On solving Exterior Orientation of an Image with Particle Swarm Optimization

Petri Rönnholm[1], Matti Kurkela[1], Matti Vaaja[1], Hannu Hyyppä[1]

[1] Department of Built Environment, Aalto University, Finland

Keywords: Exterior orientation, particle swarm, stochastic model, robustified, comparison

Abstract

Solving the exterior orientation of images is a fundamental component in photogrammetric mapping and 3D restitution processes. Additionally, it is essential in photogrammetric tasks such as visual odometry, camera-based visual simultaneous localization and mapping, camera calibration, camera-based 3D tracking of movement, and change detection. The aim of this research was to evaluate whether particle swarm optimization is suitable for finding the exterior orientation parameters of a single image using image resection. In addition, we developed a robustified particle swarm optimization by adding an iteratively changing stochastic model to the optimization criteria by attaching a weight matrix with residual vectors. The method was compared to the solution from the least squares method using both simulated ideal and noisy data. Solving the exterior orientation parameters reliably with particle swarm optimization was possible after fine-tuning the algorithm's options. The non-robustified version of particle swarm optimization provided identical results to the non-robustified least squares method. However, in the case of the robustified particle swarm optimization, only 60% of attempts resulted in the same outcome as the corresponding robustified least squares method, with sub-millimeter accuracy. In 40% of cases, the results achieved millimeter accuracy. The sub-millimeter accuracy was achieved in every case with sequential robustified particle swarm optimization, where the algorithm was rerun using stricter bounds for unknown parameters if the evaluation criteria were too large. The implementation of particle swarm optimization is easier than that of the nonlinear least squares method. However, the computation time for particle swarm optimization was significantly longer.

1. Introduction and previous work

The exterior orientation of an image represents the position and rotation of the camera coordinate system attached to an image with respect to some external coordinate system. Depending on the context, the terms spatial orientation, camera pose, extrinsic orientation, and external orientation are also used. Solving the exterior orientations of images is a fundamental component of photogrammetric mapping and 3D restitution processes. In addition, it is essential in photogrammetric tasks such as visual odometry (Chakraborty and Verma, 2024), camera-based visual simultaneous localization and mapping (SLAM) (Li et al., 2025), camera calibration (Luhmann et al., 2016), camera-based 3D tracking of movement (Ristanto et al., 2022), and change detection (Quin et al., 2016). In addition, there are special applications where a known exterior orientation is required. One example is the estimation of human height from a single image (e.g., Lee et al., 2008; Liscio et al., 2021) or from a video stream (e.g., Ciampini et al., 2024). Another example is non-stereoscopical gait analysis of a walking human using three video cameras with known exterior orientations (Jensen and Juhl, 2009).

There are alternative methods for solving the parameters of exterior orientation. Direct georeferencing sensors, such as Global Navigation Satellite System (GNSS) combined with an inertial measurement unit (IMU), provide a good approximate exterior orientation in outdoor environments (Stöcker et al., 2017; Jouybari et al., 2024). A more accurate solution is achieved through image resection if known ground control points are available and corresponding image points are measured. In photogrammetric tasks, an image block typically contains many overlapping images, and the exterior orientations of all images are solved simultaneously using a bundle block adjustment. This adjustment is a nonlinear least squares method when the collinearity equations are utilized as the mathematical model (Triggs et al., 2000). Nonlinear methods require initial values for unknown parameters, which are typically obtained using direct methods methods (Grussenmeyer and Al Khalil, 2002).

This article focuses on particle swarm optimization, which was introduced by Kennedy and Eberhart (1995), as it has recently appeared in the literature as a partial replacement for the least squares method. The algorithm is inspired by the behavior of fish and bird swarms. There are several variants of particle swarm optimization, and recent major advances are discussed by Zhu et al. (2025), for example.

One of the earliest attempts to solve image orientations using particle swarm optimization was published by Wachowiak et al. (2004). In this research, however, particle swarm optimization was utilized for image registration, i.e., for relative orientation.

Using epipolar geometry is a popular method for finding relative orientation. For example, Elashry and Toth (2023) solved an essential matrix (relative orientation) using particle swarm optimization and compared the results with several other optimization alternatives, namely genetic algorithm, Salp Swarm Algorithm, Gray Wolf Optimization, and improved Gray Wolf Optimization. Guan et al. (2021) estimated camera attitudes by first solving the trifocal tensor with particle swarm optimization.

Li and Li (2012) applied particle swarm optimization for finding approximate values for exterior orientation parameters of six images. In their case, the 3D coordinates of tie points were also unknown. Mukhopadhyay and Hati (2017) solved both the exterior orientation and camera calibration by utilizing particle swarm optimization. However, their mathematical model was the direct linear transformation (Abdel-Aziz and Karara, 1971), which is slightly less accurate than using collinearity or coplanarity equations (El-Ashmawy, 2015). Another camera calibration study, which includes solving the exterior orientation of images using particle swarm optimization, is described in Li et al. (2016). Wang and Zhang (2026) utilized particle swarm

optimization to solve the targetless system calibration of a mapping system that includes both a camera and LiDAR.

In Low et al. (2010), particle swarm optimization was utilized to solve the landmark-based localization part of a SLAM system using a stereo camera. Siddiqui and Khatibi (2014) applied particle swarm optimization for solving visual odometry with a single camera with the aid of tracking planar templates. John et al. (2019) applied particle swarm optimization for system calibration of a stereo camera-based mobile mapping system. In addition, they utilized a multiswarm particle swarm algorithm for visual odometry.

As this literature review illustrates, particle swarm optimization has mostly been utilized to address more complex cases than simply solving the exterior orientation of a single image. An exception is Li and Zhong (2019), in which image resection was solved for individual images. However, they used the results only as approximate values for the nonlinear least squares method (specifically, the Gauss-Newton method). Since exterior orientation is a key component in many complex cases, it is worthwhile to examine it more closely.

The aim of this research was to evaluate whether particle swarm optimization is suitable for finding the exterior orientation parameters of a single image using image resection. The method was compared to the least squares method, which is typically used in photogrammetry. In addition, we examined the possibility of adding an iteratively changing stochastic model to the optimization criteria of the particle swarm optimization by integrating a weight matrix with residual vectors. In this research, we utilized simulated data that included both ideal and noisy cases.

2. Particle swarm optimization algorithm

Particle swarm optimization begins with initialization, where each particle of the swarm is assigned a random position and velocity in the parameter space. In addition, individual and global best positions are initialized. The individual best position is stored for each particle, representing the best position it has visited so far. Correspondingly, the global best position represents the best solution that any particle has found so far.

After initialization, the iterative process continues until the objective value falls below a threshold, or the maximum number of iterations is reached. According to Shi and Eberhart (1998), each particle has a velocity vector (v_{vel}) and position (x_p), which are updated by the following equations:

$$v_{vel}(i+1) = W v_{vel}(i) + h_1 u_1 \left(p(i) - x_p(i)\right) + h_2 u_2 \left(g(i) - x_p(i)\right) \quad (1)$$

$$x_p(i+1) = x_p(i) + v_{vel}(i+1) \quad (2)$$

where i is the iteration index, W is the inertia weight, h_1 is the cognitive coefficient, h_2 is the social coefficient, u_1 and u_2 are random numbers within [0,1], p is the personal-best position found so far, and g is the global-best position currently known across the swarm. Eq. 1 and Eq. 2 are illustrated in Figure 1. In principle, a weighted sum of the current velocity and the direction vectors toward the personal-best (cognitive part) and global-best (social part) positions produces a new velocity vector, which then updates the particle’s position.

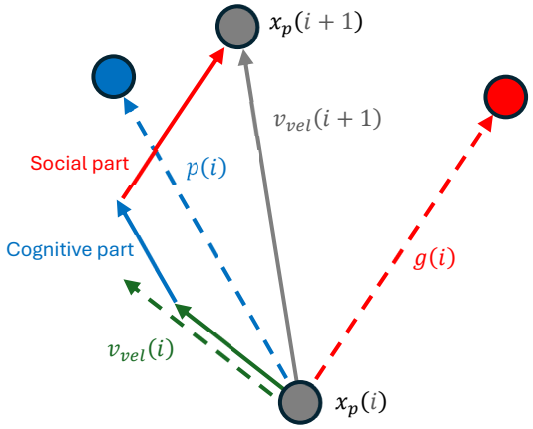

Figure 1. A 2D illustration of updating the velocities and positions of particles in particle swarm optimization.

3. Materials and methods

3.1 Creation of simulation data

Two simulated datasets were created. The first dataset includes ideal image observations and the corresponding 3D points. The second dataset uses the same 3D points, but noise was added to some of the image observations. However, half of the image observations were left unchanged to retain a connection to the known ideal exterior orientation parameters.

The camera parameters were selected to simulate a DJ3 Air medium-tele camera with a camera constant of 70 mm, an image size of 8064×6048 pixels, and a pixel size of 2.4 micrometers. For the first dataset, the image observations were chosen to form a regular grid over the image at 1000-pixel intervals (48 points in total). The exterior orientation parameters were set in a local coordinate system as X_o = 4000 m, Y_o = 5000 m, Z_o = 100 m, omega = 10 degrees, phi = 15 degrees, kappa = 20 degrees.

Using this information and the collinearity equations (Eq. 3), the corresponding 3D object points were generated for each image observation. The collinearity equations can be written as:

$$\begin{cases} x = -c\dfrac{a_{11}(X-X_o) + a_{12}(Y-Y_o) + a_{13}(Z-Z_o)}{a_{31}(X-X_o) + a_{32}(Y-Y_o) + a_{33}(Z-Z_o)} \\ y = -c\dfrac{a_{21}(X-X_o) + a_{22}(Y-Y_o) + a_{23}(Z-Z_o)}{a_{31}(X-X_o) + a_{32}(Y-Y_o) + a_{33}(Z-Z_o)} \end{cases} \quad (3)$$

Parameter c is the camera constant (X_o, Y_o, Z_o) is the location of the projection center expressed in the ground coordinate system, elements a_{11} ... a_{33} are from the 3D rotation matrix, (X, Y, Z) is a 3D object point, and (x, y) is the corresponding image point. In this form, Eq. 3 assumes that the origin of the image coordinate system is located at the principal point.

The desired Z coordinate was randomly generated for each point within the range [20 m, 40 m]. The X and Y coordinates were estimated using particle swarm optimization in MATLAB (Figure 2). The implementation, utilizing the *particleswarm* function with default parameters, achieved a 100 % success rate. The results were confirmed by projecting the created 3D points onto the image plane using the collinearity equations.

Next, noise was added to the image observations. Half of the image points were randomly selected. The x and y coordinates of these selected points were independently distorted using a uniform distribution over the range [-2, 2] pixels. This is a typical

range when natural feature points are measured from images. In Figure 3, the magnitude and directions of the created errors are illustrated. The figure also reveals the distribution of the modified image points.

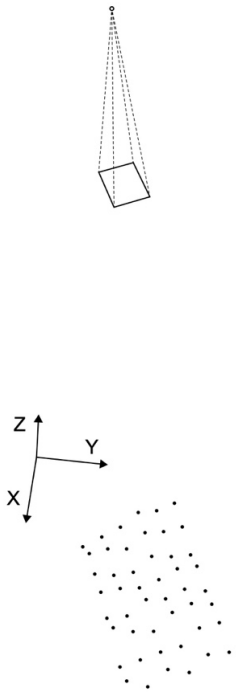

Figure 2. An illustration of the camera and the 48 generated 3D points. Note that the coordinate system demonstrates only the directions of the coordinate axes, while the origin is chosen arbitrarily.

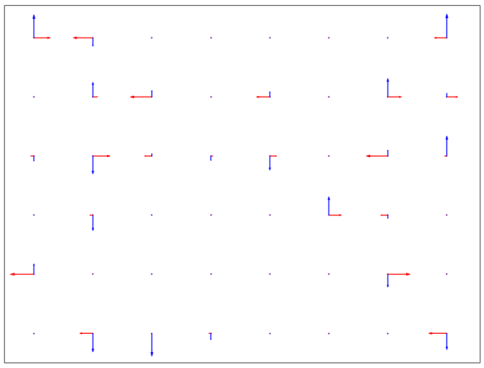

Figure 3. The locations of the distorted image observations were randomly selected, covering half of the points. The distortions were scaled by a factor of 200 to improve visibility.

3.2 Solving exterior orientation parameters with the least squares method

Since the results of particle swarm optimization are compared with the least squares method with observation equations (3D points are assumed to be constants), the latter is presented first. In addition, our implementation of particle swarm optimization uses the same objective function, i.e., minimizing the sum of squared errors. In cases where noise or outliers are expected, the objective function can be changed to minimize the weighted sum of squared errors. The nonlinear least squares method is a typical choice in photogrammetry for solving problems where no direct solution exists due to the nonlinear nature of the collinearity equations and observational noise.

The collinearity equations (Eq. 3) serve as the mathematical model. Each image observation yields two equations in a system of equations that constitutes a nonlinear functional model. A Taylor series expansion is applied to linearize the functional model, leading to the following matrix-vector representation:

$$y = Ax \tag{4}$$

in which y includes residuals computed as the observed image coordinates minus those computed from the collinearity equations using the current initial parameter values, A is the Jacobian matrix including the partial derivatives of the collinearity equations with respect to the unknown parameters, and x is the parameter correction vector.

The optimization objective is to minimize the sum of squared errors ($S(x)$).

$$\min_x\{S(x)\} = \min_x\{(Ax - y)^T(Ax - y)\} \tag{5}$$

This leads to the following solution:

$$x = (A^T A)^{-1}A^T y \tag{6}$$

The solution is iterative. Each unknown parameter requires an initial value, which is stored in the exterior orientation parameter vector E. At iteration k, E is updated using the solution from Eq. 6.

$$E_{k+1} = E_k + x \tag{7}$$

Iterations can be terminated when the corrections (x) are sufficiently small.

In the least squares method, the initial values of unknown parameters must be sufficiently close to the true values to ensure convergence to the correct solution. To find good initial values, a linear method is preferred. In our case, the pyramid method (Kraus, 1997, pp. 48 - 55) was applied to obtain the initial values.

If the data includes noise, a stochastic model can be added to the least squares method via the weight matrix P. In that case, the new minimization criteria is

$$\min_x\{(Ax - y)^T P(Ax - y)\}, \tag{8}$$

which leads to the least squares solution

$$x = (A^T PA)^{-1}A^T Py. \tag{9}$$

In this research, iteratively reweighted least squares, i.e., robustified least squares, was applied to noisy data. In our case, the weights in the matrix P, which initially start as an identity matrix, were updated after each iteration according to the following function (Figure 4):

$$f(v) = \begin{cases} 1 & if \ \ |v| \le b \\ e^{-a(|v|-b)} & if \ \ |v| > b \end{cases} \tag{10}$$

The parameter b is a threshold and v is the image residual, i.e., the difference between observed and estimated image points. Parameter a affects the strength of suppression. In practice, this function keeps the weights of points with small residuals intact and lowers the weights of points with large residuals.

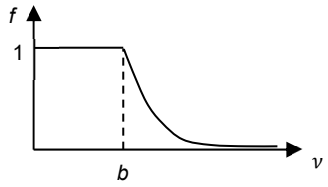

Figure 4. An illustration of the weight update function.

The parameter b was set to 0.05 pixels. After examining the effect of varying parameter a, we set a to 10 because the number of required iterations were the lowest (Figure 5) and no significant difference in 3D location error was observed (Figure 6).

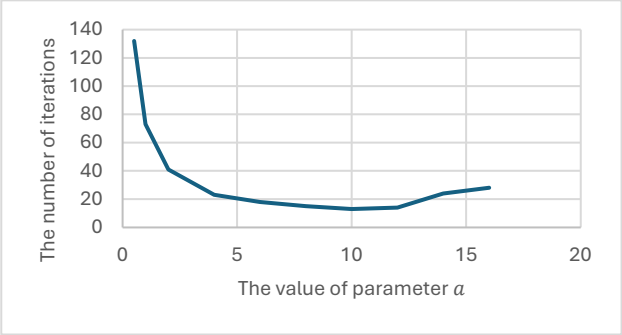

Figure 5. The effect of parameter a on the number of iterations required for the robustified least squares solution using simulated noisy data.

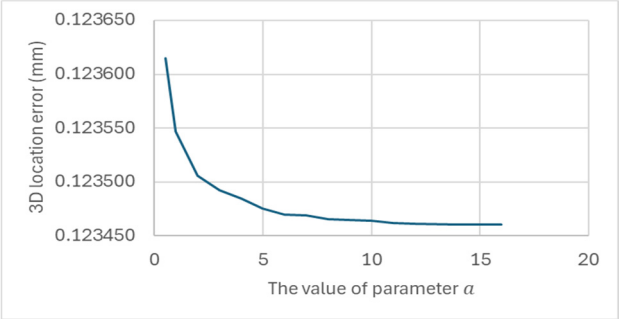

Figure 6. The effect of parameter a on the 3D location error is practically insignificant within the presented range, although a small decreasing trend is detectable. However, when a exceeds 16.4, the error suddenly rises above 3.6 mm. This is not visualized, as it would obscure the small changes.

3.3 Solving exterior orientation parameters with particle swarm optimization

Particle swarm optimization for solving exterior orientation parameters was implemented in MATLAB R2025a. MATLAB provides the function *particleswarm*, which enables solving problems by means of particle swarm optimization. As input, it requires metrics about the error that the suggested parameters cause. In this case, the known 3D points were projected onto the image plane with the collinearity equations (Eq. 3) using the current exterior orientation parameters (approximate until the solution is found), giving estimates of image points. Fundamentally, we selected the very same optimization criteria as with the least squares method, i.e., the sum of squared error (S) computed from a residual vector (v), leading to the following metrics

$$S = v^T v \quad (11)$$

The *particleswarm* solver with default parameters led to a 100% failure rate. In this case, a failure means that the solution is not even close to the correct one, but the algorithm has stuck in a local minimum. To solve this problem, the options of the *particleswarm* function were tuned. First, parameters were changed manually to find the first successful combination. After that, the parameters were examined one at a time by systematically changing their values and running the algorithm 25 times for each case.

Figure 7 reveals that by increasing the lower bound of the adaptive inertia, the number of failures reaches zero at 0.7, which was therefore selected. Examination of the upper bound of the adaptive inertia (Figure 8) led to the selection of value 1.1. The value of the self-adjustment weight was set to 0.15, which was the lowest value that resulted in no failures (Figure 9). The examination of the social adjustment weight (Figure 10) did not reveal a clear trend. From the parameter values that did not cause failures, the value 0.2 was selected because the mean error was slightly smaller than with other values. However, in practice, the difference in such error was insignificant.

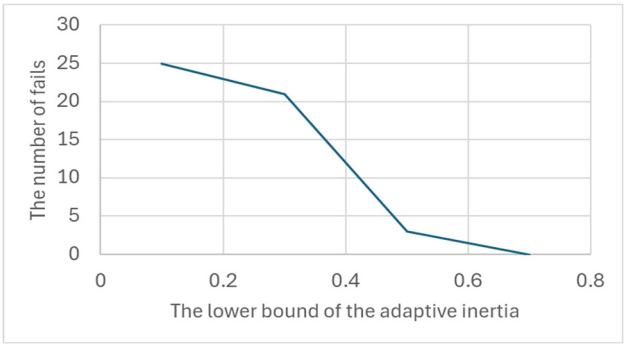

Figure 7. The effect of the lower bound of the adaptive inertia (exploration) on the number of failures.

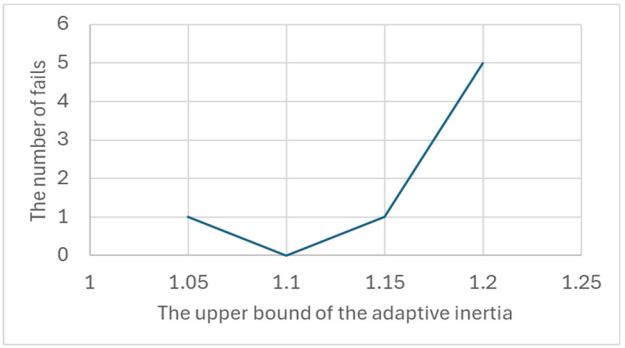

Figure 8. The effect of the upper bound of the adaptive inertia (exploitation) on the number of failures.

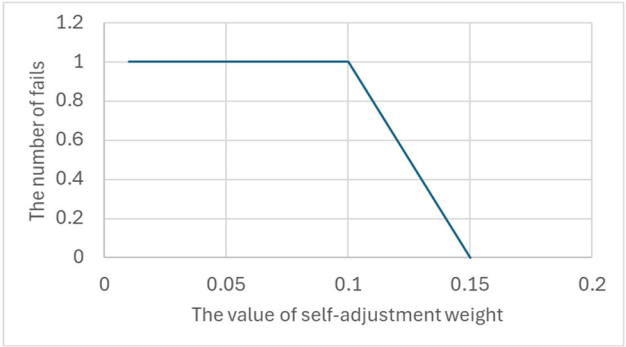

Figure 9. The effect of self-adjustment weight (personal best) on the number of failures.

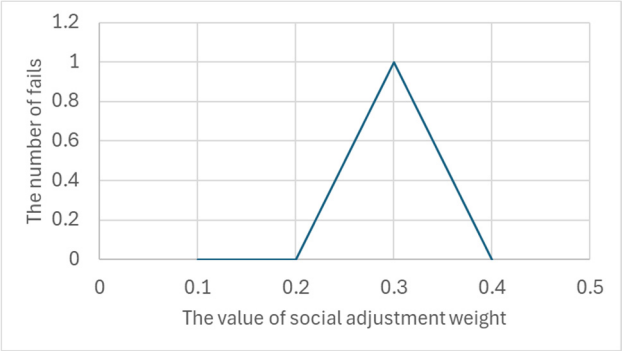

Figure 10. The effect of social adjustment weight (global best) on the number of failures.

For the noisy data, we further developed the implementation to meet the optimizing conditions of the robustified least squares. In this case, the error metrics were adjusted to use a stochastic model, specifically incorporating a weight matrix (P):

$$S_{weighted} = v^T P v \quad (12)$$

We refer to this version as robustified particle swarm optimization. To adapt this condition to the particle swarm algorithm, some additional steps were necessary. The particle swarm method does not guarantee that the suggested exterior orientation parameters will always decrease the sum of squared error, but the computation can temporarily lead to a worse solution. We updated the values of the weight matrix P using the same function (Eq. 10) as in the case of robustified least squares. A threshold for starting the update of P was applied to ensure that updating did not start too early, which could lead to incorrect weights. In our case, we set the threshold for the sum of squared errors to 70. If the data includes noise, this threshold cannot be too strict. Additionally, the update of the P matrix was permitted only if the current sum of squared errors was smaller than any previous sum of squared errors. Because convergence to the solution is much slower with particle swarm optimization than with the least squares method, the value for parameter a needs to be much smaller (Figure 11). In our case, the value 0.01 was selected to ensure a robust solution, even though the computation time significantly increased (Figure 12). Lower and upper limits for planimetric locations were set to [0, 6000], and for elevation to [0, 110]. These parameters were not specifically tested but were intended to simulate a scenario where the location is not well known. Lower and upper limits for angles were set to [0, 65] degrees. If the upper limit of the angles was increased beyond this, the solution failed. The number of particles in the swarm was set to 2000. A smaller swarm size resulted in failures.

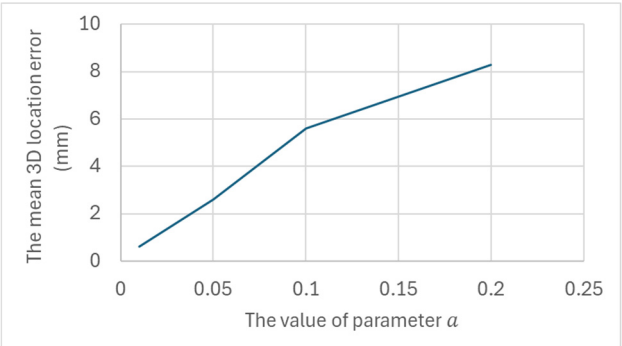

Figure 11. The effect of parameter a on the mean 3D location error.

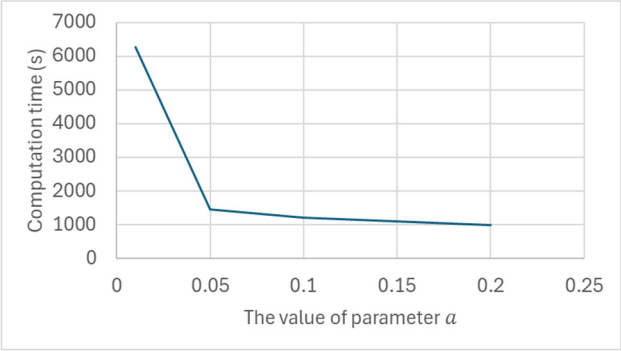

Figure 12. The effect of parameter a on computation time (25 runs).

Furthermore, we examined sequential robustified particle swarm optimization. In this approach, the sum of squared error from the was thresholded. If its value exceeded the selected threshold of 0.1, the algorithm was rerun with a new swarm size, as well as updated lower and upper bounds. The swarm size was reduced to 200 due to the smaller search range. The lower bounds were derived from the previous solution in such way that, for angles, the lower bound was the angle minus 0.0001 and for coordinates, the lower bound was minus 0.01. The upper bounds for angles were the angle plus 0.0001, and for coordinates, they were plus 0.01.

4. Results

Both methods found the correct exterior orientation parameters with an accuracy of six decimals when ideal data was used. Therefore, these results are not presented. Instead, the results obtained using noisy data are listed in Table 1. Unlike the least squares method, particle swarm optimization yields slightly different solutions each time. Therefore, the best and worst cases out of 25 attempts are highlighted if there were any differences. While robustified particle swarm optimization showed a noticeable variation in accuracy, the sequential robustified particle swarm optimization method ensured the optimal solution.

The mean values and standard deviations of each location and rotation error for robustified particle swarm optimization with noisy data are presented in Table 2 and Table 3, respectively. This confirms that most of the solutions are close to the optimal solution.

	X (mm)	Y (mm)	Z (mm)	3D (mm)	Omega (deg)	Phi (deg)	Kappa (deg)
LS	-2.8	2.2	-5.3	6.4	-0.0009	-0.0012	0.0023
rLS	-0.1	0.0	0.0	0.1	0.0000	-0.0001	0.0000
PSO, best	-2.8	2.2	-5.3	6.3	-0.0009	-0.0012	0.0023
PSO, worst	-2.8	2.2	-5.3	6.4	-0.0009	-0.0012	0.0023
rPSO, best	-0.1	0.0	0.0	0.1	0.0000	-0.0001	0.0000
rPSO, worst	0.2	-1.9	-2.0	2.8	0.0002	-0.0019	-0.0020
srPSO	-0.1	0.0	0.0	0.1	0.0000	-0.0001	0.0000

Table 1. Differences between estimated and known image locations with noisy simulated data (LS = Least squares method, rLS = robustified LS, PSO = particle swarm optimization, rPSO = robustified PSO, srPSO = sequential rPSO).

X (m)	Y (m)	Z (m)	Omega (deg)	Phi (deg)	Kappa (deg)
-0.0001	-0.0004	-0.0004	0.0004	0.0000	-0.0001

Table 2. Mean differences between estimated and known image rotations for the robustified particle swarm method with simulated noisy data.

X (m)	Y (m)	Z (m)	Omega (deg)	Phi (deg)	Kappa (deg)
0.0001	0.0005	0.0006	0.0005	0.0002	0.0001

Table 3. Standard deviations of differences between estimated and known image rotations for the robustified particle swarm method with simulated noisy data.

5. Discussion

Implementing particle swarm optimization is significantly easier than implementing the nonlinear least squares method, because there is no need to compute the partial derivatives needed for the Jacobian matrix. A particle swarm optimization implementation using the non-robustified sum of squared error as the optimizing criteria produced results very similar to the corresponding least squares method. Even though the results in Table 1 show no variation in differences along each coordinate axis, particle swarm optimization does not give exactly the same result when it is rerun. A small variation is seen in the 3D differences, where the best case shows a 6.3 mm difference in location and the worst case 6.4 mm. However, it can be said that, in our case, the camera location obtained by particle swarm optimization has sub-millimeter accuracy when compared with the least squares method. As expected, neither method could find the correct exterior orientation when applied to noisy data.

Adding a stochastic model (weight matrix) to both the robustified least squares method and robustified particle swarm optimization significantly improved the results. The authors did not find any publication presenting a similar approach using the particle swarm optimization. The least squares method achieved sub-millimeter absolute location accuracy despite noise in data. Particle swarm optimization achieved the same accuracy at best. However, there was visible variation in the results, and the worst case had an absolute 3D location error of 2.8 mm. Given that the distance from the image to the 3D points was approximately 60–80 meters, simulating the DJ3 Air medium tele camera, even the worst solution is relatively good. If Table 2 and Table 3 are examined, it can be seen that in most cases the solution is close to the best solution and only in some cases is it less accurate. For example, out of 25 runs, 15 cases (i.e., 60%) produced a solution very close or identical to the best case. In 10 cases, the solution was less accurate, but not all were close to the worst case.

One possible reason for such variation is the slow iteration, which might allow the elements of the weight matrix P to drift toward a non-optimal result. The parameter a in robustified particle swarm optimization needed to be 0.01, i.e., allowing only small changes, before the results become close to those from the robustified least squares method with sub-millimeter accuracy (see Figure 11). Whether the optimal solution was found seemed to be random. However, the cause was not thoroughly investigated. We tried resetting the elements of the weight matrix P when the error was smaller than a given threshold, but that did not completely eliminate the problem. Therefore, we examined sequential robustified particle swarm optimization. As can be seen from Table 1, the sequential robustified particle swarm method produced the same result as the robustified least squares method each time. However, this method requires a threshold for the output parameter that describes the sum of squared errors. In our case, we had some ideally accurate points and some noisy points making thresholding easy. In reality, however, all points may have noise, in which case determining a suitable threshold can be more difficult.

Particle swarm optimization is not an especially fast method for finding a solution. The execution time is highly dependent on the number of unknown parameters, the variable bounds, and the number of swarm members. In our case, the solution required up to 2200 iterations. However, the number of iterations varies significantly depending on how the initial locations of the swarm members happen to be placed in the parameter space. Therefore, the computing time can also vary randomly. While the least squares method, together with a direct method to find approximate values of the unknown parameters, took only seconds, particle swarm optimization could take minutes. For example, in our case, the average computation time with the robustified particle swarm optimization was 251 seconds (4 min 11 s) for solving the exterior orientation of one image. The sequential particle swarm optimization increased the computation time by only approximately 5 seconds, because the swarm size was significantly smaller in the second computation round. However, when the parameter bounds are wide, a larger swarm is needed to reliably find a solution.

The *particleswarm* function in MATLAB can be run in a parallel mode (CPU). We did not test that, but using parallel processing should reduce the computation time. As an example, Zhuo et al. (2023) reported that a speedup ratio of 2000 times can be achieved in some special cases with a GPU-based parallel implementation.

The reliability of the particle swarm method varied. In the case of creating simulation data, only two unknown parameters needed to be solved. This was an easy and fast task, and there were no failures even with the default parameters. In the case of solving exterior orientation parameters with particle swarm optimization, the first experiments failed completely. Tuning the options was essential in order to achieve a 100% success rate when the algorithm was run 25 times with the same options and bounds. However, it was indeed possible to find settings that led to a 100% success rate.

In our case, the search area for location was set quite large, i.e., a 6000 x 6000 m planimetric area with a height range between 0 and 110 m, simulating a situation where the location is not well known. Using such a large search area forced us to apply a large swarm. In addition, the search range for angles needed to be limited between [0, 62] degrees. We also conducted experiments to see whether stricter location bounds would allow increasing the angle range. Indeed, when the planimetric bounds were set to cover only a 200 x 200 m area including the correct location, and the elevation bounds were set to [50, 110], a 100% success rate was still achieved with an angle range of [0, 152] degrees. If the rotation bounds were [0, 180] degrees, there were 20 successful and 5 failed cases out of 25. In principle, the sequential method could be modified to solve cases in which success is possible by rerunning the algorithm repeatedly with the original parameters until at least the close solution is found. However, this would significantly increase computing time in the event of failures.

6. Conclusions

In this paper, the suitability of particle swarm optimization for solving the exterior orientation parameters of a single image was examined using simulated data. When the sum of squared errors was set as the optimization criteria in particle swarm

optimization, it was possible to find a set of settings that led to a 100% success rate with both ideal data and noisy data. Success in this case means that the estimated exterior orientation parameters were very close to the correct solution. The results were similar to those of the least squares solution.

In a robustified particle swarm method, a stochastic model was included in the optimization criterion by adding a weight matrix. The weights were iteratively updated based on the residuals. As a result, points that do not agree with the majority receive very small weights, diminishing their impact on the solution. In this case, however, only 60% of attempts produced the same result as the corresponding robustified least squares method, with sub-millimeter accuracy. In other cases, the result was close, but only with millimeter accuracy.

Finally, a sequential robustified particle swarm method was examined, in which the algorithm was rerun if the sum of squared errors was higher than expected. The swarm size was reduced, and the bounds of the unknown parameters were set near the solution from the first round. This enabled all attempts to find the same result as the robustified least squares method, with sub-millimeter accuracy.

To conclude, solving exterior orientation parameters reliably and accurately was possible after fine-tuning the algorithm’s options. In the case of robustified particle swarm optimization, the sequential version was necessary to achieve the best accuracy. Implementation with particle swarm optimization is much easier than with the traditional nonlinear least squares method, which also requires a direct method to obtain approximate values for the unknown parameters. However, the computing time with particle swarm optimization was significantly higher.

Acknowledgements

The authors would like to thank the Academy of Finland for financial support through the projects “Towards automatic road inspection” (No. 365584) and “Innovative methods for measuring and modelling the dynamic forest road quality” (No. 362926).

References

Abdel-Aziz, Y.I., Karara, H.M., 1971. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In: Proceedings of the Symposium on Close-Range Photogrammetry, pp. 1–18.

Chakraborty, S., Verma, A., 2024. Novel Stereo Visual Odometry Pipeline for Autonomous Robot Navigation on Video Data. In the *Proceedings of IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, 7 pages.

Ciampini, C., Petrillo, A., Zomparelli, F., 2024. An innovative method for human height estimation combining video images and 3D laser scanning. *Journal of Forensic Sciences*, 69(1), pp. 301-315.

El-Ashmawy, K.L.A., 2015. A comparison study between collinearity condition, coplanarity condition, and direct linear transformation (DLT) method for camera exterior orientation parameters determination. *Geodesy and Cartography*, 41(2), pp. 66-73.

Elashry, A., Toth, C., 2023. Improving camera pose estimation using swarm particle algorithms. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLVIII-M-3-2023, pp. 87-93.

Grussenmeyer, P., Al Khalil, O. 2002. Solutions for exterior orientation in photogrammetry, a review. *The Photogrammetric Record*, 17(100), pp. 615-634.

Guan, J., Li, J., Xi, J., 2021. Robust Computation of Trifocal Tensor Based on Hybrid Particle Swarm Optimization. In the *Proceedings of 3rd International Conference on Image Processing and Machine Vision (IPMV 2021)*, Hong Kong, China, pp. 35-41.

Jensen, K., Juhl, J., 2009. Gait analysis by multi video sequence analysis. *The Photogrammetric Journal of Finland*, 21(2), pp. 25-34.

John, V., Liu, Z., Mita, S., Xu, Y., 2019. Stereo vision-based vehicle localization in point cloud maps using multiswarm particle swarm optimization. *Signal, Image and Video Processing*, 13, pp. 805-812.

Jouybari, A., Bagherbandi, M., Nilfouroushan, F., 2024. Lever arm measurement precision and its impact on exterior orientation parameters in GNSS/IMU integration. *Journal of Geodetic Science*, 14: 20220179, 20 pages.

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. In the *Proceedings of the IEEE International Conference on Neural Networks,* Perth, Australia, pp. 1942-1945.

Kraus, K., 1997. Photogrammetry, Advanced Methods and Applications, Volume 2. Fred. Dümmler Verlag, Bonn, 466 pages.

Lee, J., Lee, E.-D., Tark, H.-O., Hwang, J.-W., Yoon, D.-Y., 2008. Efficient height measurement method of surveillance camera image. *Forensic Science International*, 177, pp. 17-23.

Li, X., Li, W., 2012. Obtaining approximate values of exterior orientation elements of multi-intersection images using particle swarm optimization. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XXXIX-B3, pp. 35-40.

Li, N., Hu, Z., Zhao, B., 2016. Flexible extrinsic calibration of a camera and a two-dimensional laser rangefinder with a folding pattern. *Applied Optics*, 55(9), pp. 2270-2280.

Li, W., Zhong, K., 2019. Application of improved particle swarm optimization algorithm in solving camera extrinsic parameters. *Journal of Modern Optics*, 66(18), pp. 1827-1835.

Li, L., Zuo, Z.-K., Bu, R.-F., Zhang, Y.-J., Wang, R.-M., Tan, E.-L., 2025. An Illumination-Adaptive Visual SLAM Algorithm with Lightweight Object Detection for Complex Dynamic Environments. *IEEE transactions on instrumentation and measurement*, 74: 2539822, 22 pages.

Liscio, E., Guryn, H., Le, Q., Olver, A., 2021. A comparison of reverse projection and PhotoModeler for suspect height analysis. *Forensic Science International*, 320, 12 pages.

Low, W., Nagarajan, R., Yaacob, S., 2010. Visual based SLAM using Modified PSO. In the *Proceedings of 6th International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 313-317.

Luhmann, T., Fraser, C., Maas, H.-G., 2016. Sensor modelling and camera calibration for close-range photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115, pp. 37-46.

Mukhopadhyay, A., Hati, S., 2017. Camera calibration from known correspondences using PSO. In the Computational Science and Engineering – Deyasi et al. (Eds), pp. 165-168.

Qin, R., Tian, J., Reinartz, P., 2016. 3D change detection – Approaches and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 122, pp. 41-56.

Ristanto, S., Nugroho, W., Sulistya, E., Suparta, G.B., 2022. Development of laboratory devices for real-time measurement of object 3D position using stereo cameras. *Physics Education*, 57, 025006, 12 pages.

Shi, Y., Eberhart, R.C., 1998. A modified particle swarm optimizer. In the *Proceedings of the IEEE international conference on evolutionary computation*, pp. 69-73.

Siddiqui, R., Khatibi, S. 2014. Visual Tracking using Particle Swarm Optimization. arXiv:1401.4648, Computer Vision and Pattern Recognition (cs.CV), 14 pages.

Stöcker, C., Nex, F., Koeva, M., and Gerke, M., 2017. Quality assessment of combined IMU/GNSS data for direct georeferencing in the context of UAV-based mapping. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W6, pp. 355-361.

Triggs, W, McLauchlan, P., Hartley, R., Fitzgibbon, A., 2000. Bundle adjustment – A modern synthesis. In the Vision Algorithms: Theory and Practice – W. Triggs, A. Zisserman, and R Szeliski (Eds), LNCS, Springer Verlag, pp. 298-375.

Wachowiak, M. P., Smolíková, R., Zheng, Y., Zurada, J., Elmaghraby, A. S., 2004. An Approach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization. *IEEE transactions on evolutionary computation*, 8(3), pp. 289-301.

Wang, J., Zhang, C., 2026. Research on targetless automatic calibration method of LiDAR and camera. *Measurement*, 258, 119002, 12 pages.

Zhu, D., Li, R., Zheng, Y., Zhou, C., Li, T., Cheng, S., 2025. Cumulative Major Advances in Particle Swarm Optimization from 2018 to the Present: Variants, Analysis and Applications. *Archives of Computational Methods in Engineering*, 32, pp. 1571-1595.

Zhuo, Y., Zhang, T., Du, F., Liu, R., 2023. A parallel particle swarm optimization algorithm based on GPU/CUDA. *Applied Soft Computing*, 144, 110499, 12 pages.