator, then

$$h(K\mu, K\nu) \leq \tanh(C(K)/4) h(\mu,\nu)$$

where

$$C(K) = \ln \sup_{x,y,x',y' \in \mathcal{X}} \frac{K(x,y)K(x',y')}{K(x',y)K(x,y')}$$

The following inequality connects the Hilbert metric with the total variation norm:

$$||\mu - \nu||_{TV} \leq \frac{2}{\ln 3} h(\mu, \nu)$$

assuring that if the Hilbert metric of two measure sequence tends to zero, then total variation norm convergence also holds.

2.1.1 Exponential Forgetting

Let us define the forward operator product by

$$T_{1,n} = G_{Y_t} F G_{Y_{t-1}} F \dots G_{Y_0} F$$

It is easy to see that:

$$\pi_{t+1} = \frac{T_{1,t}\pi_0}{(T_{1,t}\pi_0)(\mathcal{X})}$$

Theorem 2.1.1. *Let the dynamical system operator F be a positive kernel integral operator (i.e. $x, y \in \mathcal{X} \Rightarrow K(x,y) > 0$), and the observation operator G_{Y_t} positive operator for all t i.e.: $A \subset \mathcal{X}, \forall y \in \mathcal{Y} : \pi(A) > 0 \Rightarrow (G_y \pi)(A) > 0$. If for all $\pi \in \mathcal{M}(\mathcal{X})$, $F\pi$ admits a density, then the Bayesian filter defined by the model (f, g) forgets its prior ditribution exponentially fast, i.e. for two priors $\pi, \hat{\pi} \in \mathcal{M}(\mathcal{X})$:*

$$\lim_{t \to \infty} h(\pi_t, \hat{\pi}_t) = 0$$

Proof. As we mentioned above normalizing does not effect the Hilbert projective metric, so

$$h(\pi_{t+1}, \hat{\pi}_{t+1}) = h(G_{y_t} F \pi_t, G_{y_t} F \hat{\pi}_t)$$

As $F\pi$ admits a density, using Note 1. leads to

$$h(\pi_{t+1}, \hat{\pi}_{t+1}) = h(F\pi_t, F\hat{\pi}_t)$$

using the contraction properties of Hilbert projective metric with positive kernel operators

$$h(\pi_{t+1}, \hat{\pi}_{t+1}) \leq \tanh(C(K)/4) h(\pi_t, \hat{\pi}_t)$$

which proves the exponential convergence. □

Note 2. *For the above exponential forgetting it is enough to assume that there exists r such that $T_{1,r}$ is positive kernel integral operator, as it is argued in [Brian D.O. Anderson 2000]. In finite dimensionan spaces this assumption is equivalent to ergodicity.*

2.1.2 Robust Filtering

Finally we prove that the above exponential forgetting property means that the filter will be robust to model approximation error.

Let $\hat{F}$ and F be two dynamical operators 'close' enough to each other. We will show that evolving a filter according to F, and another one according to $\hat{F}$ give 'close' outputs. As an application we can use approximate model, still being assured that the filtering outputs are still close to the original ones.

Theorem 2.1.2. *Let the dynamical system operators F and $\hat{F}$ be positive kernel integral operators (i.e. $x, y \in \mathcal{X} \Rightarrow K(x,y) > 0$) absulutely continuous to Lebesgue-measure, and the observation operators G_y and $\hat{G}_y$ be positive for all $y \in \mathcal{Y}$. Let π_0 and $\hat{\pi}_0$ be priors, and*

$$\pi_{t+1} = \frac{G_{Y_{t+1}} F \pi_t}{(G_{Y_{t+1}} F \pi_t)(\mathcal{X})} \qquad \hat{\pi}_{t+1} = \frac{\hat{G}_{Y_{t+1}} \hat{F} \hat{\pi}_t}{(\hat{G}_{Y_{t+1}} \hat{F} \hat{\pi}_t)(\mathcal{X})}$$

If

$$\max\{\sup_{\pi} h(F\pi, \hat{F}\pi), \sup_{\pi, y \in \mathcal{Y}} h(G_y \pi, \hat{G}_y \pi), h(\pi_0, \hat{\pi}_0)\} \leq \epsilon$$

then

$$h(\pi_t, \hat{\pi}_t) \leq \frac{2\epsilon}{1 - \tanh(C(K)/4)}$$

Proof. Similar to to previous proof:

$$h_t = h(\pi_{t+1}, \hat{\pi}_{t+1}) = h(G_{Y_{t+1}} F \pi_t, \hat{G}_{Y_{t+1}} \hat{F} \hat{\pi}_t)$$

Using the triangle inequality:

$$h_t \leq h(\hat{G}_{Y_{t+1}} \hat{F} \hat{\pi}_t, G_{Y_{t+1}} \hat{F} \hat{\pi}_t) + h(G_{Y_{t+1}} \hat{F} \hat{\pi}_t, G_{Y_{t+1}} F \pi_t)$$

As F and $\hat{F}$ is absulutely continuous to Lebesgue measure $F\pi$ and $\hat{F}\pi$ admits density, so using the triangle inequality again leads to:

$$h_t \leq \epsilon + h(\hat{F}\hat{\pi}_t, F\hat{\pi}_t) + h(F\hat{\pi}_t, F\pi_t) \leq 2\epsilon + \tanh(C(K)/4) h(\pi_t, \hat{\pi}_t)$$

Therefore

$$h(\pi_t, \hat{\pi}_t) \leq 2\epsilon + 2\epsilon\tau + 2\epsilon\tau^2 + \ldots \tau^t \epsilon$$

with $\tau = \tanh(C(K)/4) < 1$. Finally:

$$h(\pi_t, \hat{\pi}_t) \leq \frac{2\epsilon}{1 - \tau}$$

□

The theorem can be extended to multi-step ergodic processes.

The above result relies heavily on the mixing properties of the dynamical model, and the reliability of the observations played no role in the analysis. However observations are a critical ingredient of the nonlinear filtering problem, and should play a central role in robustness analysis. There are only few papers in the literature that derive results where the reliability of the observations play role. Here we review a positive result due to Kushner and Budhiraja [2] and then provide several examples that show that weakening the mixing conditions in a meaningful way is a non-trivial task.

The following special systems are studied in [2] . Let the state X_n evolve as before and let the observations satisfy

$$Y_n = X_n + \nu_n$$

meaning that the signal is observed in small additive noise. Further, the noise process W_n is assumed to be bounded in amplitude.

The following "theorem" is proven in [2] :

Theorem 2.1.3. *Assume that the model (f, g) is bounded and sufficiently regular, and let (f_k, g_k) be a model sequence, with supp(g), supp$(g_k) \in [-M, M]$. Assume furthermore that*

$$\lim_{k\to\infty} \sup_{x\in[-M,M]} |\ln g_k(x) - \ln g(x)| = 0$$

Define $\rho_k(.)$ by

$$\rho_k(l) = \sup_{x\in[-l,l]} |f_k(x) - f(x)|$$

and suppose that $E\rho_k(a + b|\psi_1|) \to 0$ as $k \to \infty$ for all positive a, b, where ψ_1 is the error on the prior distribution. Then

$$\lim_{k\to\infty} \limsup_{t\to\infty} E||\pi_t - \pi_t^{(k)}||_{TV} = 0$$

where $\pi_t^{(k)}$ are the posteriors computed using the approximate models (f_k, g_k) and $||\cdot||_{TV}$ is the total variation norm of distributions.

2.1.3 Non-stable Filter Examples

It is well known that if K (or more generally $K^{(r)}$) is uniformly positive than F is contracting. In the previous sections we have seen that exponential forgetting of the Bayesian filter, and thus its stability holds if F is contracting and $G > 0$. In this section we will give some examples showing that these restrictions cannot be weakened easily. For simplicity, in this section we will deal with finite HMMs.

Example 1. Non-forgetting Bayesian filter

Figure 2.1 shows an ergodic Markov chain with $F \geq 0$ for which we will give an observation model such that the resulting Bayesian filter will not forget its prior.

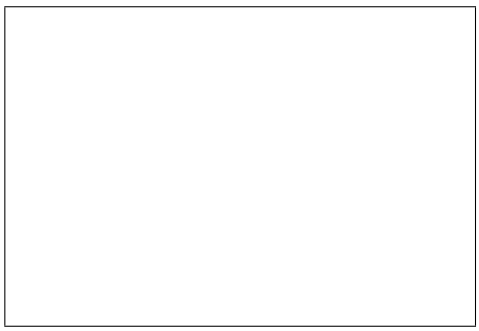

Figure 2.1: *The Markov of Example 1.*

The transition matrix we consider is:

$$F = \begin{pmatrix} 0 & 0 & 0 & \frac{1}{2} \\ 1 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{2} \end{pmatrix}$$

Assume that $\mathcal{Y} = \{y_1, y_2\}$, and let the observation model be given by:

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

i.e.: $G_{y_1} = \mathrm{diag}(1, 0, 1, 0)$, and $G_{y_2} = \mathrm{diag}(0, 1, 0, 1)$.

By some tedious calculations, one can verify that this model does not forget its prior, e.g. take the priors $p_1 = (\frac{1}{3}, 0, \frac{2}{3}, 0)^T, p_2 = (\frac{1}{2}, 0, \frac{1}{2}, 0)^T$.

The example shows that one cannot hope to prove that Bayesian filters will forget their priors in general.

Another interesting detail about the above example comes out when one chooses the observation model as

$$G = \begin{pmatrix} \alpha & 0 & 1-\alpha & 0 \\ 0 & \alpha & 0 & 1-\alpha \\ 1-\alpha & 0 & \alpha & 0 \\ 0 & 1-\alpha & 0 & \alpha \end{pmatrix}$$

where $0 < \alpha < 1$ and $\alpha \neq 0.5$. Let us note here that in general (for non-ergodic chains) forgetting should be required only for priors having the same support. In such a case we will call the priors to be *compatible*. The filter of this model will actually forget the prior exponentially in the above, restricted sense. Note that the posteriors corresponding to the incompatible priors $p_1 = (1, 0, 0, 0)^T, p_2 = (0, 0, 1, 0)^T$ are themselves incompatible.

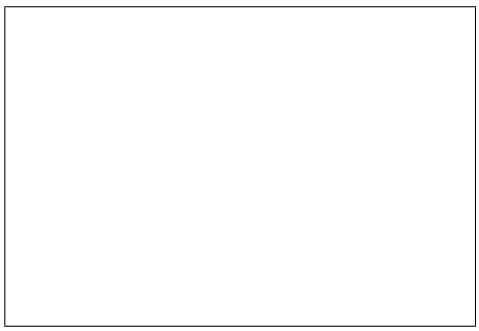

Figure 2.2: *A Markov chain that just forgets with probability one.*

Example 2. A Non-exponentially Forgetting Filter

As shown in the previous section Bayesian filters are not necessarily stable. The question we are going to investigate here is whether a stable filter is also exponentially stable.

Figure 2.2 shows a 3-state ergodic Markov chain with transition matrix

$$F = \begin{pmatrix} 0 & 0 & 1 \\ 1 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 \end{pmatrix}$$

Assume the following observation model:

$$G = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

Assume that the prior is $(1, 0, 0)^T$, and the observations we make are:$Y_0 = 1, Y_1 = 2, Y_2 = 3, Y_3 = 1, \ldots$, ie: $Y_t = (t \mod 3) + 1$.

With this series observations the filter's forgetting rate is linear. This can be seen easily by the following reasoning. At time t the possible states of the system are $((t-1) \mod 3) + 1$ or $(t \mod 3) + 1$. The probabilities of being in these states are

$$P(X_t = (t \mod 3) + 1 | Y_s = (t \mod 3) + 1, s \leq t) \propto \left(\frac{1}{2}\right)^{\frac{t}{3}}$$

and

$$P(X_t = ((t-1) \mod 3) + 1 | Y_s = (t \mod 3) + 1, s \leq t) \propto \sum_{i=1}^{\frac{t}{3}} \left(\frac{1}{2}\right)^{\frac{t}{3}}$$

as one cannot tell when the system has choosen to "stay" in state 2.

It is easy to see that the state $(((t-1) \mod 3) + 1)$ is getting more probable than $((t \mod 3) + 1)$ at a linear rate, i.e. the filter's forgetting rate is linear.

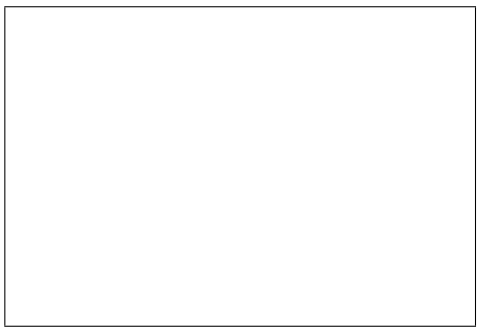

Figure 2.3: *The Markov chain of Exmaple 4. The states are numbered from top to down.*

It is important to note that the infinite limit of the observation series is not in the support of the limit posterior. The example thus shows that any reasonable exponential stability can only be proven with probability one and only asymptotically.

Example 3. A Locally Similar Example

An even more interesting Markov chain is shown in Figure 2.3. The corresponding observation matrix is

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Since the probability of a transition from $\{1, 2\}$ to $\{3, 4\}$ and vica versa is small, if the priors have supports $\{1, 2\}$ and $\{3, 4\}$ then the distance of the respective posteriors will only decrease slowly, as shown in Figure 2.4.

Relative Forgetting

Obviously, if the product operators

$$T_{y_1,\dots,y_k}\pi = \frac{G_{y_k}F \dots G_{y_1}F\pi}{||G_{y_k}F \dots G_{y_1}F\pi||_1}$$

are contractions with a common contraction coefficient, in some norm then the exponentially stability results will continue to hold. One such norm, that is also comfortable to work with, is the induced $\mathcal{L}_1$ norm:

$$||T||_1 = \sup_{x \in \mathcal{M}(\mathcal{X})} \frac{||T\pi||_1}{||\pi||_1}$$

Let us call a filter multi step contracting if for some $k > 0$ $T_{y_1,\dots,y_k}$ is contracting for all $y_1, \dots, y_k \in \mathcal{Y}$.

Here we will show that for Example 2 above (cf. Figure 2.2), the operator of the corresponding filter is not multi-step contracting. To prove this we compute the product

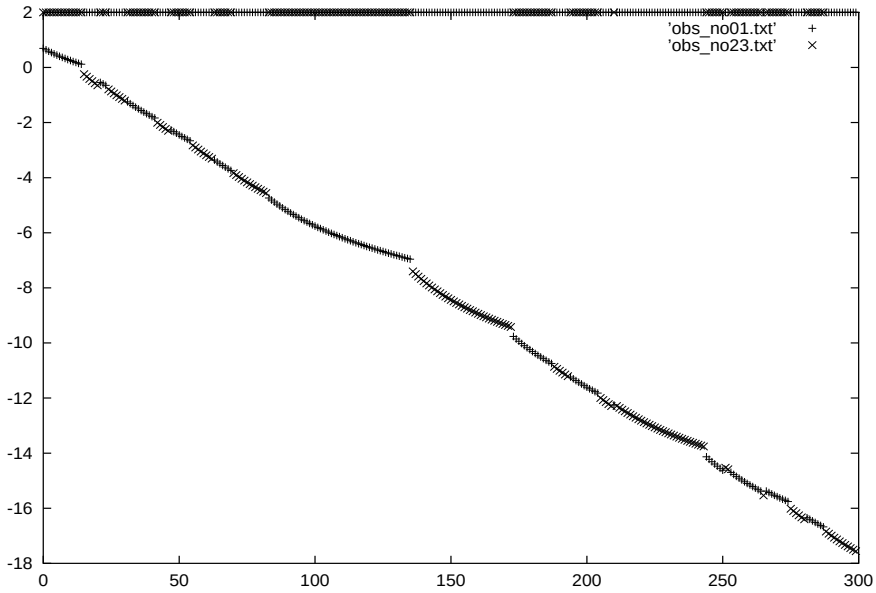

Figure 2.4: *Simulation results for Markov chain shown in figure 3. Linear forgetting segments can be seen.(see text)*

matrices. The transition matrix is:

$$F = \begin{pmatrix} 0 & 0 & 1 \\ 1 & \alpha & 0 \\ 0 & 1-\alpha & 0 \end{pmatrix}$$

where $\alpha = \frac{1}{2}$. By inspection one might verify that

$$G_{Y_3} F G_{Y_2} F G_{Y_1} F = \begin{pmatrix} 1 & \alpha & 0 \\ 0 & 0 & 0 \\ \alpha & \alpha^2 & 1 \end{pmatrix}$$

Therefore $T_{3k} = G_{Y_{3k}} F G_{Y_{3k-1}} F \ldots G_{Y_1} F = (G_{Y_3} F G_{Y_2} F G_{Y_1} F)^k$. By induction we get:

$$T_{3k} = \begin{pmatrix} 1 & \alpha & 0 \\ 0 & 0 & 0 \\ k\alpha & k\alpha^2 & 1 \end{pmatrix}$$

If the prior is

$$p(\beta) = \begin{pmatrix} \beta \\ 0 \\ 1-\beta \end{pmatrix}$$

then the posterior at time $3k$ is

$$p_{3k}(\beta) = \begin{pmatrix} \frac{\beta}{1+\beta k\alpha} \\ 0 \\ \frac{\beta k\alpha + 1 - \beta}{1+\beta k\alpha} \end{pmatrix}$$

Clearly,

$$\tau(\beta_1, \beta_2) = \frac{||T_{3k}p(\beta_1) - T_{3k}p(\beta_2)||_1}{||p(\beta_1) - p(\beta_2)||_1} = \frac{\left|\frac{\beta_1}{1+\beta_1 k\alpha} - \frac{\beta_2}{1+\beta_2 k\alpha}\right|}{|\beta_1 - \beta_2|} = \frac{1}{(1+\beta_1 k\alpha)(1+\beta_2 k\alpha)}$$

Now, if $\beta_1 = \frac{1}{n}$ and $\beta_2 = \frac{2}{n}$ then

$$\tau(\beta_1, \beta_2) = \frac{1}{(1+\frac{k}{n}\alpha)(1+\frac{2k}{n}\alpha)}$$

which, for any fixed k can made as close to 1 as desired, showing that the filter is not multi-step contracting.

Example 4. Non-stationary observation models

Assume that the observation density is non stationary. Let the Markov chain be the same as shown in Figure 2.1, and let the observation density at time t be

$$G_t = \begin{pmatrix} 1-\frac{1}{2^t} & 0 & 1-\frac{1}{2^t} & 0 \\ 0 & 1-\frac{1}{2^t} & 0 & 1-\frac{1}{2^t} \\ \frac{1}{2^t} & \frac{1}{2^t} & \frac{1}{2^t} & \frac{1}{2^t} \end{pmatrix}$$

The expected forgetting rate of simulating this filter is shown in Figure 2.5. It should be clear from the figure that the rate of forgetting is at most linear.

Example 5. A "totally" forgetting filter

The next example highlights an interesting case when the posterior becomes independent of the prior in a single step: the posterior contracts to a single point. We shall call HMMs of this type are "totally forgetting". The Markov chain underlying the example is depicted in Figure 2.6, the observation model is

$$G = \begin{pmatrix} \frac{1}{2} & \cdots & \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & \cdots & \frac{1}{2} & 0 & 1 & 1 \end{pmatrix}$$

This example shows that under general conditions it is not possible to derive non-asymptotic forgetting rates.

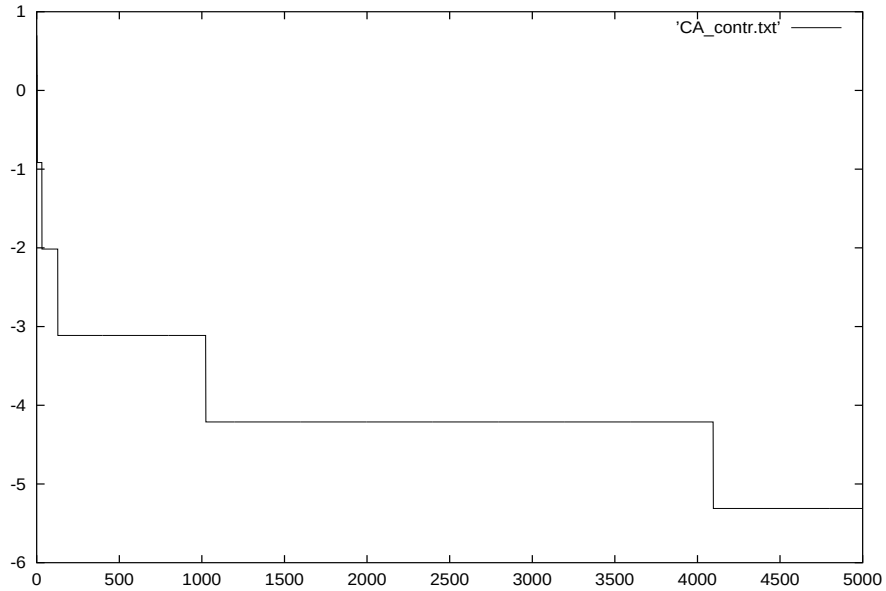

Figure 2.5: *Expected forgetting rate of a filter with non-stationary observation.*

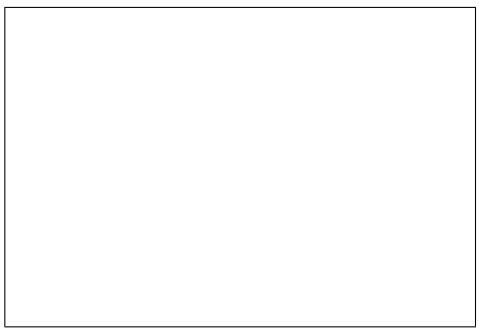

Figure 2.6: *Example showing total forgetting.*

2.2 Uniform Convergence of N-IPS

Here we revisit some results of del Moral [12] that will be needed in the next section. First we need to introduce some notation. Let $(\mathcal{X}, \Omega)$ be a measure space. For a measure $\mu \in M(\mathcal{X})$ let

$$\|\mu\|_{TV} = \sup_{A\in\Omega} |\mu(A)|$$

be its total variation norm. Further, let us interpret the substraction of measures in the usual way: $(\mu - \nu)(A) = \mu(A) - \nu(A)$.

Let K be a Markov transition kernel on $\mathcal{X}$. Recall that the ergodic coefficient of K is the quantity $\alpha(K) \in [0, 1]$ given by

$$\alpha(K) = 1 - \beta(K), \quad \text{with} \quad \beta(K) = \sup_{x,y\in\mathcal{X},A\in\Omega} |K(x, A) - K(y, A)|.$$

We shall call the number $\alpha(K)$ the Dobrushin ergodic coefficient of K (see [4]).

The quantity $\alpha(K)$ is a measure of contraction of the distance of probability measures induced by the Markov operator $F = F_K$, corresponding to K. Namely, we have the well known formula (see [4])

$$\beta(K) = \sup_{\mu,\nu\in M(\mathcal{X})} \frac{\|F_K\mu - F_K\nu\|_{TV}}{\|\mu - \nu\|_{TV}}. \tag{2.1}$$

Let us consider the model (1.1)-(1.2). Assume that the observation likelihood function $g(y|x)$ is such that for some $a > 0$

$$\frac{1}{a} \leq g(y|x) \leq a \tag{2.2}$$

holds. Assume that $K(x, \cdot) \sim m$, where m is the Lebesgue-measure and let (with a slight abuse of notation)

$$K(x, z) = \frac{dK(x, \cdot)}{dm}(z)$$

be the Radon-Nikodym derivative of K. Assume that for all (x, z), $K(x, z)$ satisfies

$$\varepsilon \leq K(x, z) \leq \frac{1}{\varepsilon} \tag{2.3}$$

for some $\epsilon > 0$. Then, the following theorem holds:

Theorem 2.2.1. *Under the conditions stated in this section the uniform bounds*

$$\sup_{t\geq 0} E(\,|\int f d\pi_t^N - \int f d\pi_t||Y_{0:t}\,) \leq \frac{5\exp(2\gamma')}{N^{\alpha/2}} \tag{2.4}$$

hold for any a measurable function f satisfying $\|f\|_\infty \leq 1$, where α and γ' are given by

$$\alpha = \frac{\varepsilon^2}{\varepsilon^2 + \gamma'} \quad \text{with} \quad \gamma' = 1 + 2\log a.$$

The proof combines Theorem 3.1 and Corollary 2.9 of [12].

2.3 Uniform Convergence with Approximate Models

In this section we consider the N-IPS model applied to approximate models, $\hat{g}(y|x)$ and $\hat{K}(x,\cdot)$.

Theorem 2.3.1. *Let $\hat{g}, \hat{K}$ be approximate models and $\hat{\pi}_0$ be an approximation of the prior π_0. Assume that the Radon-Nikodym derivative of both K and $\hat{K}$ exists w.r.t. the Lebesgue-measure and let us denote them by $K(x,z)$ and $\hat{K}(x,z)$, respectively. Assume that for some $\hat{\varepsilon} > 0$ and $\hat{a} > 0$*

$$\frac{1}{\hat{a}} \leq \hat{g}(y|x), g(y|x) \leq \hat{a} \tag{2.5}$$

and

$$\hat{\varepsilon} \leq \hat{K}(x,z), K(x,z) \leq \frac{1}{\hat{\varepsilon}}. \tag{2.6}$$

Further, assume that for some $\beta > 0$ these approximate entities satisfy

$$h(G_y, \hat{G}_y), h(F, \hat{F}), h(\pi_0, \hat{\pi}) \leq \beta,$$

where h is the Hilbert projective metrics. Consider the N-IPS model based on the approximate models and the approximate prior. Let the empirical posterior at time t, as computed by the N-IPS model, be $\hat{\pi}_t^N$. Then the following uniform bounds

$$\sup_{t\geq 0} E(\,|\int f d\hat{\pi}_t^N - \int f d\pi_t|\,|\,Y_{0:t}\,) \leq \frac{5\exp(2\hat{\gamma}')}{N^{\hat{\alpha}/2}} + \frac{4\beta}{\log(3)(1-\tanh(C(K)/4)} \tag{2.7}$$

hold for any measurable function f satisfying $\|f\|_\infty \leq 1$, and where $\hat{\gamma}'$ and $\hat{\alpha}$ are defined by

$$\hat{\alpha} = \frac{\hat{\varepsilon}^2}{\hat{\varepsilon}^2 + \hat{\gamma}'} \quad \text{with} \quad \hat{\gamma}' = 1 + 2\log\hat{a}.$$

Proof. By the triangle inequality,

$$\begin{aligned}&\sup_{t\geq 0} E(|\int f d\hat{\pi}_t^N - \int f d\pi_t|\,|Y_{0:t}) \leq \\ &\sup_{t\geq 0} E(|\int f d\hat{\pi}_t^N - \int f d\hat{\pi}_t|\,|Y_{0:t}) + \sup_{t\geq 0} E(|\int f d\hat{\pi}_t - \int f d\pi_t|\,|Y_{0:t}),\end{aligned} \tag{2.8}$$

where $\hat{\pi}_t$ is defined by

$$\hat{\pi}_{t+1} = \frac{\hat{G}_{Y_t}\hat{F}\hat{\pi}_t}{(\hat{G}_{Y_t}\hat{F}\hat{\pi}_t)(\mathcal{X})}.$$

The previous theorem continues to hold for $\hat{\pi}_t$ and $\hat{\pi}_t^N$ because of the positivity assumptions made in Section 2.2. Therefore, the first term of (2.8) can be bounded by $\frac{5\exp(2\hat{\gamma}')}{N^{\hat{\alpha}/2}}$.

By Theorem 2.1.1

$$h(\hat{\pi}_t, \pi_t) \leq \frac{2\beta}{1-\tanh(C(K)/4)}$$

Since, by the well known inequality (see [4])

$$\|\hat{\pi}_t - \pi_t\|_{TV} \leq \frac{2}{\log 3} h(\hat{\pi}_t, \pi_t)$$

and $\|\hat{\pi}_t - \pi_t\|_1 \leq \|\hat{\pi}_t - \pi_t\|_{TV}$, we arrive at

$$|\int f d\hat{\pi}_t - \int f d\pi_t| \leq \|f\|_\infty \|\hat{\pi}_t - \pi_t\|_1 \leq \frac{4\beta\|f\|_\infty}{\log(3)(1 - \tanh(C(K)/4))}.$$

Taking the expectation of both sides and combining this inequality with that of derived above for the first term yields the result. □

Chapter 3

LS-N-IPS

In this chapter we propose a modification of the N-IPS algorithm, which we shall call LS-N-IPS (local search+N-IPS). LS-N-IPS is intended to improve the efficiency of N-IPS under the assumption that the observations are sufficiently "reliable". Roughly, this assumption means that the conditional variance of the observation noise given the past states is smaller than that of the dynamics.

It is well known that under this assumption N-IPS does not perform very well since most of the particles will bear a small uniform likelihood owning to the sensitivity (peakiness) of the observation density. Therefore, a large number of particles is needed to achieve even a moderate precision.

We believe that the "reliability assumption" holds in many practical cases and is worth of being studied. In particular, we provide examples of simulated scenarios with this property, and qualitative measurements are made to show that LS-N-IPS is significantly more effective than N-IPS. In the next chapter we demonstrate that LS-N-IPS significantly outperforms N-IPS in a real world visual tracking domain.

3.1 The LS-N-IPS algorithm

The algorithm is as follows:

1. **Initialization:** Let $X_0^{(i)} \sim \pi_0$, $i = 1, 2, \ldots, N$ and set $t = 0$.
2. Repeat forever:
   (a) **Prediction:** Compute the proposed next states by $Z_{t+1}^{(i)} = S_\lambda(f(X_t^{(i)}) + W_t^{(i)}, Y_{t+1})$, $i = 1, 2, \ldots, N$.
   (b) **Evaluation:** Compute $w_{t+1}^{(i)} \propto g(Y_{t+1}|Z_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$.
   (c) **Resampling:**
      i. Sample $k_{t+1}^{(i)} \sim (w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N)})$, $i = 1, 2, \ldots, N$.
      ii. Let $X_{t+1}^{(i)} = Z_{t+1}^{(k_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

(d) t:=t+1

As it should be clear that the only difference between LS-N-IPS and N-IPS is the way they update the positions of the particles. LS-N-IPS uses a non-trivial local search operator, S_λ, to "refine" the "crude" predictions $f(X_t^{(i)}) + W_t^{(i)}$, whilst N-IPS uses the operator $S_\lambda(x, y) = x$.

One should think of S_λ as a local search for refining a particle state such that the observation likelihood given the current observation Y_{t+1} is maximized. The purpose of this is to make the particles become more "relevant".

In general, the requirement for S_λ is that $g(y|S_\lambda(x, y)) \geq g(y|x)$ should hold. The parameter $\lambda > 0$ defines the "search length". Typically, $S_\lambda(y|x) \approx \text{argmax}\{g(y|\tilde{x}) \,|\, \|\tilde{x} - x\| \leq \lambda\}$. For densities that have simple forms the result of the search can be computed analytically. For more complicated densities, g can be approximated locally e.g. by fitting a second order surface on it and then the search can be performed on the approximated surface.

3.2 The advantage of LS-N-IPS over N-IPS

Since LS-N-IPS can be viewed as an N-IPS algorithm where in each step an approximate dynamics is used in the prediction step, Theorem 2.3.1 readily states the stability of LS-N-IPS, i.e., that the tracking error can be kept bounded and reduced (to a limit) by increasing the number of particles. In this section we will show (using some simulated scenarios) that LS-N-IPS can indeed improve on the performance on N-IPS under the special conditions discussed earlier.

3.2.1 A simple example

Consider the linear filtering problem given by the equations $X_t = W_t$ and $Y_t = X_t + V_t$, where W_t and V_t are i.i.d. Gaussian random variables with $W_t \sim \mathcal{N}(0, 1)$ and $V_t \sim \mathcal{N}(0, 0.1)$.

For this special system, the Kalman-filter estimate is easy to compute in an exact form and gives $X_t^*(= E[X_t|Y_{0:t}]) = 1/(1 + 0.1)Y_t$. Thus $E[(X_t^* - X_t)^2 \,|\, X_t] \approx 0.008264X_t^2 + 0.08264$.

Consider LS-N-IPS with $S_\lambda(z, y) = (1 - \lambda)z + \lambda y$, where we restrict λ to $[0, 1]$. The estimate of X_t as given by LS-N-IPS is $\hat{X}_t^*(\lambda) = (\sum_{i=1}^N g(Y_t|X_t^{(i)})X_t^{(i)})/\sum_{j=1}^N g(Y_t|X_t^{(j)})$. For $\lambda = 1$ we get $\hat{X}_t^*(1) = Y_t$ and thus $E[(\hat{X}_t^* - X_t)^2 \,|\, X_t] = 0.1$. Note that if $|X_t|^2 > 0.021$ then LS-N-IPS gives better results with the performance measure $V(x; X_t) = E[(x - X_t)^2|X_t]$ than the Kalman filter (no wonder, though, since the Kalman filter minizes $U(x; Y_{0:t}) = E[(x - X_t)^2|Y_{0:t}]$). In fact, the estimate $\hat{X}_t^*(1) = Y_t$ is the only linear statistics of the past observations that renders V independent of X_t. In this sense this is the best robust linear estimate of X_t.

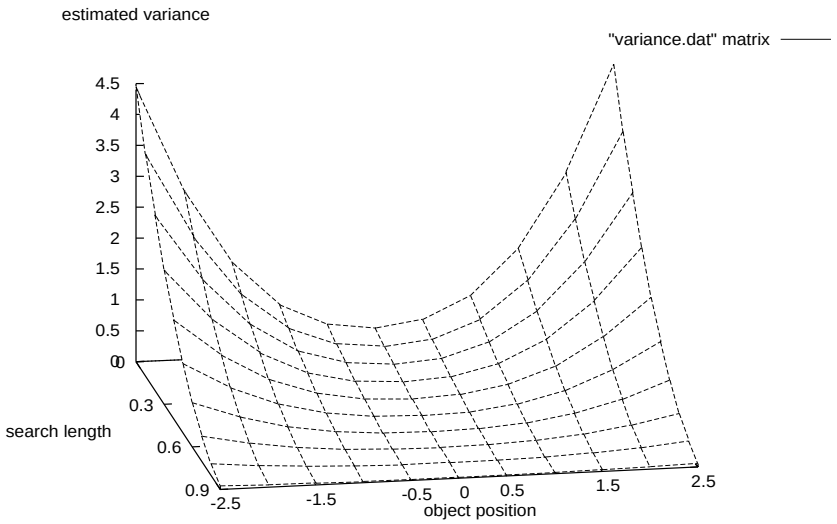

Figure 3.1: *The eastimated variance of LS-N-IPS with* 2 *particles with respect to object position and search length.*

If $\lambda = 0$ then LS-N-IPS gives the estimate that would be given by N-IPS. Clearly, if X_t takes on an extreme value then for small sample sizes we get $w_t^{(i)} \approx 1/N$ and the estimate becomes approximately equal to $\tilde{X}_t^* = (1/N)\sum_{i=1}^N X_t^{(i)}$. Therefore $V(\hat{X}_t^*(0); X_t) \approx X_t^2 + \mathrm{Var}(\tilde{X}_t^*) = X_t^2 + 1$. Clearly, this is much worse than 0.1.

In order to confirm these findings, we have run some Monte-Carlo simulations to compute $V(X_t^*(\lambda); X_t)$ for different values of X_t. Results of the simulations are shown in Figure 3.1 as a function of X_t and λ (1000 observations were drawn for each case, and for each observation $X_t^*(\lambda)$ was computed 1000 times. The resulting variance estimates were averaged over these 10^6 runs.) The number of particles was choosen to be 2, but similar figures can be obtained for bigger values of N except that as N becomes bigger the variance of the estimate approaches the limit function derived for the Kalman filter. It should be clear from the figure that LS-N-IPS performs significantly better than N-IPS, especially in the case of large values of X_t (X_t is shown on the (x) horizontal axis, $1 - \lambda$ is shown on the y axis, and the the z axis corresponds to the variance estimates V).

3.2.2 Results on a complex system

The dynamics we consider is as follows:

$$\begin{aligned}
\hat{X}_{t+1} &= X_t + S_{t+1}\Delta_t + W_t \\
S_{t+1} &= (2B_{t+1} - 1)S_t \\
U_{t+1} &= \chi(|\hat{X}_{t+1}| \leq K) \\
X_{t+1} &= U_{t+1}\hat{X}_{t+1} + (1 - U_{t+1})X_t \\
\Delta_{t+1} &= U_{t+1}(X_{t+1} - X_t) + \\
&\quad (1 - U_{t+1})(X_{t+1} - \hat{X}_{t+1}),
\end{aligned}$$

where $W_t \sim \mathcal{N}(0, \sigma)$ are i.i.d. Gaussian random variables, and B_t is a Bernoulli variable with parameter α. The dynamics can be thought to model a "bouncing ball"(with no mass), where the ball is bounced at the points $-K, +K$ and at random time instances (when $B_t = 0$).

The observation is $Y_t = X_t + V_t$ where $V_t \sim \mathcal{N}(0, \delta)$ i.i.d. The dynamics is highly non-linear, so linear filters are out of question[1]. In the experiments we used $\alpha = 0.99$ (low probability of bounce), and $\sigma = 10\delta$ (uncertainity of dynamics is higher than that of the observation), $\delta = 0.5, K = 250$.

We tested both N-IPS and LS-N-IPS on this problem. Since we are interested in the performance when the number of particles is small we used $N = 10$. Time to time, both N-IPS and LS-N-IPS loose the object, i.e., $D_t = |\frac{1}{N}\sum_i X_n^{(i)} - X_n)| > \theta$ (we used $\theta = 25$). In order to make a quantitative comparison we ran 10^4 simulations of length $t = 400$ each and for each $T \in \{1, 2, \ldots, 400\}$ estimated the probability of loosing the object at time T for the first time. Results are shown in Figure 3.2. It should be clear from the figure that LS-N-IPS performs much better than N-IPS. The SEARCH-ONLY algorithm, also shown in the picture, is an LS-N-IPS algorithm with "zero" dynamics, i.e. $\hat{Z}_{t+1}^{(i)} = S_\lambda(X_t^{(i)}, Y_{t+1})$. This algorithm performs much worse than either of the two other ones, underlining the importance of the role of the dynamics in particle filtering.

Figure 3.3 shows the tracking precision of the same algorithms as a function of the time. More precisely, the figure shows the estimates of $E[D_t | D_t \leq \theta]$ as a function of t, as estimated by computing the averages over the 10^4 runs. The ordering of the algorithms is the same as for the previous measurements: LS-N-IPS still performs significantly better than the other two algorithms.

[1]We have run some simulation with a Kamlan filter for the model that disregards the non-linear factors of the dynamics. The simulation has shown high sensitivity-the Kalman filter lost the object upon the first 'bounce'

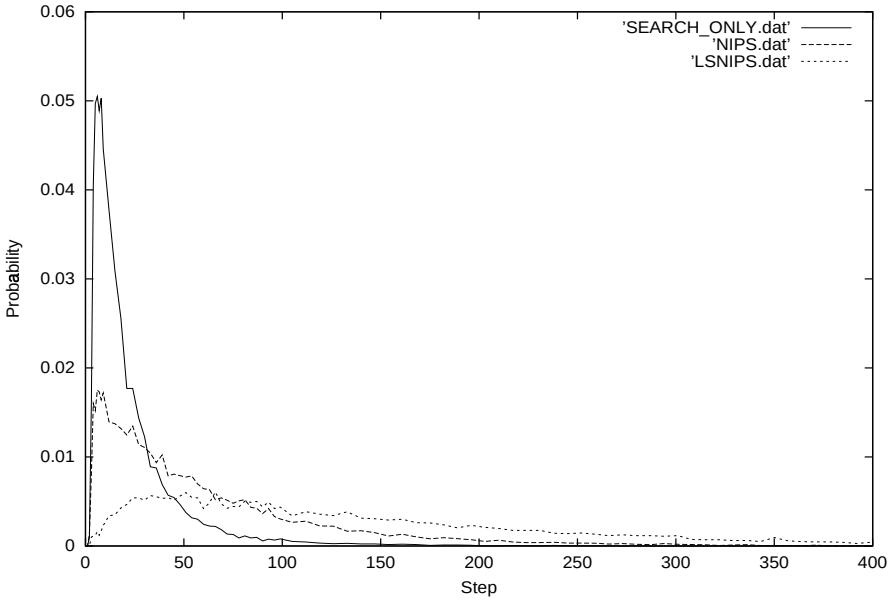

Figure 3.2: *Probability of loosing the object as a function of time.*

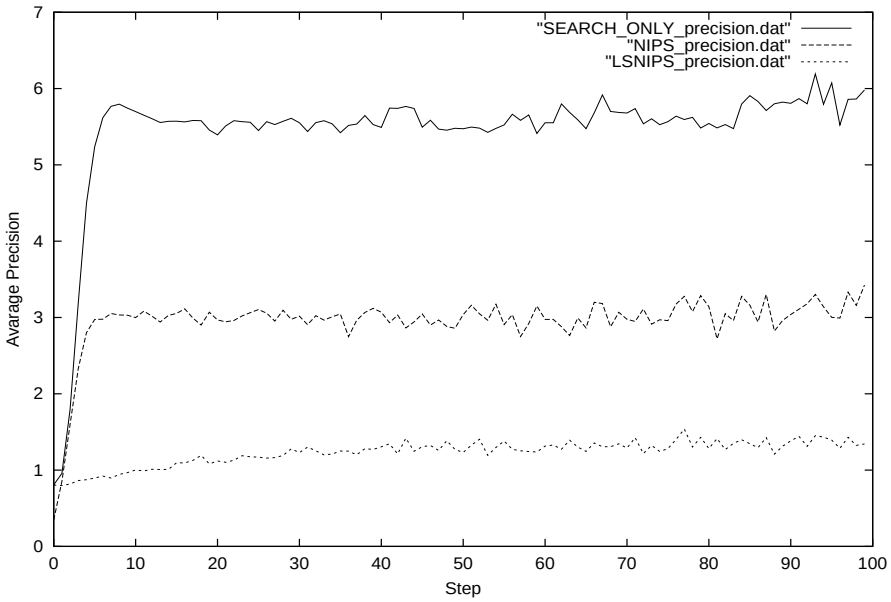

Figure 3.3: *Precision of tracking, provided that the object is not lost, in terms of time steps.*

Chapter 4

Applications to Visual Tracking

4.1 Introduction

In this chapter we will deal with the problem of tracking objects based on image sequences.

In the first section we will examine in details spline contour models as these will serve as the basis of our observation models. Spline contour models have been used succesfully for object tracking in the prominent work of Isard and Blake [1]. Here we introduce regular B-spline interpolation, and suggest an algorithm that leads to efficiently computable splines. These spline contours has some nice features, such as being free of hooks and having smooth interpolating contours.

In the following section a possible observation model is given for object contour tracking. It is also shown that local search can be performed efficiently in this framework.

Finally experimental results are given. We will compare the baseline N-IPS model Bayesian filtering (also known as CONDENSATION algorithm [7],[8]), with the LS-N-IPS model. These experiments clearly indicate that LS-N-IPS works very well when the number of particles is small and eventually yields a computationally efficient algorithm.

4.2 Spline Contours

The purpose of this section is to examine some features of regular B-spline algorithms, and to propose some modifications, to enable us to use spline interpolations in a computationally efficient way.

4.2.1 Regular B-spline Theorems

For simplicity let us consider cubic B-splines only. We believe that our ideas are applicable to B-splines of any order.

Def 2. *A function $s : [0, l] \to \mathcal{R}$, where l is a natural number, is called a cubic spline-function if it is a cubic polynomial over every intervalof the form $[i, i + 1]$, and is twice continuously differentiable.*

A word about the parameter l. Usually it is reasonable for computational and notational purposes that each polinom, which is part of the spline-function is parameterized from 0 to 1. This means that if we have l pieces of these polynoms forming the spline-function than the spline-function itself will be naturally parameterized from 0 to l.

Def 3. *An n-dimensional spline-function is a function $S : [0, l] \to \mathcal{R}^n$ whose coordinate functions are all spline-functions.*

Our aim is to find cubic spline-functions fitting that interpolate a number of points. These points will be called the support points of the spline. It follows from the definition above that it is sufficient to deal with one dimensional cubic spline-functions.

We will distinguish between open and closed spline functions. A closed spline-function satisfies $s(0) = s(l)$. Splines can be given as the linear combination of the translations of single basis function. A closed spline has the form

$$s(x) = \sum_{i=0}^{n} p_i \varphi_{i-2}(x)$$

where φ_i is the i-th spline basis function, and $p_0, p_1, \ldots, p_n$ are the so-called *control points*. In the cubic case

$$\varphi_0(x) = \left\{ \begin{array}{ll} \frac{1}{6}x^3, & \text{if } x \in [0,1) \\ -\frac{1}{2}x^3 + 2x^2 - 2x + \frac{2}{3}, & \text{if } x \in [1,2) \\ \frac{1}{2}x^3 - 4x^2 + 10x - \frac{22}{3}, & \text{if } x \in [2,3) \\ -\frac{1}{6}x^3 + 2x^2 - 8x + \frac{32}{3}, & \text{if } x \in [3,4] \\ 0, & \text{otherwise} \end{array} \right)$$

and

$$\varphi_i(x) = \varphi_0(x - i), \qquad i > 0$$

In the above formulas modulo l arithmetic was assumed.

Let us define the i-th *support point* q_i by $q_i = s(i)$.

Proposition 1. *Let s be a closed cubic spline-function defined by the control points $p_0, p_1, \ldots, p_n$. Then*

$$q_i = \frac{1}{6}p_{i-1} + \frac{2}{3}p_i + \frac{1}{6}p_{i+1}, \quad i = 0, \ldots, n$$

where the indexes are treated mod l.

Proof. Observe that

$$\varphi_j(k) = \varphi_l(h) \Leftrightarrow (h - l) \equiv (k - j)$$

This follows directly from $\varphi_i(x) = \varphi_0(x - i)$. Now

$$q_i = s(i) = p_{i-1}\varphi_{i-3}(i) + p_i\varphi_{i-2}(i) + p_{i+1}\varphi_{i-1}(i) =$$

$$= p_{i-1}\varphi_0(3) + p_i\varphi_0(2) + p_{i+1}\varphi_0(1)$$

yielding the desired result. □

Proposition 2. *Let s be a cubic spline-function given by the control points $p_0, p_1, \ldots, p_n$ and let s' be the derivative of s. Then*

$$s'(i) = \frac{p_{i+1} - p_{i-1}}{2}$$

Proof. Similar to the previous proof we have

$$s'(i) = p_{i-1}\varphi_0'(3) + p_i\varphi_0'(2) + p_{i+1}\varphi_0'(1)$$

By substitution we get the result. □

4.2.2 Control Points versus Support Points

In the previous section we have seen that in the case of closed spline-functions, control points can be converted to support points by multiplying the vector $(p_0, p_1, \ldots, p_n)^T$ by the matrix

$$A = \frac{1}{6}\begin{pmatrix} 4 & 1 & 0 & \ldots & 0 & 1 \\ 1 & 4 & 1 & 0 & \ldots & 0 \\ \vdots & & & \ddots & & \vdots \\ 1 & 0 & \ldots & 0 & 1 & 4 \end{pmatrix}$$

Now, in order to compute control points p_i given the support points q_i, we want to compute $A^{-1}q$, provided that A^{-1} exists. However, we would like to compute A^{-1} explicitly and also we are interested in the asymptotic behavior of A^{-1} as n goes infinity. In particular for computational efficiency, we would like to replace near-zero elements of A^{-1} with zero. We would like to answer two questions then: how many elements can be set to zero if we fix some bound ϵ and how big the resulting approximation error will be.

In the followings we will prove some more general theorems about the inverse of circulant matrixes. These theorems will imply fast computation of the inverse of the above matrix, and an even faster approximation of it.

Def 4. *A is a circulant matrix, if*

$$A = \begin{pmatrix} a_1 & a_2 & \ldots & a_n \\ a_n & a_1 & \ldots & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \ldots & a_1 \end{pmatrix}$$

We will use $circ(a_1, a_2, \ldots, a_n)$ to denote A.

Proposition 3. *Let $A = circ(a_1, a_2, \ldots, a_n)$. Then, provided that A^{-1} exists, it is also circulant and the vector $b \in \mathcal{R}^n$ for which $A^{-1} = circ(b)$ satisfies $b^T S_r^i a = \delta_{i0}, i = 0, \ldots, n-1$.*

Proof. Observe that if $a = (a_1, a_n, a_{n-1}, \ldots, a_2)^T$, then $A = [a, S_r a, S_r^2 a, \ldots, S_r^{n-1} a]$, where S_r is the right-shift matrix, i.e: $S_r = circ(0, 0, \ldots, 0, 1)$.

Let $B = A^{-1}$, $B = [b_1^T, \ldots, b_n^T]^T$. As $BA = I$, we have: $b^T a = 1$ and $b^T S_b^i a = 0$ if $i = 1, \ldots, n-1$. Since $S_r^i = S_r^j$ provided that $i \equiv j \pmod{n}$, clearly:

$$(b^T S_r^{n-i}) S_r^j a = \delta_{ij}$$

where δ_{ij} is the Kronecker-delta.

Therefore $circ(b)$ is the left inverse of A and thus (since left inverses are unique) $B = circ(b)$. □

Proposition 4. *If A is an invertible, circulant and symmetric matrix, and*

$$A^{-1} = circ(a_1, a_2, \ldots a_n)$$

then $a_i = a_{n+2-i}$

Proof. If A is symmetric its inverse is also symmetric and, as it was shown in the previous theorem, if A is circulant its inverse is also circulant. From this, the result follows directly. □

Theorem 4.2.1. *Fix $n \in \mathcal{N}$, let $a = (a, 1, 0, \ldots, 0, 1)^T$ and $A = circ(a) \in \mathcal{R}^{n \times n}$. Assume that $B \in \mathcal{R}^{n \times n}$ is such that $AB = cI$ and $b = circ(b_1, b_2, \ldots, b_n)^T, B = circ(b)$ and $b_n = b_2$. Let $\hat{B}$ be defined by*

$$\hat{b} = circ(-(ab_1 + b_2), b_1, b_2, \ldots, b_n, b_1)^T, \hat{B} = circ(\hat{b})$$

and let $\hat{A}$ be defined by

$$\hat{A} = (a, 1, 0, \ldots, 0, 1) = circ(\hat{a}) \in \mathcal{R}^{(n+2)x(n+2)}$$

Then

$$\hat{A}\hat{B} = (a\hat{b}_1 + 2\hat{b}_2)I = (2b_1 - a^2 b_1 - ab_2)I$$

Proof. According to Proposition 3, all we need to prove is

$$\hat{b}^T S_r^i \hat{a} = (2b_1 - a^2 b_1 - ab_2)\delta_{i0}, \quad i = 0, \ldots, n+1$$

As $S_r^T = S_l$ this is clearly equivalent to

$$(S_l^i \hat{b})^T \hat{a} = (2b_1 - a^2 b_1 - ab_2)\delta_{i0}, \quad i = 0, \ldots, n+1$$

For $i = 0$ this gives

$$\hat{b}^T\hat{a} = a(-ab_1 - b_2) + 2b_1 = 2b_1 - a^2b_1 - ab_2$$

For $i = 1$, we have

$$(S_l\hat{b})^T\hat{a} = ab_1 - (ab_1 + b_2) + b_n = ab_1 - (ab_1 + b_2) + b_2 = 0$$

When $2 \leq i \leq n$, the first and last elements of $\hat{b}$ are shifted to some position where they are multiplied by some zero elements of $\hat{a}$. Since $\hat{a}$ is augmented with two zeroes as compared to a one gets by inspection that

$$(S_l^i\hat{b})^T\hat{a} = (S_l^{i-1}b)^Ta = 0$$

Here the last equality is obtained from $AB = cI$ and Proposition 3.
For $i = n + 1$, we have

$$(S_l^{n+1}\hat{b})^T\hat{a} = ab_1 - (ab_1 + b_2) + b_n = ab_1 - (ab_1 + b_2) + b_2 = 0$$

concluding the proof. □

Corollary 1. *Let* $A = (a_1, a_2, 0, \ldots, 0, a_2) \in \mathcal{R}^{n\times n}; a_2 \neq 0,$
$B = circ(b_1, b_2, \ldots, b_n); AB = cI, c \neq 0,$
$\hat{A} = (a_1, a_2, 0, \ldots, 0, a_2) \in \mathcal{R}^{n+2\times n+2},$
$\hat{b} = circ\left(-\left(\frac{a_1b_1}{a_2} + b_2\right), b_1, b_2, \ldots, b_n\right), \hat{B} = circ(\hat{b}); AB = cI, c \neq 0.$
Then

$$\hat{A}\hat{B} = (2a_2\hat{b}_2 + a_1\hat{b}_1)I = (2a_2b_1 - \frac{a_1^2}{a_2}b_1 - a_1b_2)I$$

Proof. Let $A' = \frac{1}{a_2}A$ Then $A'B = \frac{c}{a_2}I$. By Theorem 1

$$\hat{A}'\hat{B} = (2\hat{b}_2 + \frac{a_1}{a_2}\hat{b}_1)I = (2b_1 - (\frac{a_1}{a_2})^2b_1 - \frac{a_1}{a_2}b_2)I$$

and therefore

$$\hat{A}\hat{B} = a_2\hat{A}'\hat{B} = (2a_2\hat{b}_2 + a_1\hat{b}_1)I = (2a_2b_1 - \frac{a_1^2}{a_2}b_1 - a_1b_2)I$$

□

Proposition 5. *If* $A = (a_1, a_2, 0, \ldots, 0, a_2)$ *and* $|\frac{a_1}{a_2}| > 2$, *then* A *admits an inverse.*

Proof: The result follows directly from the fact that A is a diagonally dominant, symmetric matrix.

4.2.3 Approximating the Inverse

Consider a second order recursive sequence of the form:

$$x_{n+1} = -(ax_n + x_{n-1})$$

where $a = \frac{a_1}{a_2}$. It is well known that the solution of this can be written as

$$x_n = c_1\alpha_1^n + c_2\alpha_2^n$$

where α_1 and α_2 are the roots of the characteristic polynomial

$$\alpha^2 + a\alpha + 1 = 0 \tag{4.1}$$

The constants c_1 and c_2 can be determined from the initial conditions, i.e., from x_0 and x_1.

Proposition 6. *With the above notations* $|\alpha_1| < 1$ *and* $|\alpha_2| > 1$ *if and only if* $|a| > 2$.

Proof. From Viète-formulas: $\alpha_1\alpha_2 = 1$ and $\alpha_1 + \alpha_2 = -a$. If $|\alpha_1| < 1$ and $|\alpha_2| > 1$, so $\alpha_1 \neq \alpha_2$, then

$$1 = \alpha_1\alpha_2 < \frac{\alpha_1 + \alpha_2}{2} = \frac{a}{2}$$

as α_1 and α_2 both have the same sign.

If $|a| > 2$, then $|\alpha_1 + \alpha_2| > 2$ and as they have the same sign and $\alpha_1\alpha_2 = 1$, we have $|\alpha_1| < 1$ and $|\alpha_2| > 1$. □

Theorem 4.2.2. *Let* $A_n = circ(a_1, a_2, 0, \ldots, 0, a_2) \in \mathcal{R}^{nxn}$. *If*

$$\left|\frac{a_1}{a_2}\right| > 2$$

and

$$A_n^{-1} = circ(y_{k+1}^n, y_k^n, \ldots, y_1^n, y_2^n, \ldots, y_k^n)$$

where $k = \left[\frac{n}{2}\right]$, *then*

$$\lim_{n\to\infty} y_l^n = \frac{\alpha_2^{-l}}{2a_2\alpha_2^{-1} + a_1}$$

where $|\alpha_2| > 1$ *and* α_2 *is the root of the characteristic polynomial 4.1.*

Remark: The condition $\left|\frac{a_1}{a_2}\right| > 2$ is the same as in Proposition 5.

Proof. Observe that according to Proposition 5, A admits an inverse. If x_k is the recursive sequence given by

$$x_{n+1} = -(\frac{a_1}{a_2}x_n + x_{n-1})$$

then, Corollary 1 yields

$$A_n^{-1} = \frac{1}{2a_2x_k + a_1x_{k+1}} circ(x_{k+1}, x_k, \ldots, x_1, x_1, x_2, \ldots, x_k)$$

if $n = 2k + 1$ and

$$A_n^{-1} = \frac{1}{2a_2x_k + a_1x_{k+1}} circ(x_{k+1}, x_k, \ldots, x_1, x_2, x_3, \ldots, x_k)$$

if $n = 2k$

Let us take a closer look at the matrix elements. They have the form

$$\frac{x_i}{2a_2x_k + a_1x_{k+1}}$$

For n odd this gives

$$\frac{c_1^o\alpha_1^i + c_2^o\alpha_2^i}{2a_2(c_1^o\alpha_1^k + c_2^o\alpha_2^k) + a_1(c_1^o\alpha_1^{k+1} + c_2^o\alpha_2^{k+1})}$$

and for n even we have

$$\frac{c_1^e\alpha_1^i + c_2^e\alpha_2^i}{2a_2(c_1^e\alpha_1^k + c_2^e\alpha_2^k) + a_1(c_1^e\alpha_1^{k+1} + c_1^e\alpha_2^{k+1})}$$

Fix $l = k + 1 - i$ and let $k \to \infty$. Observe that since $|\alpha_1| < 1$ thus $\alpha_1^k \to 0$, and $\alpha_1^i \to 0$. This yields the limits

$$y_l = \frac{\alpha_2^{-l}}{2a_2\alpha_2^{-1} + a_1}$$

It is easy to see that the convergence is exponential, proving the theorem. □

The above theorem provides us with a tool to compute the inverses in an efficient way, where y_l is replaced by zero for small values. The recursive-sequence in the support point transfomation case is

$$x_n = -4x_{n-1} - x_{n-2}$$

yielding

$$y_l = \frac{(-2-\sqrt{3})^{-l}}{\frac{1}{3}(-2-\sqrt{3})^{-1} + \frac{2}{3}}$$

According to the above formula we approximate A_n^{-1} with a circulant matrix keeping just those elements for which: $|\frac{(-2-\sqrt{3})^{-l}}{\frac{1}{3}(-2-\sqrt{3})^{-1}+\frac{2}{3}}| < \epsilon$.

It is easy to check that the 6th element of the sequence is already closer to zero than 10^{-4}. This means that instead of the exact sequence, it is enough to work with 11 elements: $\overline{A_n^{-1}} = circ(y_0, y_1, \ldots, y_5, 0, \ldots, 0, y_5, y_4, \ldots, y_1)$. When the spline is defined by less than 11 support points the exact inverse should be calculated.

4.2.4 Open Splines

In the previous section we have shown a computationally efficient method for calculating control points from support points for closed cubic spline-functions. We do these for open spline-functions. This we do using the following trick: instead of support points $q_1, q_2, \ldots, q_n$ treat the support point sequence $q_0, q_1, \ldots, q_n, q_{n-1}, q_{n-2}, \ldots, q_1$, and fit a closed spline-function on it.

Proposition 7. *The spline function given by the trick above is symmetric in the sense that*

$$s(n-x) = s(n+x), x \leq n$$

Proof. We prove first that the control points are "symmetric", i.e:

$$p_{n-i} = p_{n+i}, \quad 0 < i < n$$

First note that the new support point sequence has an even number of members. As we saw previously the matrix that transforms support points to control points has the form:

$$A_n^{-1} = circ(y_{k+1}, y_k, \ldots, y_1, y_2, y_3, \ldots, y_k) = circ(y^T)$$

Let $q = (q_0, q_1, \ldots, q_k, q_{k-1}, \ldots, q_1)$ and let

$$R = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \ldots & 0 & 0 \end{pmatrix}$$

Observe that $R = R^T$ and $Rq = q$.

As $A_n^{-1}q = p$, $p_{n-i} = (S_r^{n-i}y)^T q$ and $p_{n+i} = (S_r^{n+i}y)^T q$. Using the special form of y it can be seen by inspection that

$$RS_r^{n-i}b = S_r^{n+i}b$$

With this, the proof follows directly:

$$p_{n+i} = (S_r^{n+i}y)^T q = (RS_r^{n-i}y)^T q = (S_r^{n-i}y)^T Rq = (S_r^{n-i}y)^T q = p_{n-i}$$

Now the result follows from

$$s(x) = \sum_{i=0}^{n} p_i \varphi_{i-2}(x) + \sum_{i=1}^{n-1} p_{n-i}\varphi_{n+i-2}(x) = \sum_{i=0}^{n} p_i \varphi_{i-2}(x) + \sum_{i=1}^{n-1} p_i \varphi_{2n-i-2}(x) =$$

$$= \sum_{i=0}^{n} p_i \varphi_{-2}(x-i) + \sum_{i=1}^{n-1} p_i \varphi_{-2}(x-2n+i)$$

and thus

$$s(n-x) = \sum_{i=0}^{n} p_i \varphi_{-2}(n-x-i) + \sum_{i=1}^{n-1} p_i \varphi_{-2}(n-x-2n+i)$$

and

$$s(n+x) = \sum_{i=0}^{n} p_i \varphi_{-2}(n+x-i) + \sum_{i=1}^{n-1} p_i \varphi_{-2}(n+x-2n+i)$$

Exploiting that φ_{-2} is even and its support is $[0,4]$, the prrof is ready. □

4.3 Object tracking using B-spline contour models

In this section we propose an implementation of LS-N-IPS for object tracking using B-spline contour models.

In order to implement LS-N-IPS one needs to define three objects: the dynamics used in the prediction step, the local search operator and the observation density. In our case the state space will consist of the pose of the object to be tracked along with the previous pose (the dynamics is a second order AR process). The pose defines a contour mapped onto the camera plane (i.e., onto the image).

The idea is to use this B-spline based contour model as the basis of the observation calculations: the better the contour fits the image, the more likely the corresponding pose is. B-splines, however will be represented here by their support points (i.e. by points along the curve) as opposed to the usual represenation of B-splines via their control points.

This is needed since the local search is implemented on the image by finding an allowable spline contour in the vicinity of the original one that fits the image the best in that given small neighbourhood. This search is implemented by finding the most likely locations of edges along the normals of the original curve *at the support points*. By using support points instead of control points we spare some matrix vector products that would involve matrices whose size scales with the complexity of the contour. On the other hand, the usage of support points lefts the complexity of the algorithm intact otherwise.

In the following subsections we provide the details of the algorithm.

4.3.1 Spline contours, configuration space

Let us consider the spline curve $s : [0, L] \to \mathcal{R}^2$ defined by its support points $q^x = (q_1^x, \dots q_n^x)^T, q^y = (q_1^y, \dots q_n^y)^T$, i.e: $s(t) = ((A^{-1}q^x)^T\varphi(t), (A^{-1}q^y)^T\varphi(t))$, where $\varphi(t) =$

$(\varphi_1(t), \varphi_2(t), \ldots, \varphi_n(t))^T$ are the usual B-spline basis functions (cf. [1]) and $s(i) = (q_i^x, q_i^y)^T$. A is the linear transformation mapping control points to support points.

Let $q_0 = ((q_0^x)^T, (q_0^y)^T)$ be a vector of support points defining the contour s_0. If G is a group of similarity transformations of the 2D plane ($\mathcal{R}^2$) then one can find a matrix $W = W_{G,q_0}$ such that $T \in G$ iff for some $z \in \mathcal{R}^k$, the support vector $q = Wz + q_0$ yields the spline curve $T\, s_0$ (we use the convention that $z = 0$ corresponds to $T = \mathrm{Id}$). $\mathcal{R}^k$ then corresponds to the set of allowed configurations of the object.

Note 3. *If G is the group of Euclidean similarities of the plane then*

$$W = \begin{pmatrix} \mathbf{1} & \mathbf{0} & q_0^x & -q_0^y \\ \mathbf{0} & \mathbf{1} & q_0^y & q_0^x \end{pmatrix}, \tag{4.2}$$

where $\mathbf{0} = (0, 0, \ldots, 0)^T$, $\mathbf{1} = (1, 1, \ldots, 1)^T$.

4.3.2 Implementation of the Local Search

Assume that a contour (s) corresponding to some pose (z) is given. Blake and Isard define the likelihood of the contour given the image (the observation) as the product of the individual "likelihoods" of edges being located at some measurement points along the spline curve [7].

Motivated by this definition, our local search algorithm searches for maximal edge 'likelihood' values along the normals in the vicinity of the measurement points, the neighborhood itself defined by the search length, $l > 0$. The measurement points are chosen to be the support points $(q_1^x, q_1^y)^T, \ldots, (q_n^x, q_n^y)^T$.

Assume that the search yields the points $(\hat{q}_1^x, \hat{q}_1^y)^T, \ldots (\hat{q}_n^x, \hat{q}_n^y)^T$ and let $\hat{q} = (\hat{q}_1^x, \ldots \hat{q}_n^x, \hat{q}_1^y, \ldots \hat{q}_n^y)^T$ (for an example see Figure 4.2). Let $\hat{s}$ be the spline curve corresponding to $\hat{q}$.

The next step is to find a configuration whose corresponding spline curve matches $\hat{s}$ the best: $\hat{z} = \mathrm{argmin}_{z \in \mathcal{R}^k} \|s_z - \hat{s}\|_2^2$. Here s_z denotes the spline curve corresponding to the configuration z, i.e., the spline curve corresponding to the support vector $q_z = Wz + q_0$.

It is well known that if s is the spline contour corresponding to q then $||s||_2^2$ can be expressed as a function of q alone. In particular,

$$||s||_2^2 = \frac{1}{L} \int_0^L s^2(t)dt = q^T \begin{pmatrix} \mathcal{B} & 0 \\ 0 & \mathcal{B} \end{pmatrix} q = q^T \mathcal{U} q,$$

where

$$\mathcal{B} = A^{-T} \left(\frac{1}{L} \int_0^L \varphi(u)\varphi(u)^T du \right) A^{-1}.$$

Therefore, if we let $\|q\|_S^2 = q^T U q$ be the weighted ℓ^2 norm then $\hat{z} = \mathrm{argmin}_{z \in \mathcal{R}^k} \|Wz + q_0 - \hat{q}\|_S^2$. Now, by standard LMS calculations $\hat{z} = W^+(\hat{q} - q_0)$, where $W^+ = (W^T \mathcal{U} W)^{-1} W^T \mathcal{U}$. The corresponding projected support vector shall be denoted by $q^\perp = W\hat{z} + q_0$.

It remains to specify the likelihood of the projected particle $x^\perp$ (the previous configuration is not used in the observation likelihood calculations as still images reflect only

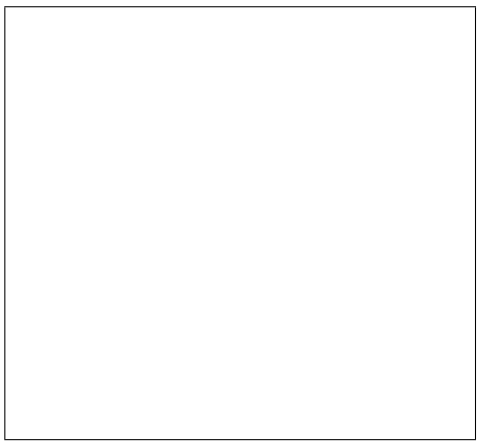

Figure 4.1: *Observation on the curve normals(black), and maximum value points(grey) are calculated.*

momentarily information). The ideal likelihood would be the product of the likelihoods of the edges now measured at the points defined by the new support vector $q^\perp$. However, in favour of speed we would like to reuse the maximal likelihoods found during the search process. Therefore we start with the product of these and compensate for the errors introduced by this approximation.

Firstly, note that it is reasonable to devalve this value by a value proportional to the distortion caused by the projection, i.e. by $||q^\perp - \hat{q}||_S$: the larger the distortion is the less the likelihood of the given configuration should be, since a large distortion means that the spline curve defined by $\hat{q}$ was far from the template.

However, $||q^\perp - \hat{q}||_2$ must be normed by the "scale" of the corresponding configurations: remember that $||q^\perp - \hat{q}||_2$ equals to the $\mathcal{L}^2$ distance between the spline curves corresponding to $q^\perp$ and $\hat{q}$, i.e., the area in between these curves. Now if $q^\perp$ and $\hat{q}$ correspond to "big" contours then $||q^\perp - \hat{q}||_2$ will be "big" itself. Therefore $||q^\perp - \hat{q}||_2$ must be scaled by the size of the represented object. In a somewhat ad hoc way, we have chosen to norm this by d^2, where d is the scale of the object being in the configuration $x^\perp$. The search length was also scaled by d, so that the search length will correspond to a fixed search length of the physical world. In summary, the algorithm works as follows:

Evaluation Algorithm with Local Search. (Inputs: image (I), configuration (x))

1. Calculate the scaling factor d given x. (In case of Eucleidan similarities $d = \sqrt{(1+x_3)^2 + x_4^2}$.)
2. Calculate the search length $l = dl_0$, where l_0 is a user defined parameter.
3. For all normal of lines of the contour at (q_i^x, q_i^y), where $q = Wx + q_0$ $(i = 1, 2, \ldots, n)$:
   - For every point on the normal and within the search length l calculate the likelihood of an edge being at that point.
   - Select the maximum value along the normal.

4. Compute v, the product of the obtained maximum values, and let the locations of the maximums define $\hat{q}$.

5. Calculate the best LMS configuration

$$x^{\perp} = W^{+}(\hat{q} - q_0)$$

and the projected support vector $q^{\perp} = Wx^{\perp} + q_0$.

6. Return

$$c = v\frac{d^2}{||q^{\perp} - \hat{q}||_2}$$

Figure 4.2 shows a contour corresponding to a single particle before (white) and after the local search (grey). It should be clear from the figure that the local search adjusted contour has a much higher likelihood of being the correct contour of the hand on the given image.

Note 4. *The search length at different support points can be different depending on the local characteristic of the template curve at each point. Actually the whole measurement process can be specialized at each normal lines, and this is how multiply color object can be tracked.*

Figure 4.2 shows a contour corresponding to a single particle before (white) and after the local search. It should be clear from the figure that local search adjusted the contour to a much higher likelihood given the state of the hand.

Note 5. *The calculation of W^{+} is computationally efficient since multiplication with $\mathcal{U}^{*}$ (being circulant matrix) needs $O(n)$ as it was argued before.*

4.4 Experiments

The proposed algorithm was tried in a number of visual object tracking problems. In the scenarios shown here the hand-tracking was attempted. Edge likelihoods along the normals were computed as the product of the edge strength along the normal and the color match "inside"the object. For color matching a Gaussian density is used that is trained on the first frame. In the present implementation users have to draw the contour of the object to be tracked on the first frame by determining the support points of the curve. For G, the full Euclidean similarity group of the plane was chosen with W given by Equation 4.2, and thus the scale of an object with configuration $z = (z_1, z_2, z_3, z_4)^T$ is given by $d = \sqrt{(1 + z_3)^2 + z_4^2}$.

The state is defined by the pose and the previous pose (i.e., $\mathcal{X} = \mathcal{R}^8$). The pose space is defined as the parameterization of the transformation group defined by the subgroups induced by x, y-translations, rotations, and scaling, i.e., the pose space and the configuration space are non-linearly related. A second order dynamics was fitted for each dimension of the pose space, independently of each other. The dynamics of scale and rotation were adjusted by hand due to insufficient training data.

Figure 4.2: *Comparing the contour predicted and the contour adjusted in the local search procedure. The white contour is the predicted contour, black + signs are at the maximum values around the contour. The grey contour is the one given by the local search.*

Processing step	**Time** [s]	**Percentage**
Image preparation	0.012	28.3 %
Likelihood calculation and local search	0.0248	58.5 %
Prediction and update	0.055	13.2 %
Total	**0.0424**	**100 %**

4.5 Results

The proposed algorithm was tested in a number of tracking tasks. In general, the algorithm performed very well under a wide range of conditions even with very small particle sizes, such as $N = 50$. Thanks to relying mostly on contour (i.e. shape) information, the algorithm could work under a wide range of illumination conditions, even when the lighting was weak.

Objects were tracked reliably and precisely (see Figures 4.4 and 4.5). In case of sudden movements the tracker occassionally lost the object but it could quickly recover in all of the cases. This can be explained by the relatively high variance of the noise of the dynamics and the high specificity of the observation likelihood: If the object is lost then all particles become roughly equally weighted and the system starts to perform a random diffusion, exploring the image. If some particle becomes in the vicinity of the true object position the local search refines the particle position quickly, the weight of the corresponding particle becomes large and the particle is reproduced with high probability in many copies in the resampling step. This causes the tracker to recover. If the shape information is not sufficiently specific then the tracker may lock on spurious "objects" having a contour similar to that of the object to be tracked. However, this is a common property of contour based tracking algorithms and is thus not special to LS-N-IPS. Note that no special mechanism was needed for the reinitialization of the tracker and the tracker was capable of finding the object to be tracked even when the configuration of the initial particle set was randomized.

Figure 4.5 and 4.4 shows every 15th frame of a typical tracking session. This image sequence was recorded at 30 frames/second. The number of particles was chosen to be 100, $l_0 = 10$. The image resolution was 240×180. Processing times projected for one frame are shown in the table below, measured on a 1 GHz Pentium 4 computer.

Note that optimization of the algorithm is still possible. In the image preparation step the color likelihoods are computed for each pixel (this step could be eliminated and the color likelihoods be computed only when and where they are needed). The prediction and update steps require negligable time. Most of the time is spent on calculating the observation likelihoods and the local search procedure. We have also made measurements with the local search switched off (and using the unmodified observation likelihoods). This corresponds to running N-IPS (or CONDENSATION). We found that running the algorithm in this way, $N = 200$ particles requires roughly the same amount of processing time than that of needed by LS-N-IPS with $N = 100$. In Figure 4.3 the tracking results of N-IPS can be seen with $N = 200$ particles. It should be clear after comparison with Figure 4.4 that LS-N-IPS performs significantly better under the "equivalent running time" condition. N-IPS

Figure 4.3: *Tracking an object with the N-IPS algorithm. The number of particles was 200. Images are shown for every* 15*th frame.*

was pretty much useless for tracking more complex shapes, such as hands, unless N was raised above 2000 which would allow a tracking speed of less than 3 frames/second, to be compared with the approximate tracking speed of 23.6 frames/second obtained with the proposed algorithm.

4.6 Related Work

Sequential importance sampling with resampling [10, 15, 5], or SIR (also known as iCONDENSATION [8] in the image processing community) is designed to overcome problems related to peaky *posteriors*. The design parameter of this algorithm is a so-called proposal distribution whose purpose is to concentrate particles to the highly probable parts of the state space.

Unfortunatelly, good proposal distributions are not easy to design. The ideal proposal

Figure 4.4: *Tracking an object with the LS-N-IPS algorithm. The number of particles was 100. Images are shown for every* $15th$ *frame.*

Figure 4.5: *Tracking hand in clutter with LS-N-IPS sample size* 100. *Black contour shows particles with high observation coefficients, white contour show the predicted position.*

distribution depends both on the dynamics and the most recent observation. However, a proposal should also be fast to sample from. Blake and Isard suggest to use a density fitted to the output of a color detector as the proposal distribution. The for of the density in their application is a mixture of Gaussians, so it is cheap to sample from it. Unfortunately however, since the importance function is not defined on the speed of the object, motion coherence information is used by this algorithm in a limited way. Despite this, they were able to obtain excellent results with particle sizes in the range of $150-400$. Note that their algorithm uses an unusual $\mathrm{O}(N^2)$ step[1] (for computing the importance weights). We have run some experiments and observed that the computation time of this $O(N^2)$ step starts to dominate the rest of the computations already for medium values of N.

The Auxiliary Variable Method (AVM) by Pitt and Shephard resembles LS-N-IPS more closely. In this method the proposal is a mixture of the prediction densities with the mixture coefficients defined as some observation likelihoods computed at certain points. These "anchor" points conceptually correspond to the predicted next states of the particles but can be chosen by means of some deterministic computation, e.g., by chosing the most likely next states (based on the prediction density) or the expected next states.

This method takes into account both the observation and the prediction densities. However, even this method will be inefficient when the observation density is highly peaky. In some sense, this method can be viewed as the dual of our method: it uses the prediction density to search for likely particles, whilst our method uses the observation density to guide the search. There seems to be a natural way to incorporate local search into the AVM algorithm by using a local search on the observation density to locate the positions of the anchor points. This would yield an unbiased particle filter. Unfortunately, the additional sampling steps that sample from the mixture prediction density would make this algorithm inefficient for peaky observations and high variance prediction densities.

[1] In fact, it seems to the authors of this article that the usual $\mathrm{O}(N)$ reweighting step would be sufficient (and correct) for their algorithm, as well.

Chapter 5

Conclusions and Future Work

We have introduced the LS-N-IPS algorithm, a modification of the basic N-IPS algorithm. The algorithm was argued to perform significantly better than the baseline N-IPS algorithm in the small particle size limit.

A theoreticaly analysis of the stability and robustness of the arising filters was studied using powerful techniques from the theory of Markov chains. A new result concerning robust approximate tracking was derived. The main assumptions used in these results is the (one-step) mixing property of the Markov models and the uniform positivity of the observation likelihood functions. In these results the role played by the observations is less characterisics as one would desire. A number of examples were given highlighting the difficulties when trying to weaken these conditions. In the future we plan to actually prove some positive results with weakened ergodicity assumptions.

The proposed model was applied to the problem of object tracking. Spline-countour and color matching based observation models were developed and the models were tested. In accordance with the expectations, the LS-N-IPS algorithm did perform significantly better than the baseline N-IPS algorithm.

Future work in the image processing area will include speeding up the algorithm by clever organization of the computing steps. Another important line of research is to consider alternative, more complex observation models since when used with simple, not sufficiently discriminatory contours contour based observations become sensitive to clutter. Possibilities include changing the search length locally, using local color models at the different measurement points to track multi-color objects, or using texture information. Yet another interesting alternative is to employ a local search operator in the style of SSD [3] that would use templates instead of contours. Further, it is very challenging to attempt to track more complex, multi-part objects or multiple objects and incorporate interaction between the objects (occlusion) into the model. Fortunately, the proposed algorithm kept much of the similarity of the original N-IPS algorithm and thus combining it with other advanced algorithms, such as partitioned sampling [11] should be a routine work. This way, we hope to be able to build systems that not only process the images but understand them, as well, at least to some tent.

Appendix A

The N-IPS Algorithm

In this chapter we provide the pseudocode of the N-IPS (N Interacting Particle System) algorithm, also known as CONDENSATION [7] or the resampling method. The input of this method are π_0, f, g, $Y_1, Y_2, \ldots$ and it is assumed that can sample from the state noise process W_t.

1. Initialization:
   - Let $X_0^{(i)} \sim \pi_0$, $i = 1, 2, \ldots, N$ and set $t = 0$.
2. Repeat forever:
   - Compute the proposed next states by $Z_{t+1}^{(i)} = f(X_t^{(i)}) + W_t^{(i)}$, $i = 1, 2, \ldots, N$.
   - Compute $w_{t+1}^{(i)} \propto g(Y_{t+1}|Z_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$.
   - Sample $k_{t+1}^{(i)} \propto (w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N)})$, $i = 1, 2, \ldots, N$.
   - Let $X_{t+1}^{(i)} = Z_{t+1}^{(k_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

Appendix B

The LS-N-IPS Algorithm

Here we provide the pseudocode of the LS-N-IPS algorithm.

1. Initialization:
   - Let $X_0^{(i)} \sim \pi_0$, $i = 1, 2, \ldots, N$ and set $t = 0$.
2. Repeat forever:
   - Compute the proposed next states by $Z_{t+1}^{(i)} = L_\lambda(f(X_t^{(i)}) + W_t^{(i)}; Y_{t+1})$, $i = 1, 2, \ldots, N$.
   - Compute $w_{t+1}^{(i)} \propto g(Y_{t+1}|Z_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$.
   - Sample $k_{t+1}^{(i)} \propto (w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N)})$, $i = 1, 2, \ldots, N$.
   - Let $X_{t+1}^{(i)} = Z_{t+1}^{(k_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

Appendix C

The Auxiliary Variable Method

Here we provide the pseudocode of the AVM by Pitt and Shepard.

1. Initialization:
   - Let $X_0^{(i)} \sim \pi_0$, $i = 1, 2, \ldots, N$ and set $t = 0$.
2. Repeat forever:
   - Compute the basis points $\mu_{t+1}^{(i)}$ e.g. by $\mu_{t+1}^{(i)} = f(X_t^{(i)})$.
   - Generate $Z_{t+1}^{(k)})$, $k = 1, 2, \ldots,$, with $R \geq N$ from

$$\sum_{i=1}^{N} \frac{g(Y_{t+1}|\mu_{t+1}^{(i)})}{\sum_{j=1}^{N} g(Y_{t+1}|\mu_{t+1}^{(i)})} f(\cdot|X_t^{(i)}).$$

     e.g. by first generating a random variable $L_{t+1}^{(k)}$ taking values in $\{1, 2, \ldots, N\}$ with distribution $P(L_{t+1}^{(k)} = i) = \frac{g(Y_{t+1}|\mu_{t+1}^{(i)})}{\sum_{j=1}^{N} g(Y_{t+1}|\mu_{t+1}^{(i)})}$ and then let $Z_{t+1}^{(k)} = f(X_t^{(L_{t+1}^{(k)})}) + W_t^{(k)}$.
   - Calculate the importance weights

$$w_{t+1}^{(k)} = \frac{g(Y_{t+1}|Z_{t+1}^{(k)})}{g(Y_{t+1}|X_t^{(L_{t+1}^{(k)})})}, \quad k = 1, 2, \ldots, R$$

   - Sample $k_{t+1}^{(i)} \propto (w_{t+1}^{(1)}, \ldots, w_{t+1}^{(R)})$, $i = 1, 2, \ldots, N$.
   - Let $X_{t+1}^{(i)} = Z_{t+1}^{(k_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

Bibliography

[1] Andrew Blake and Michael Isard. *Active Contours*. Springer-Verlag, 1998.

[2] Amarij Budhiraja and Harold J. Kushner. Robustness of nonlinear filters over the infinite time interval. *SIAM J. Control and Opt.*, 36:1618–1637, 1998.

[3] Gregory D.Hager and Peter N.Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. on PAMI*, to appear.

[4] R.L. Dobrushin. Central limit theorem for nonstationary Markov chains, I, II. *Theory of Probability and Applications*, 1(1 and 4):66–80 and 330–385, 1956.

[5] Arnaud Doucet. On sequential simulation based methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 1998.

[6] E.Seneta. *Non-Negative Matrices and Markov Chains*. Springer-Verlag, 1981.

[7] Michael Isard and Andrew Blake. CONDENSATION – conditional density propagation for visual tracking. *International Journal Computer Vision*, 29:5–28, 1998.

[8] Michael Isard and Andrew Blake. ICondensation: Unifying low-level and high-level tracking in a stochastic framework. *Proc 5th European Conf. Computer Vision*, 1998.

[9] Mishael Isard. *Visual Motion Analysis by Probabilistic Propagation of Conditional Density*. PhD thesis, University of Oxford, 1998.

[10] Tanner M.A. and Wong W.H. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 1987.

[11] John MacCormick. *Probabilistic Modelling and Stochastic Algorithms for Visual Localisation and Tracking*. PhD thesis, University of Oxford, 2000.

[12] Pierre Del Moral. A uniform convergence theorem for the numerical solving of the nonlinear filtering problem. *Journal of Applied Probability*, 35:873–884, 1998.

[13] Pierre Del Moral. On the stability of interacting processes with applications to filtering and genetic algorithms. *Annales de l'Institut Henri Poincar*, 2001. (to appear).

[14] Michael K. Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filter. *Journal of the American Statistical Association*, 94:590–9, 1999.

[15] D.B Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The sir algorithm, comment on tanner and wong(1987). *Journal of the American Statistical Association*, 1987.

[16] Georege S.Fishman. *Monte Carlo Concepts, Algorithms, and Applications*. Springer-Verlag, 1999.

[17] Anderson B. D. O. Shue, L. and Dey S. Exponential stability of filters and smoothers for Hidden Markov Models. *IEEE Transactions On Signal Processing*, to appear.

[18] Péter Torma and Csaba Szepesvári. LS-N-IPS: an improvement of particle filters by means of local search. (submitted), 2000.

[19] Péter Torma and Csaba Szepesvári. Efficient object tracking in video sequences by means of LS-N-IPS. (submitted), 2001.