A Particle Swarm Optimization inspired tracker applied to visual tracking

Christophe Mollaret, Frédéric Lerasle, Isabelle Ferrané, Julien Pinquier

A PARTICLE SWARM OPTIMIZATION INSPIRED TRACKER APPLIED TO VISUAL TRACKING

Christophe Mollaret[⋆†], *Fréderic Lerasle*[‡,†], *Isabelle Ferrané*[⋆], *Julien Pinquier*[⋆]

[⋆]Université de Toulouse, UPS, IRIT {mollaret, ferrane, pinquier}@irit.fr
[‡]CNRS, LAAS, {cmollare, lerasle}@laas.fr
[†]Université de Toulouse, UPS, LAAS

ABSTRACT

Visual tracking is dynamic optimization where time and object state simultaneously influence the problem. In this paper, we intend to show that we built a tracker from an evolutionary optimization approach, the PSO (Particle Swarm optimization) algorithm. We demonstrated that an extension of the original algorithm where system dynamics is explicitly taken into consideration, it can perform an efficient tracking. This tracker is also shown to outperform SIR (Sampling Importance Resampling) algorithm with random walk and constant velocity model, as well as a previously PSO inspired tracker, SPSO (Sequential Particle Swarm Optimization). Experiments were performed both on simulated data and real visual RGB-D information. Our PSO inspired tracker can be a very effective and robust alternative for visual tracking.

***Index Terms*—** visual tracking, particle swarm, particle filter, video analysis, RGB-D sensors

1. INTRODUCTION

Visual tracking is one of the most studied topics in the research field of video analysis [1]. It has many uses such as video surveillance, vision-based control and human-computer interface. These techniques are more and more used in human-Machine Interaction (HMI) [2], which needs to fuse a lot of visual information which is given by many sensors and/or algorithms, which will also be considered as sensors. For our application, we need to characterize the user's will to communicate with a machine also known as "awareness" in Human-Machine Interaction community [3].

In visual tracking, two kinds of tracking algorithms are mainly used. They are Kalman filters and Particle filters. Kalman filters are widespread in industry and research for multiple use such as 3D-modeling, path following and object tracking. Chen in [4] has made a survey on different techniques and algorithm implementations. However the original Kalman algorithm is limited to linear problems and becomes more complex if we need to apply variants of this algorithm (unscented, extended) on multimodal and non-linear cases. Meanwhile, particle filtering is a very simple algorithm adapted to multimodal and non-linear target tracking. Nevertheless, it requires an increase in the number of particles to perform efficient estimation, raising its complexity exponentially with the dimension of the estimated state vector. This effect, known as dimensionality curse, is very important in particle filtering techniques. This becomes a problem in visual tracking task since we need to approach real time tracking by decreasing the computing time, i.e. decreasing the number of particles.

For many years, Particle Swarm Optimization (PSO) has been gaining attention [5], [6], since this algorithm is able to solve non-linear, multimodal and high-dimensional optimization problems. Contrary to other particle-based algorithms, particles interact with each other with a "social" and "cognitive" component in the update equation. This leads to finer particle behaviour during the process of optimization. Furthermore, the computation cost remains low since only one computation of fitness function per iteration and per particle is needed, which is, usually, the bottle-neck part in visual tracking problems.

In this paper, a new approach on PSO based tracking is proposed, merging particle filter versatility with a lower computational cost, adapting to real time constraints. The idea is also to keep the "PSO behaviour" of particles which seems more interesting than the usual "particle filter behaviour". Because of its social and cognitive components, particles of PSO algorithm interact with each other, contrary to particle filter. We think that this behaviour could lead to a more efficient estimation of the target as there is no particle degeneration in PSO. Therefore, a resampling step becomes needless and we can boost the algorithm execution. Finally, like the particle filter with constant velocity model, the PSO has an implicit estimation of state vector speed, which seems to model the system dynamics better than a standard particle filter. But contrary to the former particle filter, we do not need to double the state vector dimension to perform the estimation, since only the speed in PSO has a random component.

This work is partly supported by a grant from the ANR (Agence Nationale de la Recherche) with reference ANR-12-CORD-003.

In a first section we will present some related works, then, we will focus on the formalism of the Particle Swarm Optimization algorithm and our new PSO-based tracker. Next, we will implement and compare four algorithms on simulated data to evaluate the robustness and efficiency of our new tracker. Finally we will perform tracking on real recorded RGB-D data with ground truth in our applicative context, human awareness characterisation in HMI. We used two modalities outputs which worked for us as "sensors". The first one was provided by OpenNI/NITE library [7] for the shoulder orientation and the detection of head position. The second sensor was developed by Fanelli in [8] for research purposes and enables the full head pose detection on depth informations.

2. RELATED WORKS

Particle Swarm Optimization algorithms has been used in a wide range of application for many years. Poli in [5] proposed an overview of them, and it appears to have been implemented for parameter tuning in particle filter, fuzzy systems, video analysis, and image analysis. This algorithm presents good performances with high dimension problems, and seems to suffer less from dimensionality curse. Moreover, it is computationally lower than Sequential Monte-Carlo (alias SMC) methods such as particle filtering applied in optimization problems. However, this algorithm belongs to the meta-heuristics branch such as genetic algorithms, and its convergence cannot be proven.

For past years some investigations have aimed at adapting PSO in a tracking framework. Zhang first implemented it in [9], where he added PSO iterations in a particle filtering framework. However, the computation cost is proportional to iteration numbers and the algorithm finally become much more costly than particle filtering techniques, despite the fact that it performs a more precise tracking. Recently, Li in [10] replaced the resampling step in particle filter by one PSO iteration. This is as computationally costly as particle filters. However, they lose the formalism of the resampling step which has to conserve the estimated MMSE while repositioning particles.

Particle filter is also one of the only particle based tracker, and is widely used in the vision literature for visual tracking problems. There are many variants of this algorithm and one of them is used by Sedai in [11] for human pose tracking which is getting closer to our context. The Annealed Gaussian Process Guided Particle Filter (AGP-PF) that they used is better than basic Particle Filter, but much heavier computationally speaking and complicated to implement.

Thus, a PSO-based tracker is created, more in original PSO algorithm respect, mainly in terms of particles behaviour. It is also wanted lighter in computational resources than a particle filter while being more efficient. Finally, the algorithm aims to remain simple, while some improved Particle Filter implementations become very heavy in code and processing cost.

3. FORMALISM

Four algorithms are compared for this work. Two SIR trackers with two different dynamic models, and two PSO based trakers namely [9] and ours.

SIR-RW: The first one, the more usual, is a SIR-RW where "RW" stands for Random Walk dynamic model.

SIR-CV: The second one is a SIR-CV, where "CV" abbreviates Constant Velocity dynamic model. Contrary to the previous one, the state vector dimension is doubled in order to add its first derivative as a component of the state vector. Thus, the state vector velocity is also estimated. When $s_t^{(i)}$ is the state vector in SIR-RW algorithm, we have to model the SIR-CV state vector by $x_t^{(i)} = [s_t^{(i)}\ \dot{s}_t^{(i)}]$, where t is time and i the particle index. The update equations become $\dot{s}_t^{(i)} = \dot{s}_{t-1}^{(i)} + v_t$ where v_t is a gaussian noise and $\dot{s}_t^{(i)}$ the state vector velocity part. $s_t^{(i)} = s_{t-1}^{(i)} + \omega * \dot{s}_{t-1}^{(i)} + w_t$ where w_t is gaussian noise as well, $s_t^{(i)}$ is the state vector position part and ω is a multiplicative constant modelling inertia.

SPSO: The third algorithm is the SPSO (Sequential Particle Swarm Optimization) as formalised by Zhang in [9], and its implementation will not be recalled in this paper.

TPSO: Finally, the fourth one is our modified PSO for tracking (alias TPSO). This algorithm is described in details in this section after a short reminder on PSO algorithm for a better understanding of our contribution.

In our approach, optimization and tracking are partially the same problem: in optimization, the algorithm has to converge to a fixed global maximum in a certain number of iterations. However, in tracking, also called dynamic optimization, the algorithm has to converge to a moving global maximum (the target), and iterations are represented by frames (one frame = one iteration). Regarding this, SPSO is not a proper tracker since it iterates more than once for each frame. Considering this, we took the PSO and modified the parts were the target is mentionned in the equations to take into account the target motion. In algorithm 1, the PSO algorithm principe is reminded. This algorithm being adapted for regular optimization and not dynamic optimization, algorithm 2 shows our adaptations to tracking.

In algorithms 1 and 2, $U(a, b)$ is a uniform distribution between a and b, N is the number of particles, b_{lo} and b_{up} are lower and upper boundaries of search space, $f(.)$ is a fitness function, or likelihood for SIR algorithms. r_p and r_g are factors that respectively ponderate social and cognitive terms randomly in the next line. ω models particles inertia. ϕ_p and ϕ_g are constant weights of cognitive and social factors respectively. The criterion can be a maximum number of iterations or a threshold on the MAP estimate fitness $f(g)$.

Algorithm 1: Particle Swarm Optimization (PSO)

Result: MAP estimate : $\hat{x} = g$

1 **for** *i=0 to N (initialization)* **do**
2 $x_i \sim U(b_{lo}, b_{up})$
3 $p_i \leftarrow x_i$, $g \leftarrow argmax(f(p_i))$
4 $v_i \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$
5 **while** *criterion is not met (iterations for optimization)* **do**
6 **for** *i=0 to N* **do**
7 $r_p, r_g \sim U(0, 1)$
8 $v_i \leftarrow \omega v_i + \psi_p r_p(p_i - x_i) + \psi_g r_g(g - x_i)$
9 $x_i \leftarrow x_i + v_i$
10 **if** $f(x_i) > f(p_i)$ **then**
11 $p_i \leftarrow x_i$
12 if$(f(p_i) > f(g))$ $g \leftarrow p_i$

In our algorithm 2, lines 8 and 10 of PSO are modified to add a "prediction" component represented by a dynamic model, and there is now only one iteration per frame as explained before. The $d(.)$ function represents the filter dynamic model. For our experiments, a random walk was considered($d(p_i) = p_i + w$ where w is a gaussian noise). We could have set the dynamics model differently like the constant velocity model used in the second particle filter presented in this paper but chose not to. We added this function assuming that the optimal positions relative p_i, and global g change, as the target position changes each frame. Then, $f(p_i)$ has to be re-computed as it will have moved since the last measurement. These two minor changes allowed us to create a robust dynamic optimizer (or tracker), using PSO formalism, keeping the algorithm extremely simple.

4. EVALUATIONS ON SYNTHETIC DATA

Our new tracker was then processed on synthetic signals while one free parameter was changed at a time, to evaluate and compare its accuracy versus the other algorithms. The synthetic signal is a multidimensionnal signal constructed from concatenation of sinus functions, fast value changes and straight lines, in order to make a challenging target to follow.

Our observation model is a multivariate gaussian with a diagonal covariance matrix. The measurements are samples of synthetic signals added to a gaussian random noise with the same covariance matrix as the observation model, to make this optimal. The similar parameters of the four trackers are set equals: number of particles N, process noise σ in dynamic model for the four algorithms; ω for SPSO, TPSO and SIR with constant velocity model; ψ_p and ψ_g for SPSO and TPSO.

The following effects of parameters are evaluated on tracking process: particle number, noise power added to the observations and finally, dimension of the state vector. More-

Algorithm 2: PSO based Tracker

Result: MAP or MMSE estimate for each frame

1 **for** *i=0 to N (initialisation)* **do**
2 $x_i^{(0)} \sim U(b_{lo}, b_{up})$
3 $p_i^{(0)} \leftarrow x_i^{(0)}$, $g^{(0)} \leftarrow argmax(f(p_i^{(0)}))$
4 $v_i^{(0)} \sim U(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$
5 **for** *t=1 to frame M (iterations on video sequence)* **do**
6 **for** *i=0 to N* **do**
7 $r_p, r_g \sim U(0, 1)$
8 $v_i^{(t)} \leftarrow \omega v_i^{(t-1)} + \psi_p r_p(d(p_i^{(t-1)}) - x_i) +$ $\psi_g r_g(d(g^{(t-1)}) - x_i)$
9 $x_i^{(t)} \leftarrow x_i^{(t-1)} + v_i^{(t)}$
10 **if** $f(x_i^{(t)}) > f(p_i^{(t-1)})$ **then**
11 $p_i^{(t)} \leftarrow x_i^{(t-1)}$
12 if$(f(p_i^{(t)}) > f(g^{(t-1)}))$ $g^{(t)} \leftarrow p_i^{(t)}$
13 MAP estimator : $\hat{x}^{(t)} = g^{(t)}$ or
14 MMSE estimator : $\hat{x}^{(t)} = \sum_{i=0}^{N} \frac{f(p_i)^{(t)}}{\sum_{i=0}^{N} f(p_i)^{(t)}} x_i^{(t)}$

over, we computed the $Neff$ estimate (Number of efficient particles), in SIR and TPSO. In SIR, it is given by the equation $Neff = 1/(\sum_{i=1}^{N} w_i^2)$. In TPSO, an equivalent to $Neff$ is the equation $Neff = (\sum_{i=1}^{N} f(x_p^{(i)}))^2 / \sum_{i=1}^{N} f(x_p^{(i)})^2$. We did not implement this estimator in SPSO as there is more than one $X_p^{(i)}$ estimation per iteration and, therefore, no true equivalent to usual $Neff$.

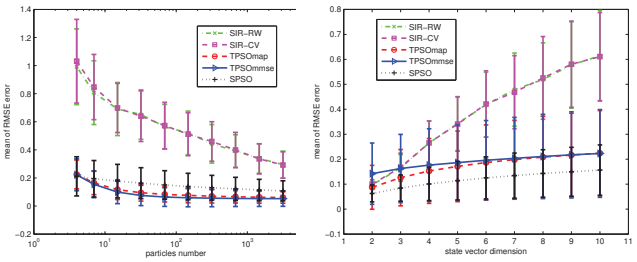
In Fig. 1. are displayed the RMSE error means of 50 runs. TPSOmap and TPSOmmse are respectively MAP and MMSE estimator of the same TPSO run due to the stochastic nature of the process.

Noise: First we can see (Fig. 1 (c)) that our tracker reacts better to noise, with a tracking error divided by 2 for low SNR, where SNR is Signal to Noise Ratio in dB: $SNR = 10log(PowerSignal/PowerNoise)$.

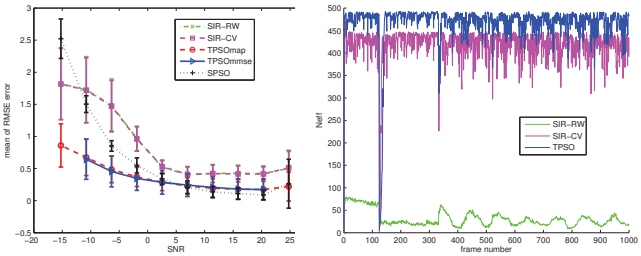
Particles: Then, we can notice (Fig. 1 (a)) that the number of particles can be divided by 5 while keeping the same precision as SIR algorithms. Thus, even tough we have to compute likelihood two times by iteration and particle, we can have a reduced complexity by reducing tremendously the number of particles.

Dimension: Moreover, we can see (Fig. 1 (b)) that our tracker also better reacts when state vector dimension increases, and estimates the target state vector with a reduced variance in high dimensions.

Neff: Finally, we can explain these results with $Neff$ estimator for TPSO and SIR algorithms (Fig. 1 (d)). For 500 particles, SIR-RW $Neff$ degenerates to 50, while SIR-CV acts better with 450 particles, and ours achieve the best with



(a) Particles number (b) State vector dimension



(c) Signal to Noise Ratio (SNR) (d) Neff mean for one track

Fig. 1. Simulation results

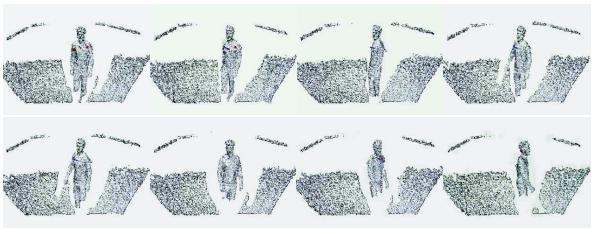


Fig. 2. RGB-D sequence capture: Red markers represent shoulder detection, green markers are head pose detections.

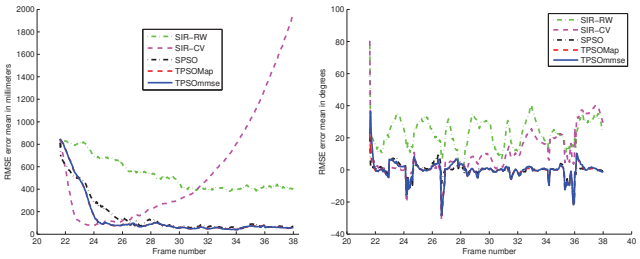
490 particles. This result is very interesting considering that our algorithm has a simpler dynamic model than SIR-CV and does not need any resampling step to avoid degeneration.

5. APPLICATION TO VISUAL TRACKING

In our context (cf. Fig. 2), we want to track head pose (yaw, pitch, roll), and shoulder orientation in space as they are key information of machine awareness. Full video sequences can be found here [1]. These modalities are given by OpenNI [7] for shoulder orientation and head position, and Fanelli [8] for head pose. However we had a problem with angles as they vary from -180 to +180 and can jump between these extremes values. We needed to take care of it during sampling phase. This led us to a 7-state vector $s = [x\ y\ z\ \theta_{head}\ \phi_{head}\ \psi_{head}\ \theta_{shoulders}]$.

The observation model is a multivariate gaussian model with a diagonal covariance matrix $diag(\Sigma) = [diag(\Sigma_{position})$ $diag(\Sigma_{angles})]$. The dynamic model is a random walk for all filters, excepted for the SIR-CV, where there is a speed component. The process gaussian noise is modelled by a diagonal covariance matrix as well. Parameters defined in previous sections are set to: $\omega = 0.9$, $\psi_p = 0.8$, $\psi_g = 1$, and $N = 100$ particles. Finally, data acquisition was performed by a calibrated RGB-D sensor. The ground truth was created with a marker-based Motion Capture (Mocap) disposal which is also calibrated and synchronized temporally with the RGB-D sensor.

[1] https://projects.laas.fr/riddle/riddle-project/papers/



(a) Position tracking error (b) Angular tracking error

Fig. 3. Tracking results

A set of 100 runs for each tracker were carried out on the same path, and results are plotted on Fig. 3, representing the tracking error means. The plot (a) displays RMSE mean position error in mm, and (b), the RMSE mean angular error in degrees, since a global error on position and angles would have no sense. We can see that our algorithm achieved better performances on both position and angles, while particle filtering techniques had problems to perform an accurate tracking jointly on position and angles. These results came to comfort our simulation results and showed that our tracker was a suitable solution for robust tracking, keeping a low computational cost with only 100 particles and a relative complexity of 2/5 of the SIR, based on how often the observation function is called in a loop.

6. CONCLUSION

In this paper, we proposed a new tracking framework based on Particle Swarm Optimization formalism and, demonstrated on both, synthetic and real data, that our algorithm outperforms SIR algorithm while remaining as simple in implementation, and faster than SPSO. Our algorithm also seems to be more robust to state vector dimensionality, noise in observations, and an efficient tracking can be performed with fewer particles, as each one of them carries more information. All of these aspects present our PSO-inspired tracker as a promising alternative for visual tracking and others applications.

In future work, we intend to evaluate the influence of each TPSO parameter on tracking performance and will try to apply it to multi-target tracking.

7. REFERENCES

[1] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, and Afshin Dehghan S. Calderara, "Visual tracking: An experimental survey," *Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1, October 2013.

[2] B.-J. Chen, C.-M. Huang, and L.-C. Fu, "Real-time visual tracking with adaptative particle filter for human-machine interaction," *SICE*, pp. 1344–1349, September 2011.

[3] J. L. Drury, J. Scholtz, and H. A. Yanco, "Awareness in human-robot interactions," *Systems, Man and Cybernetics*, vol. 1, pp. 912–918, October 2003.

[4] S. Y. Chen, "Kalman filter for robot vision: A survey," *Industrial Electronics*, vol. 59, pp. 4409–4420, November 2012.

[5] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," in *Journal of Artificial Evolution and Applications*, 2008.

[6] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: A brief survey," *Systems, Man and Cybernetics*, vol. 41, pp. 262–267, March 2011.

[7] PrimeSense, "Openni/nite, open-source sdk for 3d sensors, http://www.openni.org/," .

[8] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *CVPR*, 2011.

[9] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu, "Sequential particle swarm optimization for visual tracking," in *CVPR*, 2008, pp. 1–8.

[10] X. Li, W. Chen, and Z. Shang, "A video tracking method based on niche particle swarm algorithm-particle filter," in *WCICA*, 2012.

[11] S. Sedai, M. Bennamoun, and D. Q. Huynh, "Robust perception of an interaction partner using depth information," in *Image Processing*, November 2013, vol. Image Processing, pp. 4286 – 4300.