Combining Local Search, Neural Networks and Particle Filters to Achieve Fast and Reliable Contour Tracking

Péter Torma
Mindmaker, Ltd.
Budapest 1121 HU
Konkoly Th. M. u. 29-33
tyus@mindmaker.hu

Csaba Szepesvári
Mindmaker, Ltd.
Budapest 1121 HU
Konkoly Th. M. u. 29-33
szepes@mindmaker.hu

Abstract

LS-N-IPS (*L*ocal-*S*earch-*N*-*I*nteracting-*P*article-*S*ystem) is an extension of the standard N-IPS particle filter (also known as CONDENSATION in the image processing literature). The modified algorithm adds local search to the baseline algorithm: in each time step the predictions are refined in a local search procedure that utilizes the most recent observation. A critical choice in the design of LS-N-IPS is the way the local search is implemented. Here, we introduce a method based on training artificial neural networks for implementing the local search. In experiments with real-life data (visual tracking) the method is shown to improve robustness and performance significantly, surpassing the performance of previous state-of-the-art algorithms.

1 Introduction

In this paper we study methods for improving the speed and robustness of contour tracking algorithms that employ particle filters to achieve real-time performance on commercial computers on the problem of tracking complex contours moving fast in highly cluttered environments.

Contour tracking with particle filters has been studied in the image processing literature e.g. in [2], where the CONDENSATION algorithm, also known as the N-IPS algorithm [4][1] has been introduced.

As it is well known, N-IPS suffers from inefficiency problems when the observation likelihood function is highly spatially concentrated (see the left hand side of Figure 1). In [6] a solution was proposed to overcome this problem. The resulting algorithm, called LS-N-IPS (Local Search based N-IPS), uses a local search procedure to relocate particles after the prediction step to regions where the observation likelihood function admits large values. The algorithm was shown to decrease the variance of tracking error at the price of introducing a small bias and increasing the computational burden of the algorithm slightly, making it possible to increase the overall efficiency and precision of tracking whilst keeping

[1] The name N-IPS comes from "N-Interacting Particle System".

the computational cost fixed. In [5] the same algorithm was shown to boost performance significantly in a contour tracking task.

In this paper we show that using machine learning one can further improve the performance by optimizing the way particles are moved in the local search procedure. In the new algorithm a mapping minimizing the expected configuration error given the initial configuration and the adjusted observed contour points is used, whilst in the algorithm studied in [5] a contour was fitted to the adjusted contour points first and then this contour was used to derive the adjusted configuration. The new algorithm is shown to be more robust and to be twice as fast as a previous state-of-the-art algorithm, achieving a frame rate 60 frames/second on an average commercial computer, without ever sacrifying performance.

The paper is organized as follows: In Section 2 the model and the basic LS-N-IPS algorithm are given. In Section 2.1 the learning approach is proposed. Experiments on a real-world object tracking problem comparing the old and new methods are presented in Section 3. Finally, conclusions are drawn in Section 4.

2 Model and Algorithm

We consider the problem of filtering the system

$$X_{t+1} = f(X_t) + W_t, \quad (1)$$
$$Y_t = g(X_t) + V_t, \quad (2)$$

where $t = 0, 1, 2, \ldots$ is the time, $X_t, W_t \in X$, X is the state space of the system, $Y_t, V_t \in Y$, Y is the observation space of the system, and W_t, V_t are zero mean i.i.d. random variables. Let the posterior given the observations $Y_{0:t} = (Y_0, \ldots, Y_t)$ be π_t: $\pi_t(A) = P(X_t \in A|Y_{0:t})$,. Here $A \subset X$ is any measurable set.

The LS-N-IPS algorithm proposed in [6] is shown in Table 1. LS-N-IPS and N-IPS differ

1.	Initialization:
	Let $X_0^{(i)} \sim \pi_0$, $i = 1, 2, \ldots, N$ and set $t = 0$.
2.	Repeat forever:
2.1.	Let $Z_{t+1}^{(i)} = S_\lambda(f(X_t^{(i)}) + W_t^{(i)}, Y_{t+1})$, $i = 1, 2, \ldots, N$.
2.2.	Compute $w_{t+1}^{(i)} \propto g(Y_{t+1}|Z_{t+1}^{(i)})$, $i = 1, 2, \ldots, N$.
2.3.	Sample $k_{t+1}^{(i)} \propto (w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N)})$, $i = 1, 2, \ldots, N$.
2.4.	Let $X_{t+1}^{(i)} = Z_{t+1}^{(k_{t+1}^{(i)})}$, $i = 1, 2, \ldots, N$.

Table 1: The LS-N-IPS Algorithm

only in the way the proposed states ($Z_{t+1}^{(i)}$) are updated: LS-N-IPS uses a non-trivial local search operator, S_λ, to "refine" the predictions: S_λ is typically chosen to locally maximize the observation likelihood function $g(\cdot|x)$[2], whilst in N-IPS no such step is implemented (here S_λ equals to the identity mapping). Figure 1 illustrates the advantage of LS-N-IPS over N-IPS: LS-N-IPS makes better use of the available particles because in the local search step particles are moved towards regions having higher observation likelihood values. At the same time, the computational cost of LS-N-IPS is only slightly larger than that of N-IPS, in particular the number of likelihood calculations is kept fixed.[3] This makes LS-N-

[2]Function $g(y|x)$ denotes the observation density function of the system to be filtered (cf. Equation 2).

[3]This would not be the case for importance sampling based methods. The discussion of this topic, however, is out of the scope of the present article.

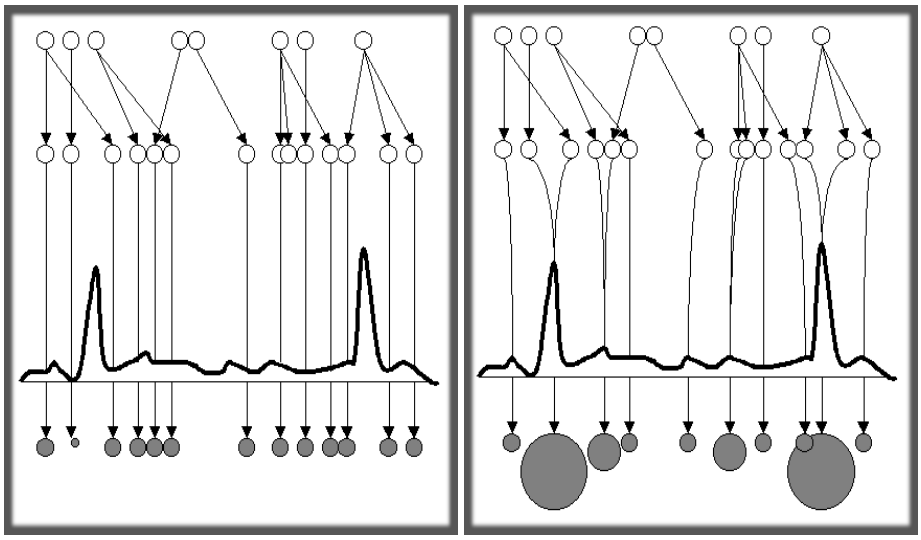

Figure 1: Illustration of the working mechanisms of N-IPS and LS-N-IPS. The figures, from top to bottom show the position of particles before the prediction step (top row), after the prediction step (second row), an example observation likelihood function, whilst the last row shows the position of the particles after the local search. In this row the sizes of the circles are proportional to the weights associated with the respective particles.

IPS an attractive alternative of the many other known modifications of N-IPS for tracking problems where likelihood calculations are expensive, such as e.g. for visual tracking where evaluating the observation likelihoods involves the image processing steps.

Clearly, the efficiency of LS-N-IPS depends heavily on the effectiveness of the local search procedure.

2.1 Training Local Search Operators

Here we propose to use machine learning (in this paper we shall use artificial neural networks (ANNs) trained by a variant of backpropagation) to learn the mapping that maps observations, and the initial configuration to a "corriged" configuration. The suggested criterion that forms the basis of training is to minimize the cost $\mathrm{E}[\|S_\lambda(X', Y) - X\|^2 \mid \|X - X'\| \leq \lambda]$, where λ is a parameter governing the "range" of local search and X, X' are random configurations corresponding to the true configuration, and the initial (predicted) configuration, respectively, and $Y \sim g(\cdot|X)$. In practice, S_λ may take the form of e.g. a neural network and the empirical error is minimized with the training data being generated by random sampling. In the next section we will see how this is done for contour tracking.

3 Experiments

3.1 The Contour Tracking Algorithm

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 (configuration) of the object to be tracked along with the previous pose. The dynamics is modelled as a second order AR process, whose parameters

are defined experimentally.[4] A pose defines a *contour* mapped onto the camera plane (i.e., onto the image) and observations will be defined in terms of contours and the observed image. Due to the lack of space only the outline of the contour tracking algorithm is given here. The reader may find more detailed information about this algorithm in the paper [5].[5]

Local search. Contours are given by a B-spline representation and B-splines themselves are represented by their support points (i.e. points *on* the contour). The advantage of this representation is that the local search method can be implemented readily using support points as follows (the same would not hold true for control points based contours): After the state of a particle has been predicted, the support points corresponding to the predicted pose can be computed by means of an affine mapping. Then, the image is searched for likely contour points (points having large likelihood of being a point of an edge) along the normals of the curve at the support points in a small neighborhood of those support points.[6] These adjusted support points are then used together with the computed edge-likelihood values in the subsequent calculations.

LMS-based Local Search. In the algorithm of [5] the adjusted pose is computed by fitting a contour from the family of admissible contours to the adjusted support points by minimizing the L^2 distance of the B-spline curve defined by the adjusted support points. It turns out, that these computations amount to solving a standard weighted finite dimensional least-mean square (LMS) problem, hence we call this method "LMS-based".

ANN-based Local Search. In the data-oriented approach a neural network is used to predict the adjusted pose given the adjusted support points and the initial pose. In order to reduce the complexity of the learning problem we have further simplified the task by providing the network with normalized training data, whereas the training data is generated such that the initial pose is always the same. Therefore no inputs were needed for the initial pose and the network was trained to learn the best "correction" factor for a single default pose. Both for training and when the network is used, the adjusted support points were normalized accordingly (by means of a simple affine transformation) and the output of the network was interpreted relative to the initial pose, too.

Computing the Observation Likelihoods. In order to make results comparable, likelihood calculations were the same irrespective of how the local search was carried out. The likelihood calculation that was shown in [5] to give the best results in terms of both tracking accuracy and efficiency was adopted. This algorithm works as follows: First the product of the individual edge likelihoods is calculated as in [2] at the adjusted support points. Then, this number is compensated for the errors that can be attributed to the adjusted support points being different from the support points corresponding to the spline of the adjusted pose. For this the distances of these respective supports points were calculated and summed up. This number was then normalized with the scale of the predicted contour (because larger contours give rise to larger absolute distances) and the resulting number was used as a rough estimate of the reliability of the initial product as an estimate of the likelihood of the final predicted contour. The final likelihood value is obtained by dividing the initial product by its reliability estimate.

3.2 Results

In the experiments translation, rotation and scaling of contours was modelled.

---

[4] The same model was used throughout all the experiments independently of how the local search was implemented.

[5] Data and experimental results are given at www.mindmaker.hu/~szepes/lsnips

[6] The size of the neighbourhood was scaled with the scale of the predicted contour.

Experiment	LMS	ANN0	ANN1	ANN2
Prediction error on artificial data	0.2268	0.1437	0.1414	0.1387
Prediction error on real images	0.2854	0.1818	0.1835	0.1743
Prediction time (ms)	0.31	0.31	0.788	1.062

Table 2: Results on artificial and real-data.

Training. Several networks were tried with various contours. In all the experiments the net outputs were scaled so that to ensure that they are on the same scale. All networks were trained with $100,000$ randomly generated data. Training data was generated by drawing a random perturbation of the default pose and then searching for the intersection of the contour defined by the default pose and along the normals of the perturbed curve at the support points of it. These intersection points were then perturbed by Gaussian noise, modelling observation noise and then inputted to the network. The desired output was set to the perturbed pose. The networks were trained by RProp [3].

Network configurations. ANN0 is a degenerate neural network, with no hidden layer and no non-linearity, i.e. a linear model. ANN1 is a neural network with one hidden layer consisting of 20 neurons, whilst ANN2 is a neural network with one hidden layer of 40 neurons. Other network configurations were tried with results consistent with those given below.

3.2.1 Accuracy of the learnt local search operator

We tested the local search algorithms on both artificial and real data. For the purpose of tests on synthetic data another set of samples was generated randomly, the cardinality of which was $10,000$. No noise was applied on the adjusted support points in this case. Uniformly distributed random poses were drawn in a symmetric rectangular neighborhood of zero having sizes $(14, 14, 0.3, 0.2)$, where the first two coordinates correspond to translation parameters (the contour to be tracked was about of size 60×60 on the same scale), the third coordinate is rotation in radians, whilst the last one is the scale parameter.

In the real life experiment $5,000$ random test poses were generated on a selected frame of the sequence of Figure 4, in the vicinity of the "true" pose.

Table 2 shows the measured error on both the simulated and real-world data. Prediction times (measured on an Intel Pentium 4, 1.4 GHz processor) are also given. Paired t-tests have shown that the measured performance differences were significant at $p > 0.99$.

Note that every learning based algorithm outperformed the LMS method of [5] in terms of accuracy. On the other hand, the prediction time of the new models is higher than that of the LMS method, except for ANN0 that has the same complexity as the LMS method (both involve a single matrix product, with a given fixed matrix). The case of the linear model is particularly interesting anyway. Its superior performance as compared with the LMS based method indicates the suboptimality of the optimization criterion employed by the LSM method. Since we felt that the additional computational complexity of the more complex neural network models do not justify their usage, the ANN0 model was used in all the subsequent experiments.

3.2.2 Tracking Results

Data. The above algorithm was tested on several real-world tracking problems. In one of the most difficult scenarios a fast moving hand was tracked in a highly cluttered environment through 175 frames of a video sequence (corresponding to almost 6 seconds). In this sequence the hand travelled 9 times from one corner of the image to another one. The

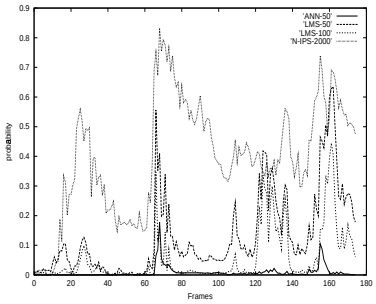

Figure 2: Probability of loosing the object.

resolution of the images in the sequence was 240×180. The pose of the hand to be tracked was identified by refining the poses returned by one of the tracking algorithms manually.

Algorithms. The number of support points of the contour was set to 30. Four variants of the tracking algorithms were compared: *(i)* plain N-IPS with 2000 particles (N-IPS-2000), *(ii)* LS-N-IPS with LMS-based local search with 100 particles (LMS-100), *(iii)* LS-N-IPS with LMS-based local search with 50 particles (LMS-50), *(iv)* and LS-N-IPS with the local search implemented by the ANN0 network trained as described in the previous subsection and with 50 particles (ANN-50).

Results. By running Monte-Carlo experiments, we have measured the *probability of loosing lock* of the object. Each algorithm was run 1000 times and for each frame the probability of loosing lock was estimated by counting the average number of times when the average distance of the support points of the estimated target contour to the true target contour was larger than 5 pixels. The resulting curves are shown in Figure 2 as a function of the frame index. It should be clear from the figure that N-IPS-2000 performed the worst, followed by LMS-50, followed by LMS-100. The best results were obtained for ANN-50. N-IPS-2000 has the tendency to loose the object within the first second with probability 0.3, whilst the other algorithms are capable of tracking the object except for a few really difficult cases.[7]

In the same set of tracking sessions, the *effective sample size* (ESS) of each of the particle filters was measured. The effective sample size measures the uniformity of the weights of the particles and is defined by $1/(\sum_{i=1}^{N}(w_t^{(i)})^2)$ [1]. The larger the ESS is, the more particles are concentrated in the neighborhood of the object to be tracked and thus, the better is the chance of the algorithm to respond to fast changes (e.g. illumination changes, or rapid changes in the direction of motion). The left-hand-side subfigure of Figure 3 shows the ESS for all the algorithms as a function of frame index. The previous ordering (N-IPS-2000$\ll$LMS-50$\ll$LMS-100$\approx$ANN-50) applies again. The figure indicates that all algorithms respond as expected in the difficult scenarios: the ESSs of all algorithms increased around these points. Among all the algorithms, in these regions ANN-50's ESS is the largest as compared to the number of particles the algorithms use: at the difficult parts ANN-50's relative ESS reaches ranges from 50% to 70%.

Tracking accuracy was also measured as a function of time. Results are shown in the right-hand-side subfigure of Figure 3. These curves are averaged over the 1000 runs, and are smoothed with a median filter working in the time domain. Again, the advantage of ANN-50 should be clear: its accuracy is clearly superior to that of even LMS-100.

[7] Around frames 70 and 158 the hand is in the lower dark part of the image, moving fast and also changing its direction of motion.

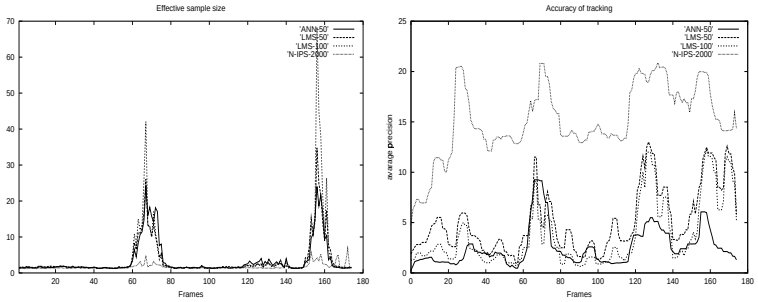

Figure 3: Effective sample sizes (left subfigure) and tracking accuracy (right subfigure) as a function of frame number.

4 Conclusion and Discussion

In this article, we have proposed a data oriented approach to boost the performance of LS-N-IPS. The new method was shown to reduce the number of particles needed for tracking significantly and to increase robustness as compared with the previous method proposed in [5]. In particular tracking speed of 60 frames$/$second was achieved whilst at the same time the algorithm proved to be more robust than the previous algorithm.

The new algorithm minimizes the expected distortion in the configuration space, whilst in the previous approach the local search was implemented by choosing a configuration that minimized the error of fitting the corresponding spline contour to some feature points in the vicinity of the predicted contour. The difference of these criteria explains the performance improvement.

One weakness of the approach is that the neural nets underlying the local search has to be trained for each class of contours of interest. However, this is counterbalanced by the fact that even linear mappings suffice to achieve good results and training data is available at virtually no cost. In conclusion, we believe that LS-N-IPS with learnt local search operators represent an important step towards the implementation of highly efficient tracking systems.

References

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

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

[3] Riedmiller Martin and Braun Heinrich. RPROP-a fast adaptive learning algorithm. *Proc. of ISCIS VII.*, 1992.

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

[5] Csaba Szepesvári and Péter Torma. Vision based object tracking using LS-N-IPS. *In Proc. ISAP'01*, pages 277–282, 2001.

[6] Péter Torma and Csaba Szepesvári. LS-N-IPS: an improvement of particle filters by means of local search. *In Proc. Non-Linear Control Systems(NOLCOS'01)*, 2001.

Figure 4: Tracking hand in clutter with 30 particles and the ANN0 method. Black contours show particles having high observation likelihoods, whilst the white contour shows the predicted contour of the hand. When no black contour is given, then typically there is only a single particle that has significantly higher likelihood than the others.