PCA-enhanced stochastic optimization methods

Alina Kuznetsova, Gerard Pons-Moll, and Bodo Rosenhahn*

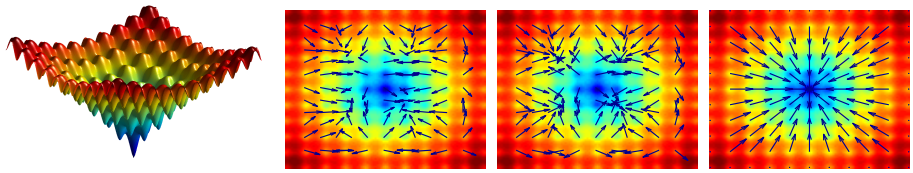
Institute for Information Processing (TNT),
Leibniz University Hanover, Germany
{kuznetso,pons,rosenhahn}@tnt.uni-hannover.de

Abstract. In this paper, we propose to enhance particle-based stochastic optimization methods (SO) by using Principal Component Analysis (PCA) to build an approximation of the cost function in a neighborhood of particles during optimization. Then we use it to shift the samples in the direction of maximum cost change. We provide theoretical basis and experimental results showing that such enhancement improves the performance of existing SO methods significantly. In particular, we demonstrate the usefulness of our method when combined with standard Random Sampling, Simulated Annealing and Particle Filter.

1 Introduction

A large number of computer vision problems requires to optimize a cost function that depends on a high number of parameters. When the cost function is convex, the global optimum can be found reliably. Unfortunately, many problems are hard or even impossible to formulate in convex form. It is well known that in such cases typical gradient-based local optimization methods easily get trapped in local minima and therefore stochastic optimization algorithms must be employed. In general terms, stochastic optimization algorithms consist of generating random proposals, or particles, to find regions of parameter space with low costs. However, the number of particles needed to reach the global minimum with high probability grows exponentially with the dimension of parameter space. To improve sampling efficiency in high-dimensional spaces a common strategy is to follow the cost function gradient of good samples. Unfortunately, this has two major limitations. Firstly, for many cost functions used in computer vision there is no analytic expression for the gradient and approximating it by finite differences is computationally expensive. Secondly, the gradient is a very local measure of the cost function landscape that ignores the underlying global shape. Hence, this approach is susceptible to get the particles trapped in local minimum. Inspired by methods based on Hamiltonian Monte Carlo (HMC) [14], we introduce PCA-based Stochastic Optimization (PCA-SO). We propose to improve the particles in terms of cost function based on the landscape geometry, constructed from already computed neighboring samples. In that sense,

* This work has been partially funded by the ERC within the starting grant Dynamic MinVIP



(a) (b) (c) (d)

Fig. 1: Shift directions: (a) Ackley function with many local minimum, (b) directions of maximal function decrease - oppositely directed to the gradient of the function, (c) shift direction computed from a small neighborhood and (d) shift direction computed from a bigger neighborhood. As it can be observed, for small neighborhoods (c) our method computes directions that are parallel to the local gradients and for bigger neighborhoods (d) the computed directions capture more global properties of landscape.

our method is related to particle swarm optimization methods [10]. Specifically, we approximate the cost function in a neighborhood by a hyperplane, efficiently computed using PCA. This results in the direction of the maximum cost function variance in a bigger neighborhood, while for a small enough neighborhood this direction coincides with the local gradient. Then, the step size is chosen in the direction of the maximal cost function change in the parameter space. In several experiments, we show that this modified stochastic search scheme results in faster convergence and reduced variance of final solution.

2 Previous work

Perhaps one of the most widely used stochastic inference methods in vision is the Particle Filter (PF) [9], which can be seen as one instance of importance sampling. For many applications however, the dimension of parameter space is too high and a huge number of samples are needed to approximate the posterior or even to find a good maximum a posteriori (MAP) approximation. An example of such application, for instance, is human pose estimation from video in which the number of parameters to estimate ranges from 20 to 60. One way to make the search more efficient is to use annealing schemes. The traditional Simulated Annealing (SA) [11] starts from a set of hypotheses and decides to move the system to a new state with an acceptance probability that depends on the optimization gain and the temperature T of the system. During the selection of random neighboring states, the temperature is gradually decreased during the iterations. The idea is that the choice between the previous and current solution is almost random (at the beginning) when T is large, but it increasingly selects closer located "downhill" samples as T approaches zero [5, 6]. Although in principle such schemes explore the search space more efficiently, very often all samples concentrate around a single local optimum. Another commonly used strategy is to follow the gradient of good samples during stochastic search [3,

16, 2], this has been shown to reduce the effect of volume wastage which occurs when a large number of particles are rejected. When the gradient cannot be computed covariance matrix adaptation can be very effective [8, 7]. Closely related are HMC methods [14, 4]. The basic idea behind HMC is to construct a Hamiltonian in which the potential energy is proportional to the cost function and the position corresponds to the variables of interest. Then a momentum is added artificially to the particles. Intuitively, this can speed up convergence and increase robustness to local minimum because the momentum of the particles allows them to go downhill faster and continue when they encounter uprising slopes. In HMC methods the momentum is calculated independently for every particle and is closely related to following the gradient direction [4]. By contrast, we compute the particle shift based on local neighborhood of a particle. This allows to better capture the underlying landscape of the cost function, smoothing out local irregularities. Other approaches combining PCA and Stochastic optimization perform sampling in PCA space instead of performing sampling in the original space [12], which is completely different to our approach.

2.1 Contributions

In this work, we propose a simple but very effective modification for stochastic optimization that adds robustness to local minimum and speeds up convergence. One of the main advantages of the proposed method is that it can be easily integrated to any stochastic optimization technique. We demonstrate the benefits of PCA-SO in three different methods, namely plain Random Sampling (RS), Simulated Annealing (SA) and Particle Filter (PF), applied to typical vision problems, such as image registration and tracking.

3 Stochastic optimization

In this section we briefly describe the basic ingredients of a global stochastic optimization algorithm. Most stochastic search algorithms consist of three steps, namely *weighting*, *selection* and *mutation*. In the weighting step, the cost function is evaluated and the particles are given a proportional weight. This is usually the most time consuming step. In the selection step, the weighted particles are accepted or rejected with some probability that can depend on their weight, optimization gain and temperature, if an annealing schedule is used. In the mutation step new candidate locations are generated from the current particles. A commonly used heuristic is that particles give offspring proportional to their current weights so that more computational resources are allocated for promising candidate locations. One of the main advantages of PCA-SO is that it can be easily integrated in the mutation step of any stochastic optimization algorithm.

4 PCA-based Stochastic Optimization

Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a multivariate cost function where d is the dimension of the parameter space. Optimization entails finding the parameter vector $\boldsymbol{x} =$

$(x_1, \ldots, x_d)$ that minimizes the function f:

$$\boldsymbol{x}^* = \arg\min_{x_1,\ldots x_d} f(x_1, \ldots, x_d). \tag{1}$$

The function f defines a hyper-surface in $\mathcal{S} \subset \mathbb{R}^{d+1}$, whose points are $\boldsymbol{y} = (x_1, \ldots, x_d, z)$ with $z = f(x_1, \ldots, x_d)$. Equivalently, we can represent the points in the hyper-surface as the zero level-set of the function $F(x_1, \ldots, x_d, z) = f(x_1, \ldots, x_d) - z$. The gradient of F is trivially related to the gradient of the original cost function f gradient by

$$\nabla_{\boldsymbol{y}} F = \left(\frac{\partial F}{\partial x_1}, \ldots, \frac{\partial F}{\partial x_d}, \frac{\partial F}{\partial z} \right) = (\nabla_{\boldsymbol{x}} f, -1)\,. \tag{2}$$

Obviously, since the gradient must be orthogonal to the level sets of the function $F(x_1, \ldots, x_d, z) = 0$, $\nabla_{\boldsymbol{y}} F$ is perpendicular to the hyper-surface defined by $f(\boldsymbol{x})$. The gradient of the cost function $\nabla_{\boldsymbol{x}} f$ provides the direction of local maximum increase of the cost function f. In principle, this direction can be used to shift the particles which can speed up convergence in many situations. However, this approach is susceptible to get the particles trapped in local minima, because particles falling into local minima will not be shifted since the gradient vanishes at those points.

Therefore, we propose to compute the direction of maximum increase (or decrease, depending on the optimization direction) of the cost function in a bigger neighborhood. Specifically, for every particle $\boldsymbol{y}^i = (\boldsymbol{x}^i, z^i)$ we take all the samples inside a ball of radius r, $\mathcal{N}_i[\boldsymbol{y}^i] \triangleq \{\boldsymbol{y} \in \mathcal{S} \mid d(\boldsymbol{x}^i, \boldsymbol{x}) \leq r\}$. PCA provides means to compute the orthogonal directions in which the variance of the sample set is maximized. Conceptually, it is desirable to shift the particle in the direction $\delta\boldsymbol{x} \in \mathbb{R}^d$ of parameter space that maximizes the cost function change. Given the center of the neighborhood $\boldsymbol{\mu}^i = \frac{1}{N} \sum_{\boldsymbol{y}_n \in \mathcal{N}_i} \boldsymbol{y}_n$, the directions provided by PCA are the eigenvectors $(\boldsymbol{\varphi}^i_1, \ldots, \boldsymbol{\varphi}^i_{d+1})$ of the sample covariance matrix $\boldsymbol{\Sigma}^i = \frac{1}{N-1} \sum_{\boldsymbol{y}_n \in \mathcal{N}_i} (\boldsymbol{y}_n - \boldsymbol{\mu}^i)(\boldsymbol{y}_n - \boldsymbol{\mu}^i)^T$ with eigenvalues $\sigma^i_1 > \sigma^i_2, \ldots > \sigma^i_{d+1}$ corresponding to the variance of the sample set, projected into the principal directions. It can be shown that the direction $\boldsymbol{\varphi}^i_{d+1}$ of the smallest variance for a small enough neighborhood of a differentiable function is parallel to the normal $\hat{\boldsymbol{n}}^i$ to the surface at the point $\boldsymbol{y}^i$:

$$\hat{\boldsymbol{n}}^i = \left. \frac{\nabla_{\boldsymbol{y}} F^T}{\|\nabla_{\boldsymbol{y}} F\|} \right|_{\boldsymbol{y}^i} = \gamma \frac{\boldsymbol{\varphi}^i_{d+1}}{\|\boldsymbol{\varphi}^i_{d+1}\|}, \qquad \gamma = \pm 1 \tag{3}$$

Therefore, the linear space spanned by $(\boldsymbol{\varphi}^i_1, \ldots, \boldsymbol{\varphi}^i_d)$ forms the tangential hyperplane to the surface $\mathcal{S}$ at the point $\boldsymbol{y}^i$, based on the neighborhood $\mathcal{N}_i[\boldsymbol{y}^i]$. Thereby, at the k-th iteration particle $\boldsymbol{y}^i$ is shifted by $\delta\boldsymbol{x}^i$

$$\boldsymbol{x}^i_{k+1} := \boldsymbol{x}^i_k - \lambda \delta \boldsymbol{x}^i_k \qquad\qquad \delta \boldsymbol{x}^i_k := \frac{\pi_{d+1}(\boldsymbol{\varphi}^i_{d+1})}{\|\pi_{d+1}(\boldsymbol{\varphi}^i_{d+1})\|} \tag{4}$$

where $\pi(\cdot)_j$ is the projection operator that drops the j-th component of a vector and λ is a step size parameter. For a small enough neighborhood radius r $\delta\boldsymbol{x}$

is parallel to the cost function gradient $\nabla_{\boldsymbol{x}} f$. Therefore, methods that combine stochastic search with gradient descent [13, 3, 4, 16, 2] are special cases of PCA-SO. As the ball radius r increases, Eq. 3 does not hold anymore and the shift vector reflects the direction of maximum function change in a bigger neighborhood. Notably, this provides a direction that is robust enough to make a particle jump over spurious local minimum, see Fig 1.
To summarize, the PCA-SO algorithm consists of the following steps:

1. Sample an initial set of particles randomly and evaluate the cost function.
2. Sample one or more new particles, find already evaluated particles in their neighborhood, compute shift directions and shift steps (see (4)), using already evaluated particles.
3. Shift the particles in the found directions and evaluate the cost function at the new locations.
4. Accept the particles with improved cost function values and add to the initial set of particles
5. Go to step 2.

A time-consuming step of our approach can be finding the neighborhood of every particle. If necessary, K-D trees can be used [1], since they allow finding neighbors in $\log(N)$ time. Since shift direction and step varies smoothly for neighboring particles, a more efficient alternative is to cluster the particles and give the same shift to every particle within the same cluster.

5 Experiments

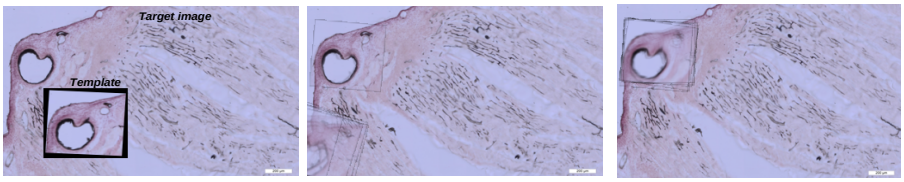
In this section, we show the benefits of PCA-SO integrated in three different stochastic optimization/sampling methods, namely random sampling (RS), simulated annealing (SA) and particle filtering (PF). We apply our technique to two different vision applications: image registration and target tracking. To evaluate the effectiveness of a given optimization method we report two measures: (i) the number of iterations needed to reach a good approximation of the solution, and (ii) the value of the cost function after a fixed number of iterations.

5.1 Image registration using random sampling

We tested our approach for image registration using random sampling (RS). In Fig. 2a the images used for registration are presented. They are histology images of heart tissue cross-sections that need to be registered to generate 3D models. Assuming an affine transformation between images

$$\boldsymbol{T} = \begin{pmatrix} s_x \cos(\theta) & s_x \sin(\theta) & t_x \\ -s_y \sin(\theta) & s_y \cos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{5}$$

we need to estimate rotation angle θ, scaling s_x, s_y and translation t_x, t_y parameters. Let $I(\boldsymbol{p})$ denote the target image value at point $\boldsymbol{p}$ and $I_t(\boldsymbol{p})$ denote the



(a) (b) (c)

Fig. 2: Image registration results: (a) target image and template; (b) standard RS registration; (c) PCA-SO RS registration

template image value at point $\boldsymbol{p}$. Rectangle $\mathcal{R}$ defines the region to be matched. Here $|\mathcal{R}|$ denotes the number of pixels in the rectangle. The cost function $J(I_t, I)$ is defined as following:

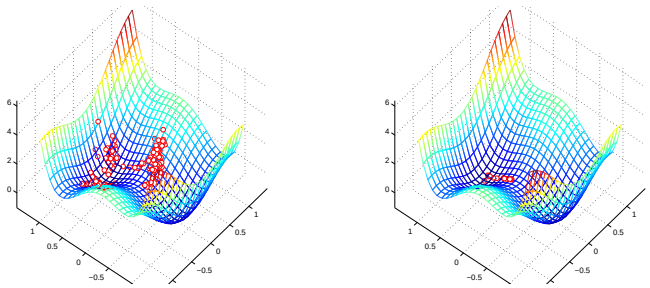
$$J(I_t, I) = \frac{1}{|\mathcal{R}|} \sum_{\boldsymbol{p} \in \mathcal{R}} (I(\boldsymbol{p}) - I_t(\boldsymbol{T}\boldsymbol{p}))^2 \tag{6}$$

As can be seen, no analytical expression for derivatives of J exists. Integration of our enhancement is straightforward: sample small number of particles; for each new sample use already evaluated particles to build local neighborhood and improve it with respect to the cost function; accept it if the cost function value improved. As it can be seen in Table 1, PCA-SO reduces both the number of required iterations to reach a good fit and the quality of the fit after the fixed number of iterations.

Table 1: Comparison between standard random sampling and its enhancement using PCA-SO; cost function values are in $[9.50, 20.30]$; results are aggregated from 50 independent runs of optimization till the termination criteria

termination criteria	without PCA (mean, $\pm$ std)	with PCA (mean, $\pm$ std)
number of iterations		
value reached ($J \leq 13$)	22 ± 15	17 ± 11
value reached ($J \leq 12$)	43 ± 13	37.66 ± 15
minimum cost function value		
# of iteration ($N = 10$)	13.04 ± 0.83	12.88 ± 0.81
# of iteration ($N = 200$)	11.53 ± 0.56	10.92 ± 0.40

In Fig. 2b,2c we show the registration results of 5 different runs of the algorithm, with 200 particles sampled during each run. As it can be observed, the quality of the RS result, shown in Fig. 2b, heavily depends on the particular run of the method in contrast to PCA-SO (results are shown in Fig. 2c), that consistently converges to the same solution.



(a) (b)

Fig. 3: (a) standard SA experiment (b) SA experiment enhanced by PCA-SO; as can be seen, much less samples are needed to achieve desired function value

5.2 PCA-enhanced Simulated Annealing

To show the flexibility of PCA enhancement we integrate it in another SO technique - Simulated Annealing (SA). Here, we optimize the famous non convex *six-hump camelback* function $f(\boldsymbol{x})$; for analyzing the proposed algorithm and used the open source implementation of SA available at [17]. Our method is applied as following: for each new simulated particle, PCA tangential hyperplane is build based on (already evaluated) particles that are in the neighborhood of the new particle; particle is shifted in the direction of the cost function improvement; then the particle is processed as in standard SA method. The results for 50 independent optimization runs of SA and SA+PCA-SO with random initialization are shown in Table 2. It can be observed that the number of iterations required by SA to reach a good approximation of the global optimum is several orders of magnitude larger compared to PCA-SO. Notably, to reach a cost function value lower than -0.4 the average number of iterations needed by PCA-SO is only 22 ± 15 which is a remarkable improvement compared to the iterations needed by SA alone: 115 ± 54. Fig 3 illustrates results of one optimization run of each method and shows significant difference in the number of particle needed to reach the value, close to minimum. We attribute such good performance to the fact that PCA-SO shifts the particles in the direction of the minimum, allowing them to travel longer distances and therefore reduce random walk behavior.

5.3 Tracking with particle filter

In the last experiment, we also used our approach to improve object tracking accuracy with a Particle Filter (PF). Here, we used an open source tracking algorithm [15]. The PF aims to approximate a distribution and can be used to obtain minimum cost function value by taking the particle with the highest weight. For this purpose,a fixed number of particles N is sampled from a specified distribution and evaluated. A color histogram sampled in a region, which is bounded by an ellipse with parameters $\boldsymbol{x} = (x, y, a, b, \theta)^T$, represent the object

Table 2: Comparison between standard simulated annealing and its enhancement; minimum function value is -1.03

termination criteria	without PCA-SO	with PCA-SO
number of iterations		
value reached ($f \leq -0.4$)	115 ± 54	22 ± 15
value reached ($f \leq -1$)	492 ± 2291	55 ± 28
minimum cost function value		
# of iteration ($N = 200$)	-0.66 ± 0.43	-1.00 ± 0.03
# of iteration ($N = 1000$)	-0.96 ± 0.13	-1.00 ± 0.02

appearance. Here (x, y) is the center of the ellipse, (a, b) are the lengths of principal axis and θ is the rotation angle. For every particle, the color histogram $\mathcal{Q}^i$ is then compared to a target color histogram $\mathcal{P}$ of the helicopter obtained in the first frame of the sequence. The Bhattacharyya coefficient $\rho(\mathcal{P}, \mathcal{Q})$ [15] is used to measure similarity between the two histograms. As it is usual for tracking, a first order Markov Chain with linear dynamics is assumed. The full state vector is given by $\boldsymbol{s} = [x, y, \dot{x}, \dot{y}, a, b, \theta]^T$, where $(\dot{x}, \dot{y})$ is the velocity of the center of the ellipse. The Bhattacharyya coefficients, i.e. the cost function values, are mapped to probabilities w_t^i using the exponential function:

$$w_t^i = \frac{\hat{w}_t^i}{\sum_{i=1}^N \hat{w}_t^i}, \qquad \hat{w}_t^i = \exp\left(-\frac{1 - \rho(\mathcal{Q}_t^i, \mathcal{P})}{2\sigma^2}\right) \tag{7}$$

After weighting, the particles are resampled (selection) and propagated over time with the linear dynamical model: $\boldsymbol{S}_{t+1} = \boldsymbol{A}\boldsymbol{S}_t + \boldsymbol{\varepsilon}$ (mutation) where $\boldsymbol{\varepsilon}$ is noise, A is a constant motion matrix defined in [15], $\boldsymbol{S}_t = [\boldsymbol{s}_t^1 \dots \boldsymbol{s}_t^N] \in \mathbb{R}^{d \times N}$. We apply our enhancemnent directly after the dynamic step. In PF, several particles are created at the same time; we optimize only those particles, that have high weights, since optimization of all the particles will lead to the degradation of weight distribution due to normalization; for these particles, we, as usual, build a hyper-plane based on the particles generated in the current time step; note, that the cost function $f(\boldsymbol{x}) = \rho(\mathcal{P}, \mathcal{Q}(\boldsymbol{x}))$ depends only on the ellipse parameters $\boldsymbol{x}$ and therefore PCA space is build for the part of the state vector $\boldsymbol{s}$; improved particles are then resampled according to their weights.

To compare the two methods, we use the standard Euclidean distance between the ground truth object coordinates and the results produced by the given tracking algorithm as an error measure. Fig. 4 shows once more, that PCA-SO reduces the mean tracking error as well as the variance. PCA-SO can track the object after a heavy occlusion occurring between frames 270 and 310 in contrast to standard PF that looses track of the object; this can be observed both quantitatively, see Fig. 4, and qualitatively, see Fig. 5. It should be noted that a sufficient number of samples, e.g. more than $70 - 80$, are needed in order for PCA-SO to work well, because the PCA-based approximation is more reliable with denser neighborhood samples.

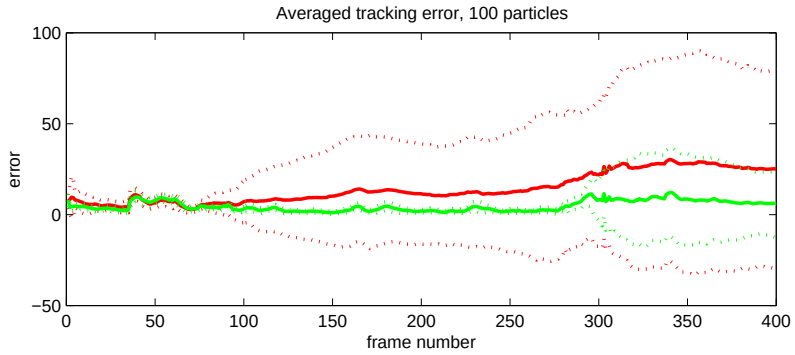
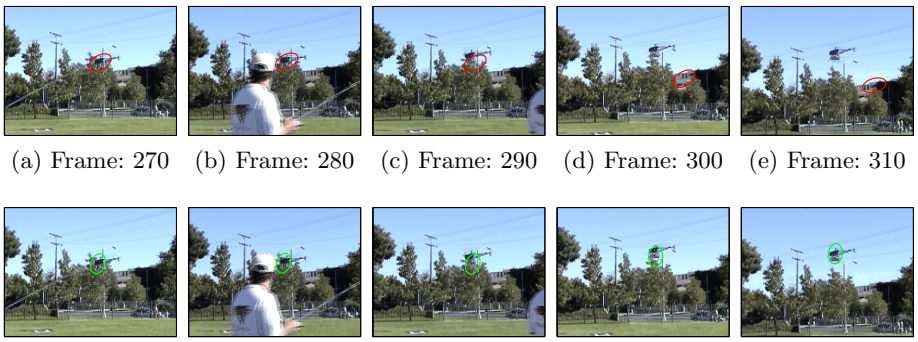


Fig. 4: Comparison of the tracking error for the standard Particle Filter (blue) and the improved PCA-SO Particle Filter (red); solid lines show mean tracking error over 30 independent runs; dotted lines show tracking error variance.



(a) Frame: 270 (b) Frame: 280 (c) Frame: 290 (d) Frame: 300 (e) Frame: 310

(f) Frame: 270 (g) Frame: 280 (h) Frame: 290 (i) Frame: 300 (j) Frame: 310

Fig. 5: Tracking results obtained with PF (top row) with proposed PCA-SO (bottom row). The PF method loses track of the object due to occlusions (the ellipse becomes red), while the proposed method is able to correctly follow the object after the occlusion (the ellipse stays green).

6 Conclusions

In this paper, we presented PCA-SO, a method to improve global stochastic optimization algorithms by improving the samples. The direction is given by the PCA component with smallest eigenvalue computed in a local neighborhood around the sample. We have shown that this yields the gradient of the cost function for small neighborhoods; on the other hand, for larger neighborhoods the direction reflects more global properties of the cost function landscape. Therefore, methods that combine stochastic search with local methods are a special case of our algorithm. The main advantages of PCA-SO methodology are its effectiveness and easy integration into stochastic optimization methods. In sev-

eral experiments, we have shown improvement in both accuracy and convergence rates using Random Sampling, Simulated Annealing and Particle Filter.

References

1. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.
2. M. Bray, E. Koller-Meier, and L. Van Gool. Smart particle filtering for 3d hand tracking. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 675–680. IEEE, 2004.
3. T. Cham and J.M. Rehg. A multiple hypothesis approach to figure tracking. In *CVPR*, 98.
4. Kiam Choo and D. J. Fleet. People tracking using hybrid monte carlo filtering. In *International Conference on Computer Vision*, volume 2, pages 321–328, 2001.
5. J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *IJCV*, 61(2):185–205, 2005.
6. J. Gall, J. Potthoff, C. Schnorr, B. Rosenhahn, and H. Seidel. Interacting and annealing particle filters: Mathematicsanda recipe for applications. *Journal of Mathematical Imaging and Vision*, 28:1–18, 2007.
7. N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 312–317. IEEE, 1996.
8. C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.
9. Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.
10. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948 vol.4, nov/dec 1995.
11. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
12. Alexander Kreinin, Leonid Merkoulovitch, Dan Rosen, and Michael Zerbs. Principal component analysis in quasi monte carlo simulation. *Algo Research Quarterly*, 1(2):21–30, 1998.
13. O. Martin, S.W. Otto, and E.W. Felten. *Large-step Markov chains for the traveling salesman problem.* 1991.
14. R.M. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.
15. Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110, January 2003.
16. C. Sminchisescu and B. Triggs. Covariance scaled sampling for monocular 3D body tracking. In *CVPR*, volume 1, 2001.
17. Joachim Vandekerckhove. General simulated annealing algorithm. `http://www.mathworks.de/matlabcentral/fileexchange/10548`, 2006.