Discrete attribute-based particle swarm optimization for robust parameter estimation

Alexey S. Chernyavskiy
State Research Institute of Aviation Systems (FGUP GosNIIAS), Moscow, Russia
achern@gosniias.ru

Abstract

In this paper a new robust estimator based on the discrete-attribute Particle Swarm Optimization (PSO) is presented. The proposed algorithm, called SwarmSAC, improves the standard RANSAC algorithm by making the search for the solution more guided and less random. The tentative solutions are iteratively updated using proportional likelihoods taking into account the best solutions obtained so far. The performance of the new method is evaluated in the context of image matching based on epipolar geometry estimation. Results demonstrate that for a fixed number of iterations, the SwarmSAC algorithm finds more inliers than the standard RANSAC method. The SwarmSAC method is generic and can be applied to a number of robust model estimation tasks.

Keywords: *RANSAC, robust model estimation, particle swarm optimization, image registration.*

1. INTRODUCTION

Robust model estimation is an important topic in computer vision. In image registration, a model such as a transformation matrix is fit to the data, usually consisting of 2D points coordinates. In order to get an accurate model, the point coordinates need to be known with high precision. Point coordinates are often subject to noise, either because of errors in point positioning by a human operator or, in case of automatic point matching, due to ambiguous feature descriptors leading to mismatches.

In case of small errors (noise) in point localization, the problem of model estimation can be treated by least-squares methods. When wrong point correspondences (mismatches) are present, least squares methods are not efficient since the assumption that measurement errors are generated by the underlying model is violated.

The data points which are generated by a hypothetical model are called *inliers* with respect to that model; other points are called *outliers*. Several strategies for classifying points into inliers and outliers have been proposed in the past. They include filtering of mismatches [1], [5], as well as robust estimators such as the widely used RANSAC [6].

RANSAC consists of randomly sampling the data and deriving hypotheses about the underlying model. Due to its randomness, many iterations are needed to explore a representative subset of the noisy data and find a reliable model which will be supported by most data points. In this paper we propose a technique of guided sampling based on the Particle Swarm Optimization (PSO) [8], [3], which guides the search towards samples which are more likely to produce more inliers.

The structure of this work is the following. In Section 2 several methods of mismatch removal are described, basic information about robust model estimation is summarized. Also, we discuss the motivation of this paper. Section 3 presents the standard continuous and binary particle swarm optimization (PSO) algorithm. The new method of robust model estimation SwarmSAC based on PSO is fully described in Section 4. Experimental results and the comparison of SwarmSAC with RANSAC are given in Section 5.

2. BACKGROUND

Mismatches are frequent in automatic image registration. The reason is that the features (corner points, line junctions, textured image patches) are described by vectors of parameters (descriptors) which tend to be ambiguous and far less discriminative than the human eye. As a result, some points in one image may be erroneously matched to points in another image located in a totally different place. Therefore, the problem of automatic mismatch removal is of great importance.

It is also important that after finding out which points are a true match, the number of inliers remains high. For camera pose estimation a large number of inliers assures a more accurate estimate; in object recognition, a larger number on inliers provides more support to the object classifier. One way of dealing with outliers is to filter out all possible mismatches, and then apply a least-squares method to derive a model that best fits the remaining points. By denoting as $\gamma \in [0,1]$ the share of inliers in the input data, a model estimation method is called robust if it can estimate the model from data where $\gamma < 1$. It is assumed that mismatches possess some property which sets them apart from the rest of the points. Locating and removing all such points will hopefully lead to $\gamma \approx 1$ in the remaining dataset.

The topological filter proposed in [5] tests all sidedness relations among matched image features. If feature c_3 in the first image is located in the left/right half-plane with respect to the directed line from c_1 to c_2 in this image, then its matched counterpart c'_3 should stay on the same side with respect to the directed line from c'_1 to c'_2 in the second image. Otherwise, the sidedness constraint is violated and a mismatch is possible. A violation score is computed, the most violating match is removed and the procedure is repeated again. Authors claim that the method tolerates up to 65% outliers ($\gamma = 0.35$). This figure was produced for the case when outliers were *uniformly* distributed across the image, and, based on our experience, the tolerance gets lower when there are regions (such as trees or grass) on an image that generate more mismatches than other regions. Also, the method has high complexity of $O(N^2 \log_2 N)$ [4], where N is the total number of data points, and it does not tolerate parallax effects.

Another method of mismatch detection is distance filtering used in [1]. The idea of the method is that the spatial distribution of features in two images should be similar; the distribution itself can be described as a set of distances from each feature to all the others. For each feature, a histogram of distance differences involving this feature and its matched counterpart is computed. Mismatches are discarded after comparing the histogram entries to a threshold. This method works well for aerial imagery because of small viewpoint changes but tends to lose performance in wide-baseline stereo situations.

The random sampling consensus method (RANSAC, [6]), proposed more than 25 years ago, follows a different strategy. It takes *samples of minimal size* to minimize the probability that there are outliers among the sampled data. A vector of model parameters θ is estimated from the data sample, and the score of this model $f(\theta)$ is computed by using *all* data points. Originally, the score was equal to the number of inliers, with a goal to maximize it. Other types of cost functions may be used:

$$f(\theta)=\sum_{i=1}^{N}\rho(r_i(\theta)^2),\ i=1,...,N\ ,\quad \rho(r_i^2)=\begin{cases}0, r_i^2\le T^2\\ 1, r_i^2>T^2\end{cases}\quad \text{(RANSAC)}$$

or $\rho(r_i^2)=\begin{cases}r_i^2, r_i^2\le T^2\\ T^2, r_i^2>T^2\end{cases}$ (MSAC [14]),

where T^2 is the outlier rejection threshold, $r_i(\theta)^2$ is discrepancy of data point x_i. The minimization of $f(\theta)$ is performed by trial-and-error, without computing its gradient.

Sampling in the context of robust model estimation is reduced to deciding which points will be more likely to generate a model which will produce the most inliers. Sampling in RANSAC is purely random and therefore many trials are to be performed until an optimal solution is found. It is possible to reduce the computational cost by means of guided sampling. Guided sampling should be regarded as a robust technique, while random sampling should not [10]. In guided sampling the search is depending on information about reliability of the data points. It is either provided by the user, or by a preliminary matching routine (as in PROSAC, [2]) or derived automatically during the sampling process.

In [12], a systematic trial strategy GASAC was proposed. For each sample, a fitness function was computed, and new samples were generated from the current ones based on their fitness using a genetic algorithm. The GA-based method achieved significant acceleration over RANSAC without using any prior information.

In this work we propose another technique, also inspired by natural processes, based on particle swarm optimization. We demonstrate that the new method allows guiding the search towards areas in the *M*-dimensional space such that the quality of found models is higher, and the convergence to the best solutions is significantly faster than RANSAC.

3. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population-based evolutionary computation technique, inspired by the social behavior of birds [8]. A simple set of rules such as *evaluate*, *observe*, *imitate*, guide a flock of birds, without apparent leader, towards a target (usually food, or resting place). These rules, expressed in mathematical form, may be applied to numerical optimization.

When looking for a potential solution to an optimization problem, each *M*-dimensional solution vector, called a *particle*, $\mathbf{X}_i=(x_{i,1},x_{i,2},...,x_{i,M})$ is updated by using information about its current performance, its best previous performance and that of the whole set of particles $\{\mathbf{X}_i\}_{i=1}^{P}$, called a *swarm*. A *fitness* function $f(\mathbf{X}_i)$ measures how well the solution $\mathbf{X}_i$ solves the current problem, and for each particle $\mathbf{X}_i$ the best fitness $f(\mathbf{B}_i)$ found in position $\mathbf{B}_i$ is kept in an archive. Also, the global best position **G** and its fitness value *f*(**G**) is preserved. In many PSO realizations, a local best position within a specified neighborhood is used instead of the global best. Each particle has an associated vector of *velocity* $\mathbf{V}_i=(v_{i,1},v_{i,2},...,v_{i,M})$. Both $\mathbf{X}_i$ and $\mathbf{V}_i$ are initially generated randomly and updated according to the following rules:

$$v_{i,j}=v_{i,j}+\varphi_1(b_{i,j}-x_{i,j})+\varphi_2(g_j-x_{i,j})\ ,$$

$$x_{i,j}=x_{i,j}+v_{i,j}\ \text{(continuous PSO)}$$

$$x_{i,j}=\begin{cases}1,\ \text{if } rand<S(v_{i,j})\\ 0,\ \text{otherwise}\end{cases}\ \text{(binary PSO)}$$

where φ_1, φ_2 and *rand* are random numbers uniformly distributed in the interval (0,1); $b_{i,j}$ and g_j are the components of particles $\mathbf{B}_i$ and **G** correspondingly; $S(z)=[1+\exp(-z)]^{-1}$ is the sigmoid function. The parameters φ_1 and φ_2 regulate how likely the particle will be attracted to the global best solution and its previous best position.

PSO has been successfully used for a wide range of optimization problems, including image registration [15]. An advantage of PSO is that it allows optimization of very complex cost functions without the need to compute the gradients, which is sometimes hard or impossible. Several researchers have generalized PSO to handle variables belonging to discrete sets of integer or real numbers [11]. In [2] a variant of discrete PSO was introduced, in which the particles represented *unordered* non-repeating integer numbers: in other words particle {1, 5, 8} was treated as equivalent to {8, 1, 5}. The authors of [2] applied this algorithm to identify attributes relevant for classification in data mining. Their discrete attribute-based particle swarm algorithm was initially devised for the case of variable attribute length. In the current work, a fixed attribute length *M*=8 is used. The application of particle swarm optimization to robust model estimation is presented in the next section.

4. THE SWARMSAC METHOD FOR ROBUST MODEL ESTIMATION

4.1 The SwarmSAC algorithm

Suppose that the total number of data points is *N*, the length of a sample is *M* and fix a swarm size (number of particles) as *P*; *N*, *M*, *P* > 0. A set of *P* particles is randomly generated, where each element is a unique positive integer index varying from 1 to *N*. There are no repeating elements within a given particle. After the initial population is generated, the global best **G** and personal best values $\mathbf{B}_i$ are computed and stored, along with fitness values.

Next, the velocities are computed. The discrete-attribute DPSO algorithm of [2] deals with proportional likelihoods instead of velocities. Every particle $\mathbf{X}_i$ is associated with a 2x*N* array $\mathbf{V}_i$ of proportional likelihoods. Each of the *N* elements of the first row represent the proportional likelihood that a point will be selected. The second row of $\mathbf{V}_i$ shows the indices of points associated with the respective proportional likelihoods. At the beginning, all points are equally likely to be selected for the next iteration of the particle swarm algorithm: $\mathbf{V}_i = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 2 & \dots & N \end{pmatrix}$. Afterwards, the array $\mathbf{V}_i$ is altered based on whether each particular index from the second row of $\mathbf{V}_i$ is also part of the current personal best and global best solution. Three constant updating factors $\alpha, \beta, \gamma > 0$ chosen by the user are used to update the elements of $\mathbf{V}_i$. These parameters define the influence of $\mathbf{X}_i$, $\mathbf{B}_i$ and $\mathbf{G}$ to the adjustment of every element $v_{i,j}$. All indices present in $\mathbf{X}_i$ have their proportional likelihood increased by α; additionally all indices present in $\mathbf{B}_i$ and/or $\mathbf{G}$ are increased by β and γ. For example, if *N*=5 and *M*=3, and $\mathbf{X}_i = \{2,3,4\}$, $\mathbf{B}_i = \{3,5,2\}$, $\mathbf{G} = \{1,2,4\}$, then the updated $\mathbf{V}_i$ will be:

$$\mathbf{V}_i = \begin{pmatrix} 1+\gamma & 1+\alpha+\gamma & 1+\alpha+\beta & 1+\alpha & 1+\beta \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

In our experiments we used the values $\{\alpha, \beta, \gamma\} = \{0.3, 0.5, 0.9\}$. Next, the first row of $\mathbf{V}_i$ is multiplied by uniform random numbers between 0 and 1. All elements of the first row of $\mathbf{V}_i$ are ranked in decreasing order of value, and the indices of attributes follow their respective proportional likelihoods. The *M* indices from the second row corresponding to the largest proportional likelihoods are selected to compose a new particle $\mathbf{X}_i$. Indices that have a higher proportional likelihood are, on average, more likely to be selected. The above procedure is repeated for all the particles of a swarm, the personal and global best are updated if necessary. The algorithm continues until a pre-defined number of iterations is performed or some other termination criterion is met. The outline of SwarmSAC is the following:

```
for $i$:=1 to $P$ particles in the swarm do
  Create $\mathbf{X}_i$ as a vector of $M$ random non-repeating indices
  Generate model hypothesis from $\mathbf{X}_i$
  Evaluate fitness of model $f(\mathbf{X}_i)$
  $\mathbf{B}_i := \mathbf{X}_i$, $f(\mathbf{B}_i) := f(\mathbf{X}_i)$, update $\mathbf{G}$ and $f(\mathbf{G})$
end
for $t$:=1 to number of swarm iterations do
  for $i$:=1 to $P$ particles in the swarm do
    Generate 2x$N$ array $\mathbf{V}_i$
    Update $\mathbf{V}_i$ using $\mathbf{X}_i$, $\mathbf{B}_i$, $\mathbf{G}$, and $\alpha, \beta, \gamma$
    Multiply first row of $\mathbf{V}_i$ by uniform random numbers from
    (0,1) and sort $\mathbf{V}_i$ according to proportional likelihoods
    Replace $\mathbf{X}_i$ by indexes from $\mathbf{V}_i$ corresponding to $M$ largest
    likelihoods
    Generate model hypothesis from $\mathbf{X}_i$
    Evaluate fitness of model $f(\mathbf{X}_i)$
    Update $\mathbf{B}_i$, $\mathbf{G}$, $f(\mathbf{B}_i)$ and $f(\mathbf{G})$ if necessary
  end
end
Return model corresponding to particle G with best fitness
```

4.2 Computational load

Compared to the standard RANSAC, SwarmSAC requires slightly more memory and CPU time. Since the particle swarm needs to store the previously achieved results, memory must be allocated for: *P* particles each consisting of *M* indices (*PM* integers), *P* personal best positions (*PM*), *P* values of personal best fitness (*P* integers or real numbers – depending on the definition of fitness), global best particle (*M* integers), one global best fitness (real number). In total, 2*PM*+1 integers and *P*+1 real numbers will be stored. One can see that the memory requirements are negligible.

Apart from model estimation and residual computation present both in RANSAC and SwarmSAC, the CPU time and resources are spent in SwarmSAC during the velocity computation stage. It requires *PN* multiplications and *P* calls to a sorting procedure. Sorting is time consuming, but full sorting is not required. Instead we need to find the largest *M* elements, and therefore a selection method based on the HeapSort algorithm [9] is preferred. It has complexity of $O((N-M)\log_2 M)$. Taking into account that $M << N$, the overall cost of sorting one swarm becomes $O(PN \log_2 M)$.

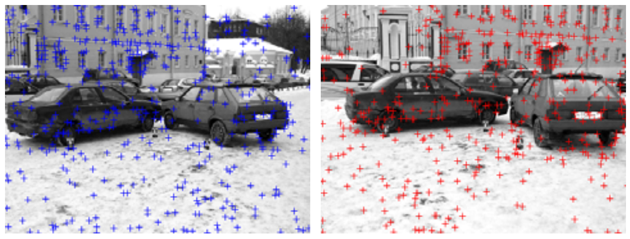

Figure 1: Tentative matches before outlier rejection

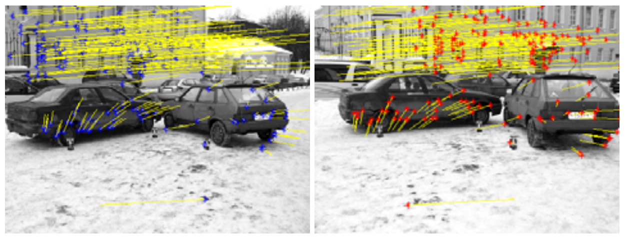

Figure 2: Inliers satisfying the epipolar constraint found by SwarmSAC

5. RESULTS

In order to demonstrate the effectiveness of SwarmSAC and compare its performance with RANSAC, we used a pair of images containing *N*=488 tentative point correspondences obtained from an automatic algorithm using local descriptors. We then applied SwarmSAC and labeled point as inliers/outliers according to the epipolar geometry, with displacement $r_i(\theta)^2$ equal to the sum of squared distances of the points from epipolar lines [16], and $T^2 = 5$. The fundamental matrix was estimated using the normalized eight-point (M=8) algorithm by Hartley [7]. Swarm size of P=20 particles was iterated for 50 iterations, which gave 1000 calls to model estimation function. Figure 1 shows the tentative matches, and Figure 2 demonstrates the 178 inliers (36% of all points) for the image pair; lines point to the place where the matched counterpart of each feature is located. The maximum number of inliers was found after 32 swarm iterations, 640 function evaluations.

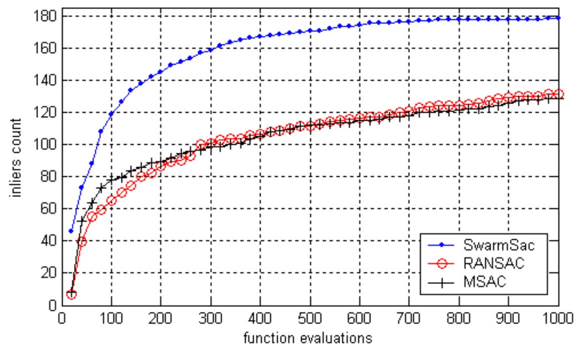

Figure 3: Improvement of the number of inliers averaged over 100 attempts

We then studied how fast SwarmSAC finds the inliers compared with RANSAC and MSAC. It is important, especially for real-time systems, to maximize the number of inliers found for a fixed number of iterations. We run both methods for 100 times and compute the mean best fitness. RANSAC and MSAC were iterated 1000 times. Figure 3 shows the number of calls to functions for fundamental matrix estimation and computation of residuals plotted along the x-axis, and the largest number of inliers found so far presented on the y-axis. For a fixed number of function evaluations, the new method finds 52% more inliers than the standard RANSAC.

6. CONCLUSION

The use of particle swarm optimization guides the sampling process towards areas which are more likely to produce more inliers. Clearly, the new SwarmSAC method achieves better results for a given number of function evaluations, proving that guided search outperforms purely random trials.

In the future we plan to compare the performance of standard RANSAC and our SwarmSAC method on images with various inlier share γ. It is hoped that for a decreasing γ, SwarmSAC will outperform RANSAC more and more. Another direction of research would be to tune parameters of the proportional likelihood update formula dynamically, based on estimated γ at each iteration of the swarm; this might also help to escape occasional local extrema of the cost function. Finally, we plan to study how the swarm optimization will perform on other popular cost functions such as the maximum-likelihood (MLESAC, [13]).

7. REFERENCES

[1] Y. Blokhinov and D. Gribov: A new approach to automatic junction of overlapping aerial imagery data, XXth ISPRS Congress, Istanbul, Turkey, July 2004.

[2] O. Chum and J. Matas: Matching with PROSAC – Progressive sample consensus, In Proc. CVPR 2005, pp. 220-226, June 2005.

[3] E.S. Correa, A.A. Freitas and C.G. Johnson: A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set, In M. Keijzer et al. (Eds.), Proc. Genetic and Evolutionary Computation Conference (GECCO-2006), pp. 35-42, ACM Press, July 2006.

[4] V. Ferrari: Affine invariant regions++, Diss., Technische Wissenschaften, Eidgenössische Technische Hochschule ETH Zürich, Nr. 15549, 2004.

[5] V. Ferrari, T. Tuytelaars and L. Van Gool: Wide-baseline multiple view correspondences, IEEE Computer Vision and Pattern Recognition (CVPR), Madison, USA, June 2003.

[6] M.A. Fischler and R.C. Bolles: Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography, In Communications of the ACM, vol. 24, pp. 381-395, 1981.

[7] R. Hartley: In defense of the eight-point algorithm, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, pp. 580-593, 1997.

[8] J. Kennedy and R.C. Eberhart: Swarm intelligence, Morgan Kaufmann Publishers, 2001.

[9] D. Knuth: The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997.

[10] P. Meer: Robust techniques for computer vision. Emerging Topics in Computer Vision, G. Medioni and S. B. Kang (Eds.), Prentice Hall, pp. 107-190, 2004.

[11] J. Pugh and A. Martinoli: Discrete multi-valued particle swarm optimization, Proceedings of IEEE Swarm Intelligence Symposium (2006), pp. 103-110, Indianapolis, USA, May 2006.

[12] V. Rodehorst and O. Hellwich: Genetic Algorithm SAmple Consensus (GASAC) - A Parallel Strategy for Robust Parameter Estimation, Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, pp. 103-110, 2006.

[13] P. Torr and A. Zisserman: MLESAC: A new robust estimator with application to estimating image geometry, Computer Vision and Image Understanding, vol. 78, pp. 138-156, 2000.

[14] P. Torr and A. Zisserman: Robust computation and parametrization of multiple view relations, In Proc. ICCV’98, pp. 727-732, 1998.

[15] M.P. Wachowiak, R. Smolikova, Yufeng Zheng, J.M. Zurada and A.S. Elmaghraby: An approach to multimodal biomedical image registration utilizing particle swarm optimization, IEEE Trans. on Evolutionary Computation, vol. 8, pp. 289-301, 2004.

[16] Z. Zhang: Determining the epipolar geometry and its uncertainty – A review, IJCA, vol. 27, pp. 161-195, 1998.

About the author

Alexey Chernyavskiy received his specialist degree in Applied Mathematics from Moscow State University, Department of Computational Mathematics and Cybernetics in 2000. In 2003 he completed his M.S. degree in Geophysics at the University of Utah, USA. He now works at the State Research Institute of Aviation Systems (GosNIIAS) in Moscow, Russia. His contact email is achern@gosniias.ru .