On the Efficiency of Population-Based Optimization in Finding Best Parameters for RGB-D Visual Odometry

Submitted: 4th February 2019; accepted: 15th April 2019

Aleksander Kostusiak, Piotr Skrzypczyński

DOI: 10.14313/JAMRIS/2-2019/13

Abstract:
Visual odometry estimates the transformations between consecutive frames of a video stream in order to recover the camera's trajectory. As this approach does not require to build a map of the observed environment, it is fast and simple to implement. In the last decade RGB-D cameras proliferated in robotics, being also the sensors of choice for many practical visual odometry systems. Although RGB-D cameras provide readily available depth images, that greatly simplify the frame-to-frame transformations computation, the number of numerical parameters that have to be set properly in a visual odometry system to obtain an accurate trajectory estimate remains high. Whereas setting them by hand is certainly possible, it is a tedious try-and-error task. Therefore, in this article we make an assessment of two population-based approaches to parameter optimization, that are for long time applied in various areas of robotics, as means to find best parameters of a simple RGB-D visual odometry system. The optimization algorithms investigated here are particle swarm optimization and an evolutionary algorithm variant. We focus on the optimization methods themselves, rather than on the visual odometry algorithm, seeking an efficient procedure to find parameters that minimize the estimated trajectory errors. From the experimental results we draw conclusions as to both the efficiency of the optimization methods, and the role of particular parameters in the visual odometry system.

Keywords: *Particle Swarm Optimization, Evolutionary Algorithm, Visual Odometry, RGB-D*

1. Introduction

The new generation of RGB-D cameras allowed for development of new solutions for Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) [24]. Sensors that are available on the market since 2010, when the Kinect v1 has been introduced, are inexpensive, but yield accurate measurements, sufficient for indoor localization [21]. A VO algorithm computes camera motion between the consecutive keyframes, and estimates the trajectory (consecutive camera poses) without building any explicit representation of the environment [23]. Visual odometry is conceptually and practically simpler than a full SLAM system. In VO the estimated trajectory of the camera is not guaranteed to be optimal in the light of all collected data. Usually the estimated trajectory drifts with the distance traveled. However, an implementation of the VO method can be adopted as a "front-end" for a full SLAM system, which is then combined with an optimization-based "back-end" for post-processing of the obtained camera trajectory. This trajectory is represented as a pose-graph whose edges are constraints between the camera poses, and is optimized to obtain a best explanation of all the collected frame-to-frame transformations [5].

In the classic approach to VO monocular images are often used [23], which requires sophisticated algorithms to estimate the spatial transformations between the neighboring frames from the correspondences between sets of 2D points that are determined in the image space [19]. The problem becomes much easier if RGB-D frames are used, as they provide readily available depth information that can be used to turn the 2D visual features into 3D points, keeping however the correspondences established between the 2D features using local point descriptors. Then, estimation of the transformation (rototranslation) between the two sets of 3D points can be accomplished applying closed formulas [15].

In spite of their simplicity RGB-D VO algorithms are very useful in several areas of robotics. A VO system can be applied as a stand-alone localization method [20], used as a front-end in pose-based SLAM [5], exploited to pre-process RGB-D data in global localization [26], or combined with other methods of estimating the robot's pose [12].

There are many different solutions to the problem of visual odometry using RGB-D frames. An early implementation employing sparse point features has been presented in [4], while a feature-less, dense approach was proposed in [16]. The work of Endres *et al.* [10] is often considered a standard implementation of the feature-based RGB-D VO matched to an optimization back-end, finally solving the full SLAM problem. It is difficult to say exactly what blocks a RGB-D VO system should be composed of, and according to which principles one should choose parameters in these blocks. In [5] we tackled this problem at the level of the building blocks, showing that poor performance of the VO front-end hardly can be compensated by a more developed optimization back-end. This result makes it clear that parameters of the VO algorithm are of pivotal importance for any RGB-D visual navigation system.

On the other hand, in some of the best known RGB-D-based localization systems, e.g. the system developed by Endres *et al.* [9, 10], numeric parameters were selected for a given RGB-D sequence by exhaustive search in the parameter space. Those parameters governed the feature detector parameters, feature matching and outliers detection. While such a strategy made it possible to obtain impressive results in terms of the trajectory estimation accuracy on benchmark RGB-D sequences [25], it is inefficient due to the time required by the exhaustive search when the objective function is the minimal discrepancy between the estimated and the reference (i.e. ground truth) trajectory. Moreover, the found parameters are appropriate only for the given experiment, and it is unclear if they are transferable to any other environment or sequence of RGB-D frames.

Therefore, we try to find the best parameters for a simple RGB-D VO system using a more efficient approach, which gives also some promise of generalization to other, environments, not seen before by the localization system. This approach is population-based optimization, which evaluates a population of possible solutions (called individuals or particles) in order to find the best one, but at the same time applies some heuristic strategy to create an offspring that should inherit properties from the best solutions found so far, allowing for further improvements. This very general scheme may be implemented in many different ways, leading to Genetic Algorithms [14], Evolutionary Algorithms [3], Particle Swarm Optimization [7] or the recent Cuttlefish Algorithm [8]. Algorithms from this broad family are considered for our parameters search problem, due to their global optimization ability, and because they can handle complex, non-differentiable search spaces, still showing good explorative properties.

Although classic genetic algorithms have proven to be useful search methods in many robotics-related applications, it is not clear how a genetic algorithm should be configured and parametrized for our problem, as the results largely depend on the population size and the strategy of creating new individuals. Therefore, we apply two algorithms that require a minimal setup with respect to the meta-parameters (i.e. parameters of the optimization algorithm itself, not the VO method). These are the Particle Swarm Optimization [7] and an Evolutionary Algorithm in the ecology-inspired variant proposed by Annunziato and Pizzuti [2]. These two methods can deal with very difficult optimization problems [22] and they have proven their usefulness in our earlier research, for instance in the development of a dynamics model for simulation of a hexapod robot [6].

This article is an improved and extended version of the conference paper [17] published in Polish, which provides more quantitative results (e.g. RPE plots that were omitted in the conference paper), but introduces also a novel on-line optimization procedure for the point feature detector.

2. Visual Odometry Architecture

2.1. System Structure

The RGB-D VO system used in this research employs the popular OpenCV library for most of the RGB-D data processing tasks. The VO structure (Fig. 1) is

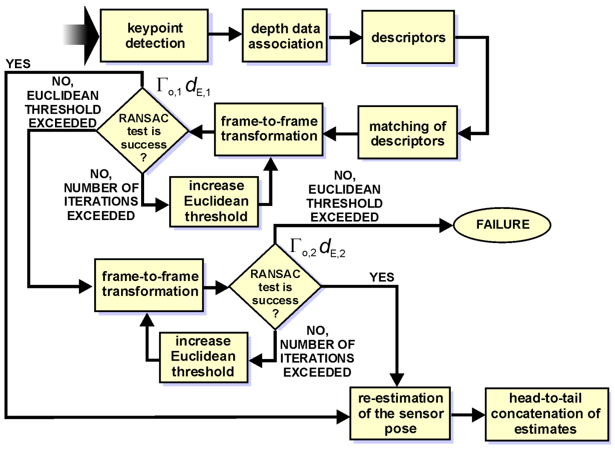
keypoint detection
depth data association
descriptors
matching of descriptors
frame-to-frame transformation
RANSAC test is success ?
$\Gamma_{o,1}\ d_{E,1}$
YES
NO, EUCLIDEAN THRESHOLD EXCEEDED
NO, NUMBER OF ITERATIONS EXCEEDED
increase Euclidean threshold
NO, EUCLIDEAN THRESHOLD EXCEEDED
FAILURE
$\Gamma_{o,2}\ d_{E,2}$
frame-to-frame transformation
RANSAC test is success ?
YES
increase Euclidean threshold
NO, NUMBER OF ITERATIONS EXCEEDED
re-estimation of the sensor pose
head-to-tail concatenation of estimates

Fig. 1. Block scheme of the simple RGB-D visual odometry system

embedded into the investigated population-based optimization method serving as the main part of the fitness evaluation block.

Therefore, during the initialization process, the program sets the few meta-parameters of the population and particles/individuals in the investigated algorithm, then loads the first RGB image from the assigned sequence, and extracts keypoints of the salient visual features. Only keypoints with corresponding depth information are harvested for further processing, while these located in the areas of degraded depth data are discarded. This process is repeated for all incoming RGB-D data frames. Then, point features from the newest frame are matched to the ones from the previous frame with cross-checking of the matchings to clear out spurious associations as soon as possible [18].

We use RANSAC [13] procedure twice to remove the remaining bad-matches while transformations between the consecutive frames are computed [18]. In each of the RANSAC instances, we draw three point pairs, which is the minimal number of points required by the Kabsch algorithm [15] to determine the rototranslation between two successive frames. Once a candidate transformation is computed, we apply this transformation to the whole set of keypoints from the current frame and check how far these points are located from their matching counterparts from the previous frame. Feature points lying too far from their counterparts, according to a given Euclidean distance threshold, are considered as outliers. If the ratio between the number of outliers and inliers is not satisfactory (i.e. we have too few inliers), then we discard the candidate transformation and draw another three points. However, if too many draws were needed, then we increase the distance threshold. All point pairs that passed through the RANSAC filtration are then used to calculate the rotation and translation between the considered RGB-D frames. Finally, the camera pose is updated by concatenating the newly computed rototranslation with the existing camera pose.

Motivated by the results of previous experiments [18], we use the AKAZE algorithm [1] for feature detection and description. This detector-descriptor allows to achieve point feature detection and matching results that are very similar to the popular SURF detector-descriptor performance, but with the use of a binary descriptor, which requires less computation power in the matching process. Moreover, the AKAZE license allows for free usage even in commercial applications, while SURF is patented.

The AKAZE detector searches for maxima of normalized and approximated determinant of image Hessian, which are then compared to the threshold value τ_A to find keypoints at different scale levels. The scale space is built in a similar manner as in SIFT, but with the use of Fast Explicit Diffusion algorithm instead of a Gaussian kernel. To detect an interest point the determinant of Hessian is calculated for each image in the nonlinear scale space and weighted accordingly. The local maxima are picked as salient point candidates and compared with other candidate points.

The AKAZE descriptor is very similar to the BRIEF descriptor, with the difference that instead of particular pixel values it compares regions average intensity and average horizontal and vertical derivatives of the image intensity function. The computation of AKAZE descriptor starts with estimating the orientation by using a histogram method, and the pattern is rotated accordingly. Finally, the descriptor vector is generated by performing binary tests of average areas and the mean of the horizontal and vertical derivatives in the areas.

2.2. System Parameters

Both the efficiency of keypoint matching between the consecutive frames, and the small residual distance errors between the sets of corresponding features are of great importance for achieving good quality of the estimated camera trajectory. On the other hand – the strive for small residual errors can lead to rejections of many matched pairs in the first RANSAC loop, which further can lead to computation of the rototranslation by using a relatively small number of keypoints, which in turn decreases quality of the final solution.

Thus, for achieving a good quality trajectory the selection of RANSAC parameters is crucial: the Euclidean distance thresholds $d_{E,1}$ and $d_{E,2}$ (respectively for the first and the second RANSAC loop), and the inliers to outliers ratios $\Gamma_{o,1}$ and $\Gamma_{o,2}$ (also for the first and the second RANSAC loop). The $d_{E,1}$ and $d_{E,2}$ parameters are the initial thresholds in the respective RANSAC procedures. The VO system needs also to set the AKAZE detection threshold τ_A adequately to the scene, because the number of detected features directly depends on this parameter. In turn, the number and quality of the point features influences the quality of the recovered trajectory. Eventually, the five described above parameters constitute the vector $\theta = [d_{E,1}, d_{E,2}, \Gamma_{o,1}, \Gamma_{o,2}, \tau_A]$, and are optimized by population-based methods. Two strategies are applied to the problem of finding the best parameters:

- all parameters are optimized together in a five-dimensional search space,
- parameters are optimized hierarchically – after fixing the best RANSAC thresholds, we only seek the best AKAZE detector threshold, assuming that it is more environment-specific.

3. Population-based Optimization Methods

3.1. Objective Functions

In the optimization process, it is important to choose a suitable function that shows how much the particles/individuals fit to the environment. In the investigated case, the VO system is the environment, and the fitness function has to be related to the quality of the estimated trajectory. For this reason, we chose the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE) as the objective functions. They were proposed in [25] and are used commonly in robotics research for SLAM and VO evaluation. The ATE specifies the translational errors between the estimated camera poses along the trajectory, and the ground truth trajectory poses. The ground truth trajectory is obtained from an external motion capture system [21]. Because ATE compares absolute distances between poses taken from two synchronized trajectories, these trajectories have to be aligned prior to ATE computation. This is implemented by finding a rotation between these two rigid sets of points that minimizes the distance between them assuming a common starting point [25]. Conversely, the RPE calculates the difference in transformation (that is why it has rotational and translational parts) which would exist after following the estimated trajectory and the ground truth trajectory independently for the given number of frames (or time amount, in our case 1 s), and then computing the rototranslation between the estimated trajectory pose and its counterpart from the ground truth one.

In order to determine the errors we need to have the estimated trajectory $\mathbf{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_k\} \in$ SE(3), and the ground truth trajectory $\mathbf{T}^{\mathrm{gt}} = \{\mathbf{T}_1^{\mathrm{gt}}, \mathbf{T}_2^{\mathrm{gt}}, \ldots, \mathbf{T}_k^{\mathrm{gt}}\} \in$ SE(3), where $\mathbf{T}_i$ and $\mathbf{T}_i^{\mathrm{gt}}$ are camera poses for the i-th frame in the sequence expressed by 4×4 homogeneous matrices, and k is the number of camera poses in the trajectory. The ATE for the i-th frame is computed as:

$$\mathbf{E}_i^{\mathrm{ATE}} = \left(\mathbf{T}_i^{\mathrm{gt}}\right)^{-1} \mathbf{T}_i \tag{1}$$

and the ATE RMSE value for the whole trajectory is computed as the Root Mean Square Error of (1) for all nodes of $\mathbf{T}$ and $\mathbf{T}^{\mathrm{gt}}$:

$$\mathrm{ATE}^{\mathrm{RMSE}} = \sqrt{\sum_{i=1}^{k} \left(\mathbf{E}_i^{\mathrm{ATE}}\right)^2 \frac{1}{k}}. \tag{2}$$

The RPE for i-th frame is given by the equation:

$$\mathbf{E}_i^{\mathrm{RPE}} = \left((\mathbf{T}_i^{\mathrm{gt}})^{-1}\mathbf{T}_{i+1}^{\mathrm{gt}}\right)^{-1} \left(\mathbf{T}_i^{-1}\mathbf{T}_{i+1}\right). \tag{3}$$

Then, we can obtain the relative translational $\mathrm{RPE}_{t(i)}$ or rotational $\mathrm{RPE}_{r(i)}$ error at i-th frame taking the translational or rotational part of $\mathbf{E}_i^{\mathrm{RPE}}$ and computing the Euclidean norm or Euler angle, respectively. Similarly to ATE, the RPE RMSE for the whole trajectory is obtained calculating the RMSE over the respective per-frame errors for the whole trajectories:

$$\mathrm{RPE}_r^{\mathrm{RMSE}} = \sqrt{\sum_{i=1}^{k} \left(\mathrm{Rot}\left(\mathbf{E}_i^{\mathrm{RPE}}\right)\right)^2 \frac{1}{k}}, \qquad (4)$$

$$\mathrm{RPE}_t^{\mathrm{RMSE}} = \sqrt{\sum_{i=1}^{k} \left(\mathrm{Trans}\left(\mathbf{E}_i^{\mathrm{RPE}}\right)\right)^2 \frac{1}{k}}, \qquad (5)$$

where $\mathrm{RPE}_r^{\mathrm{RMSE}}$ and $\mathrm{RPE}_t^{\mathrm{RMSE}}$ stand for the rotational and translational RPE RMSE, respectively, while $\mathrm{Rot}(.)$ is the operator that extracts rotational part of the transformation as the Euler angle, and $\mathrm{Trans}(.)$ is the operator that extracts translational part of the same transformation and calculates its Euclidean norm.

Having the ground truth trajectory $\mathbf{T}^{\mathrm{gt}}$, the estimated trajectory $\mathbf{T}(\theta)$ that depends on the VO parameters θ, and the ATE and RPE given by (1) and (3), respectively, we define the optimization problem of VO system parameters with the use of ATE:

$$\underset{\theta}{\operatorname{argmin}}\, F_{\mathrm{ATE}} = \sum_{i=1}^{k} \left(\mathbf{T}_i^{\mathrm{gt}}\right)^{-1} \mathbf{T}_i(\theta). \qquad (6)$$

Similarly, for the translational part of RPE we minimize the following form:

$$\underset{\theta}{\operatorname{argmin}}\, F_{\mathrm{RPE}_t} = \sum_{i=1}^{k} \mathrm{Trans}\left(\left((\mathbf{T}_i^{\mathrm{gt}})^{-1} \mathbf{T}_{i+1}^{\mathrm{gt}} \right)^{-1} \left(\mathbf{T}_i^{-1}(\theta)\mathbf{T}_{i+1}(\theta)\right)\right). \qquad (7)$$

3.2. Particle Swarm Optimization

This approach simulates a flock of birds flying around in search of a cornfield [7]. The block scheme in Fig. 2 shows the general structure of this method. During the initialization stage, we draw a population of parameters that are represented by m particles. Each particle has n parameters (in our case n=5 or n=1, depending on the considered optimization strategy) that are the particle's position in the n-dimensional space, and the corresponding number of velocities. The velocities have no direct physical interpretation in the VO algorithm, but they control exploration of the search space – the higher the velocity, the bigger the parameter changes between the consecutive iterations.

Each particle is evaluated to determine how good the parameter set it defines in the search space fits to the "environment", which in our case is the VO problem. Thus, the fitness measure is defined by the VO performance. Namely, the fitness value depends on the translational RPE RMSE or ATE RMSE value, according to the chosen optimization variant. For the m-th particle, we keep the best parameter vector found up to the current iteration of the algorithm, as well as the globally best parameter vector found by any particle. From these data we compute the velocities and positions of each particle in the parameter space for the next iteration, as given by the formulas:

$$\begin{aligned} \mathbf{v}_m^{i+1} &= \mathbf{v}_m^i \; +c_1 * \mathrm{rand}() * (\mathbf{l}_{\mathrm{best}} - \mathbf{p}_m^i) \\ &\qquad +c_2 * \mathrm{rand}() * (\mathbf{g}_{\mathrm{best}} - \mathbf{p}_m^i), \\ \mathbf{p}_m^{i+1} &= \mathbf{p}_m^i + \mathbf{v}_m^{i+1}, \end{aligned} \qquad (8)$$

where $\mathbf{v}_m^i$ is the velocity of the m-th particle $\mathbf{p}_m^i$, c_1 and c_2 are constant values, $\mathbf{l}_{\mathrm{best}}$ is the best parameter vector found by the given particle, $\mathbf{g}_{\mathrm{best}}$ is the globally best parameter vector, while $\mathbf{p}_m^{i+1}$ and $\mathbf{v}_m^{i+1}$ are the new positions and velocities of the m-th particle. We perform these operations until one of the stop criteria is reached. These stop criteria are:

1) a satisfying fitness value has been achieved,
2) no fitness improvement has been noticed for several consecutive iterations,
3) the maximum allowed number of iterations has been exceeded.

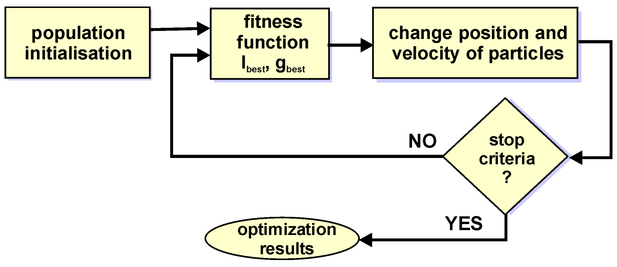
population initialisation
fitness function $\mathbf{l}_{best}$, $\mathbf{g}_{best}$
change position and velocity of particles
NO
stop criteria ?
YES
optimization results

Fig. 2. Block scheme of the PSO algorithm

The implemented PSO is the most simple variant of the algorithm, which is characterized by fast computation of the particle updates and high speed of convergence. Unfortunately, fast convergence can lead to the loss of diversity among the particles, resulting in premature convergence. A number of solutions to this problem has been proposed in the literature, such as the perturbed PSO algorithm [27], that introduces additional perturbance to the global best solution in order to maintain diversity. However, more complicated PSO variants are more time consuming, because they perform additional computations for each particle update. Hence, we stick with the canonical PSO variant, observing however the behavior of the particles (see Fig. 7). The visualizations suggest that sufficient diversity is preserved among the particles till the final iteration, even though the stop criteria is the lack of fitness improvement for several iterations.

The PSO meta-parameters have been chosen in such a way, that results should be obtained in a reasonable amount of time (several hours). We have set c_1=c_2=2 and limited the maximum allowed velocity for particles to the range from -0.005 to 0.005. The velocities are initialized as random values close to zero, as suggested in [11]. We have used 40 particles to effectively use available threads for parallel computation for maximum 20 iterations. In all experiments the PSO optimization has ended detecting no improvement for several iterations, and newer reached the maximum iterations limit.

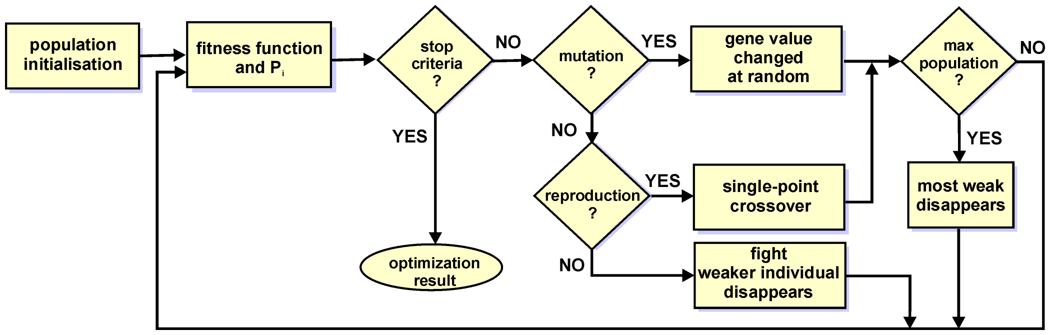

Fig. 3. Block scheme of the evolutionary algorithm

3.3. Evolutionary Algorithm

As an alternative to the PSO algorithm we have chosen an variant of the Evolutionary Algorithm (EA), a self-adapting algorithm inspired by natural ecosystems [2]. Contrary to typical Genetic Algorithms, it requires setting a minimal number of parameters. Few changes have been made in order to adopt the algorithm described in [2] to our problem. We use real numbers (of `float` type) instead of binary strings to code the genome. Initially, we specify the maximum size of the population r and the number of individuals in the first generation, for which we draw the initial parameters (genomes). Successive generations are created and evaluated in a loop until one of the stop criteria is met. The stop criteria are exactly the same as used for PSO. In the loop we draw the individuals that interact with other individuals, while the remaining individuals undergo a mutation process. The individuals are drawn with the probability P_i (where i is the iteration number), which is equal to the ratio between the population size in the i-th generation, and the maximum allowed population size. Among the individuals that have to interact with the others we draw with the same probability P_i those that will fight with others, while the remaining ones can reproduce. In the fight interaction the stronger individual (having better fitness value) always wins, while the weaker one disappears. The reproduction is accomplished by randomly exchanging parts of the genomes in a single-point crossover operation. The crossover point is drawn from a normal distribution. Mutation is implemented as initiation of a new individual with one gene randomly changed with respect to its "parent" individual, which in this action is also preserved. If any of these actions would lead to exceeding the maximum population size limit, then the weakest individual from the population disappears. For fair comparison we set parameters of EA to operate on a population of approximately the same size as in PSO. In our experiments, the initial number of individual equals 10, and their maximal number is 40. We set the maximum number of iterations to 20.

4. Off-line Optimization Results

As an important aim of our research was to demonstrate that the VO parameters optimized using the proposed approach are transferable between different environments, we have used two different sequences of RGB-D frames: *fr1_desk* and *fr1_room* from the popular TUM RGB-D Benchmark [25] as the learning sequences. Through the whole paper the parameter set obtained using the *fr1_desk* sequence is denoted OP1, while the one obtained using the *fr1_room* sequence is denoted OP2. For verification of the results we used the *putkk_Dataset_1_Kin_1* sequence from an entirely different publicly available dataset [21], which was newer used for optimization.

Tab. 1. Number of point features per frame detected by the VO with the AKAZE detection threshold τ_A optimized by the Evolutionary Algorithm with ATE-based fitness function

RGB-D data sequence	min. num. of points	aver. num. of points	max. num. of points
fr1_desk	105	659	1435
fr1_room	12	509	1301
putkk_Dataset _1_Kin_1	42	328	795

The three RGB-D sequences used in our research are characterized by different size of the scene, different dynamics of the moving camera, and different objects being observed. All that results in different numbers of feature points that are detected in a single RGB-D frame (Tab. 1). In the TUM RGB-D Benchmark sequences the Kinect sensor was moved by hand, slowly in *fr1_desk* and much faster in *fr1_room*, with the trajectory spanning much larger volume of space in the latter case. Contrarily, in the PUTKK dataset the Asus Xtion sensor was attached to a wheeled robot, which resulted in fast, but smoother motion. Thus, the minimal number of features extracted from a frame is much smaller for *fr1_room*, mostly due to fast rotations of the camera that are present in this sequence. The number of features never drops to such a small value for *putkk_Dataset_1_Kin_1*, but the average and maximum number of features are smaller than for both TUM RGB-D Benchmark sequences due to the larger room size and the limited range of depth perception in the RGB-D sensor. Example visualizations of the environments used in the research are depicted in Fig. 4.

Our experiments began with optimization of all five system parameters: the AKAZE detection threshold τ_A and the RANSAC parameters. We used the PSO algorithm configured with the ATE-based objective

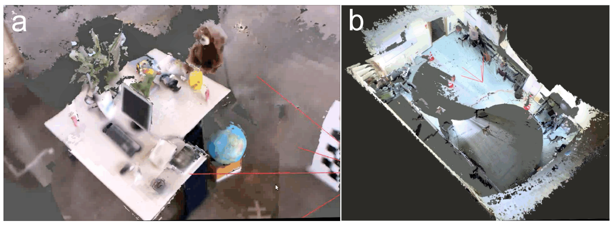

Fig. 4. Visualization of the colored point clouds registered with the ground truth trajectories from the *fr1_desk* sequence (a), and the *putkk_Dataset_1_Kin_1* sequence (b)

function. The optimization session did not exceed nine iterations. Table 2 contains the optimal parameter values found in this experiment. The ATE RMSE and RPE RMSE scores for these parameter sets, and their manually entered counterparts are collected in Tab. 3.

Tab. 2. VO parameters that got optimized and their best values

Method	$d_{E,1}$	$\Gamma_{o,1}$	$d_{E,2}$	$\Gamma_{o,2}$	τ_A
OP1	0.059	0.812	0.042	0.836	0.0021
OP2	0.042	0.916	0.012	0.855	0.0013
Manually	0.03	0.80	0.003	0.80	0.002

Tab. 3. ATE RMSE and RPE RMSE comparison for two sequences

	fr1_room		
Error metric	manually	OP1	OP2
ATE RMSE [m]	1.573	1.728	0.288
Trans. RPE RMSE [m]	0.348	0.358	0.115
Rot. RPE RMSE [°]	22.632	26.201	2.529
	Dataset1_Kin1		
Error metric	manually	OP1	OP2
ATE RMSE [m]	0.829	0.912	0.697
Trans. RPE RMSE [m]	0.019	0.020	0.013
Rot. RPE RMSE [°]	0.374	0.407	0.248

Differences in the quality of the frame-to-frame transformations estimation for the different parameter vectors are clearly visible on the plots on the RPE RMSE (Fig. 5), that shows the "local" quality of the trajectory estimation, i.e. it neglects the influence the inaccuracy of the previous parts of the trajectory has on the current camera pose. It is apparent from the comparison of these plots that the parameters optimized on a challenging enough sequence (i.e. OP2 that involved much faster motion of the Kinect and more diversified objects in the field of view) make it possible to achieve relative translations without large errors.

To complete the quantitative results for the PSO experiment we show also the qualitative results in the form of trajectories with the ATE visualized on them. Figure 6 presents the ATE plots for *fr1_desk* sequence in the first row, the *fr1_room* sequence in the second row, and for the verification sequence *putkk_Dataset_1_Kin_1* in the third row.

To demonstrate how the particles behave in the

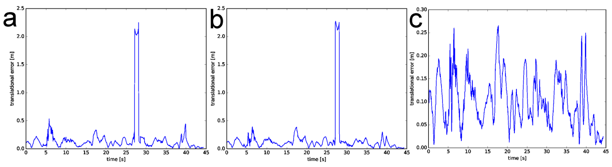

Fig. 5. Plots of translational RPE measured on *putkk_Dataset_1_Kin_1* for the VO parameters set manually (a), and optimized using *fr1_desk* (b) or *fr1_room* (c)

multi-dimensional search space we show in Fig. 7 the evolution of particles during the OP2 experiment. Red dots represent initial positions of particles, blue dots – transitional positions, and green dots the final positions achieved when the stop criteria occurred. The bigger black dot represents the best particle, i.e. the optimal parameters vector.

We decided to reuse obtained RANSAC filtration parameters in the next tests because the parameters obtained in the OP2 optimization process on *fr1_room* sequence allowed to achieve the best results and changed insignificant for different environments. Hence, in further experiments we have only optimized the AKAZE detector τ_A parameter, but this time with the use of two different approaches inspired by nature. Table 4 shows the best values of the τ_A parameter obtained after optimization with the use of PSO algorithm with the alternative ATE-based and RPE-based objective functions, and with the use of the EA with only RPE-based objective function. Table 5 collects error values of the VO system while using these parameters. The obtained detection threshold τ_A is significantly lower in case of using ATE RMSE instead of RPE RMSE in optimization and thus allows to take into account more points during trajectory estimation, but in exchange for slightly larger temporary RPE errors.

Tab. 4. The AKAZE detector parameter τ_A and its optimal values according to different optimization variants

Method and detection threshold	PSO ATE 0.000623	PSO RPE 0.001367
Method and detection threshold	EA ATE 0.000703	EA RPE 0.001412

However, during verification, this does not allow to achieve significantly better ATE RMSE results. For this reason, it is better to use the detection threshold obtained with the use of RPE-based objective function, because this accelerates the calculations. In Tab. 5 we have also included results for the VO system with SURF detector-descriptor, for which the same optimization procedure was applied to the detection threshold. This demonstrates that our approach can be transferred to visual odometry systems of different parameters.

Figure 8 presents the comparison of the recovered trajectories with the ATE visualized. In general, all sets of parameters used in the experiment allowed for correct trajectory estimation. It follows that both investigated optimization methods are suitable for automa-

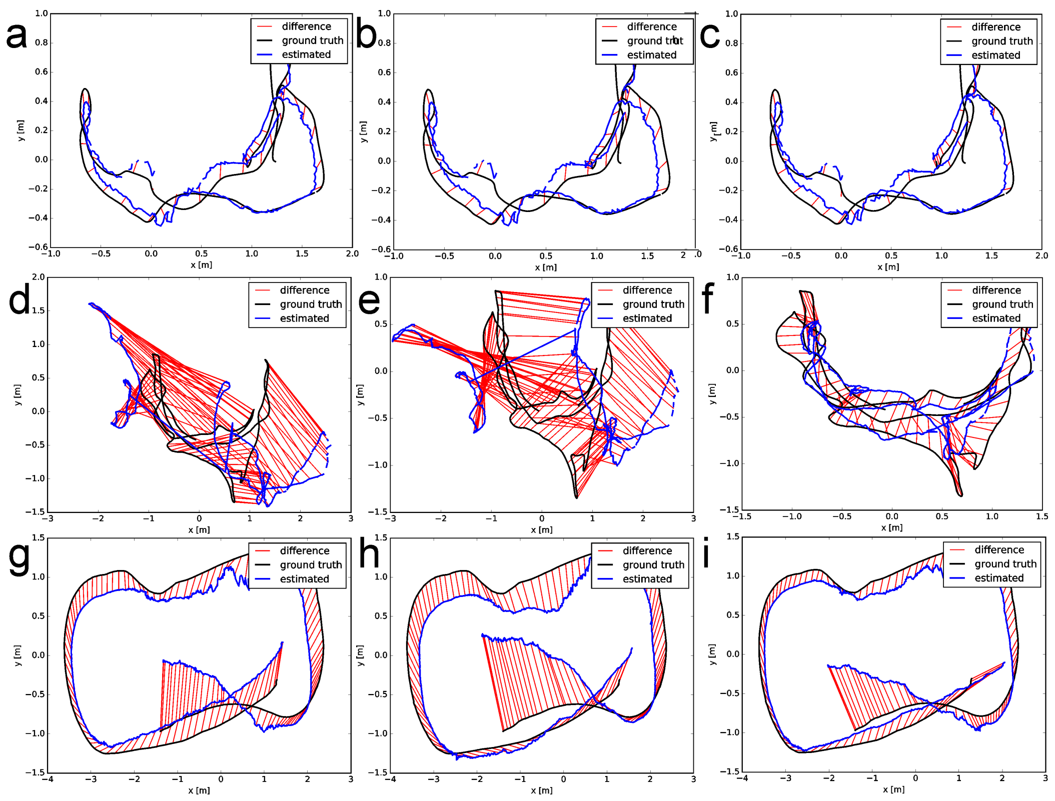

Fig. 6. ATE error plots (black lines represent the ground truth trajectory, blue lines the estimated trajectory, and red segments the Euclidean errors). Subfigures (a), (b), (c) are obtained for the *fr1_desk* sequence, (d), (e), (f) for the *fr1_room* sequence, and (g), (h), (i) for the *putkk_Dataset_1_Kin_1* sequence. The parameters were either set manually in (a), (d) and (g), or obtained by PSO optimization in the OP1 experiment (b), (e) and (h), or in the OP2 experiment in (c), (f) and (i)

Tab. 5. Trajectory estimation results for τ_A obtained through optimization

	fr1_room					*putkk_Dataset1_Kin1*			
Error metric	PSO ATE	PSO RPE	EA ATE	EA RPE	SURF EA RPE	PSO ATE	PSO RPE	EA ATE	EA RPE
ATE RMSE [m]	0.290	0.292	0.295	0.312	0.369	0.662	0.614	0.624	0.628
Trans. RPE RMSE[m]	0.120	0.114	0.119	0.115	0.122	0.009	0.012	0.009	0.013
Rot. RPE RMSE [°]	2.403	2.585	2.382	2.632	2.615	0.172	0.232	0.175	0.239

tic search of the best parameter values for a RGB-D VO system. However, there are some important differences. The PSO algorithm searched through a significantly larger area in the parameter space. In all experiments the optimization with PSO was about fivefold longer than with the use of the EA (58 h versus 11 h for the ATE-based objective function). Moreover, configurations with the RPE-based objective function required about 10% less time than those employing the ATE-based objective function. In all these cases the program running under Linux used all 12 threads available on the i5 CPU desktop PC, with nearly 100% CPU load. Having in mind the computation time and similar results, we recommend EA with the RPE-based objective function as a handy software tool to quickly find reasonable parameters of a VO system. However, for a more thorough optimization (e.g. for a particular environment that does not change much in time) the use of PSO is recommended, due to its better exploratory properties. It seems that the use of ATE-based objective function does not have any significant positive effect on the final results, thus the RPE-based objective function is recommended for both optimization methods, as it allows for faster computation of the fitness values.

5. On-line Optimization of VO Parameters

The results obtained in the off-line experiments encouraged us to go one step further and to try tuning parameters of the AKAZE detector on-line, which should make it possible to adapt the VO system to changing environment properties, such as the amount of texture that directly influences the number of point features, or changing lighting. In this case we were also optimizing the AKAZE percentile parameter τ_P, besides τ_A detection threshold. From the initial PSO experiments we have noticed that the obtained estimations were much worse in scenarios where less than 250 keypoints remained after RANSAC filtration. Therefore, we tried to use the PSO method to find parameters allowing for retrieving a satisfying number of keypoints in those demanding frames. For the sake of speed, the on-line optimization was initialized only

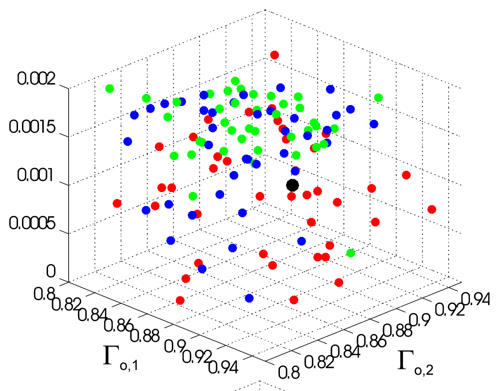
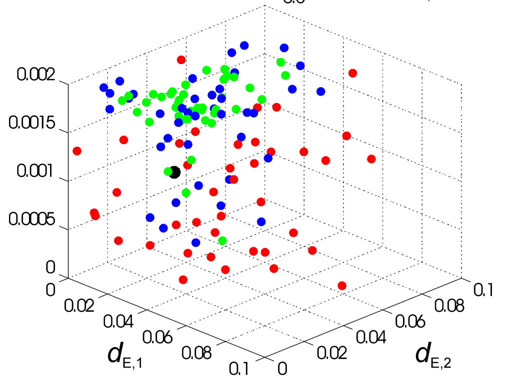

Fig. 7. Evolution of particles during optimization on *fr1_room* sequence. In (a) the x-axis shows $\Gamma_{o,1}$, and the y-axis $\Gamma_{o,2}$, while in (b) x-axis is for $d_{E,1}$, and y-axis for $d_{E,2}$. In both subfigures the z-axis shows the τ_A parameter. See main text for the meaning of the color dots

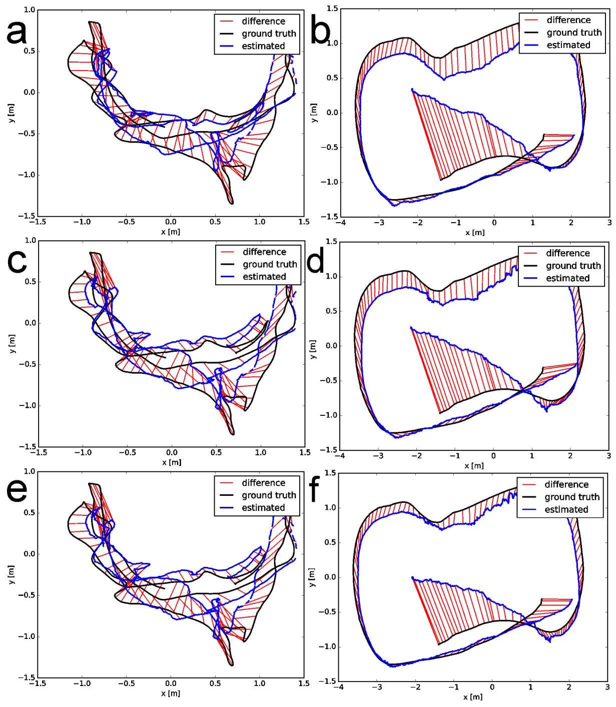

Fig. 8. Comparison of ATE results: *fr1_room* in (a), (c), (e), and *putkk_Dataset1_Kin1* in (b), (d), (f). The first row (a,b) shows PSO/ATE results, the second row (c,d) PSO/RPE results, and the third row (e,f) GA/RPE results

when we had less than 50 points during the VO operation.

In the experiment only RPE was used as the fitness value, as it reflects the local changes to the accuracy of the estimated trajectory. This time we also used 40 particles, but the first one was reserved for the already best parameters vector, found during off-line optimization. Next 11 particles (the used PC can run 12 threads simultaneously) constituted a set of previously best parameters, and they were calculated only once during the PSO optimization. In Tab. 6 we present results obtained for the same three sequences we already used in the off-line experiments, while Fig. 9 presents the corresponding ATE and RPE plots.

Tab. 6. Trajectory estimation results for the AKAZE parameters obtained through on-line optimization

sequence	ATE [m]	Trans. RPE [m]	Rot. RPE [°]
fr_desk	0.1225	0.0758	2.5094
fr_room	0.3233	0.1212	2.5794
Dataset_1	0.5000	0.0097	0.1739

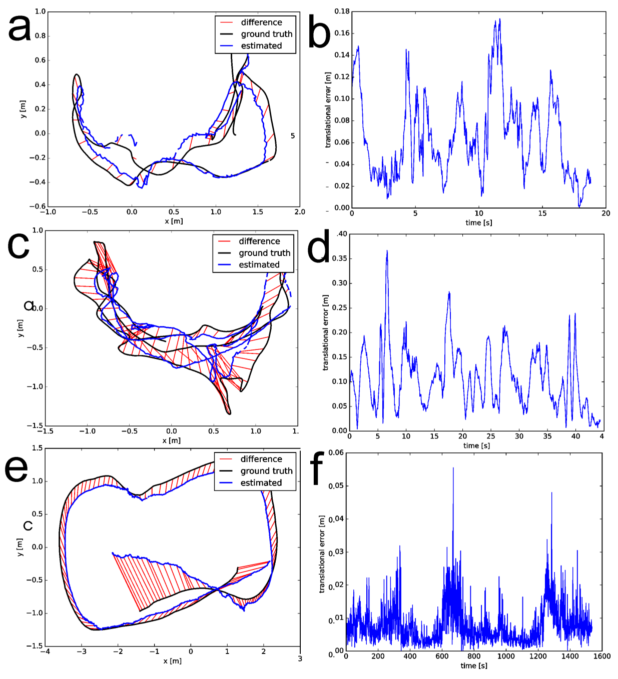

Fig. 9. Results of the on-line optimization experiments on three sequences: *fr1_desk* (a,b), *fr1_room* (c,d), and *putkk_Dataset1_Kin1* (e,f)

Although the AKAZE threshold was changed on-line by the PSO procedure, the final ATE RMSE and RPE RMSE results were in general not improved as compared to the best off-line results. Only for the *putkk_Dataset1_Kin1* sequence the results improved slightly. This is the sequence with a relatively smooth camera motion and constant lighting conditions, which suggests that the population-based optimization method is too slow to adapt on-line to sudden changes in the incoming data.

6. Conclusion

We demonstrated how simple to implement population-based optimization methods can be applied to the task of finding best parameters for a RGB-D visual odometry system. The proposed procedure can replace the exhaustive search that is sometimes used to find parameters that guarantee best results on the popular benchmark sequences. Although its main advantage is the improved speed (hours instead of several days), we have shown that the computed parameters generalize to other sequences, as long as they involve similar environments and camera dynamics.

We demonstrated also that the parameters of RANSAC, which is widely used in VO and SLAM are of great importance to provide a proper trade-off between the number of point features processed for frame-to-frame transformations and the matching accuracy.

We presented also a novel approach for on-line optimization of selected VO parameters that are expected to change along with the changes in the observed environment. Whereas the preliminary results of on-line optimization show only a small improvement, there is still room for further research with this approach using faster parameter search methods. The on-line performance could be also improved by a massively parallel implementation of the standard PSO algorithm on a GPU, but we do not consider this a feasible approach for VO on-board a mobile robot due to the energy consumption constraints.

AUTHORS

Aleksander Kostusiak* – Poznań University of Technology, Institute of Control, Robotics and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: aleksander.kostusiak@doctorate.put.poznan.pl.

Piotr Skrzypczyński – Poznań University of Technology, Institute of Control, Robotics and Information Engineering, ul. Piotrowo 3A, 60-965 Poznań, Poland, e-mail: piotr.skrzypczynski@put.poznan.pl.

*Corresponding author

REFERENCES

[1] P. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: *Procedings of the British Machine Vision Conference 2013*, Bristol, 2013, 13.1–13.11, 10.5244/C.27.13.

[2] M. Annunziato and S. Pizzuti, "Adaptive parameterization of evolutionary algorithms driven by reproduction and competition". In: *Proceedings of the European Symposium on Intelligent Techniques (ESIT 2000)*, 2000, 246–256.

[3] J. Arabas, *Lectures in evolutionary algorithms (in Polish: Wykłady z algorytmów ewolucyjnych)*, WNT: Warsaw, 2001.

[4] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments", *The International Journal of Robotics Research*, vol. 31, no. 11, 2012, 1320–1343, 10.1177/0278364912455256.

[5] D. Belter, M. Nowicki, and P. Skrzypczyński, "On the Performance of Pose-Based RGB-D Visual Navigation Systems". In: D. Cremers, I. Reid, H. Saito, and M.-H. Yang, eds., *Computer Vision – ACCV 2014*, 2015, 407–423, 10.1007/978-3-319-16808-1_28.

[6] D. Belter and P. Skrzypczyński, "Population-based Methods for Identification and Optimization of a Walking Robot Model". In: K. R. Kozłowski, ed., *Robot Motion and Control 2009*, 2009, 185–195, 10.1007/978-1-84882-985-5_18.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory". In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, 39–43, 10.1109/MHS.1995.494215.

[8] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems", *Expert Systems with Applications*, vol. 42, no. 5, 2015, 2670–2679, 10.1016/j.eswa.2014.11.009.

[9] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system". In: *2012 IEEE International Conference on Robotics and Automation*, 2012, 1691–1696, 10.1109/ICRA.2012.6225199.

[10] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D Mapping With an RGB-D Camera", *IEEE Transactions on Robotics*, vol. 30, no. 1, 2014, 177–187, 10.1109/TRO.2013.2279412.

[11] A. Engelbrecht, "Particle swarm optimization: Velocity initialization". In: *2012 IEEE Congress on Evolutionary Computation*, 2012, 1–8, 10.1109/CEC.2012.6256112.

[12] J. M. Falquez, M. Kasper, and G. Sibley, "Inertial aided dense & semi-dense methods for robust direct visual odometry". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, 3601–3607, 10.1109/IROS.2016.7759530.

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, vol. 24, no. 6, 1981, 381–395, 10.1145/358669.358692.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.

[15] W. Kabsch, "A solution for the best rotation to relate two sets of vectors", *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, vol. 32, no. 5, 1976, 922–923, 10.1107/S0567739476001873.

[16] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras". In: *2013 IEEE International Conference on*

Robotics and Automation, 2013, 3748–3754, 10.1109/ICRA.2013.6631104.

[17] A. Kostusiak and P. Skrzypczyński, "Population-based methods for optimization of parameters in a rgb-d visual odometry system (in Polish: Optymalizacja parametrów systemu odometrii wizyjnej rgb-d metodami populacyjnymi)", *Prace Naukowe Politechniki Warszawskiej. Elektronika*, vol. 196, 2018, 461–470.

[18] A. Kostusiak, "The Comparison of Keypoint Detectors and Descriptors for Registration of RGB-D Data". In: R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds., *Challenges in Automation, Robotics and Measurement Techniques*, 2016, 609–622, 10.1007/978-3-319-29357-8_53.

[19] A. Kostusiak, "Frame-to-Frame Visual Odometry: The Importance of Local Transformations". In: M. Kurzynski, M. Wozniak, and R. Burduk, eds., *Proceedings of the 10th International Conference on Computer Recognition Systems CORES 2017*, 2018, 357–366, 10.1007/978-3-319-59162-9_37.

[20] A. Kostusiak, M. Nowicki, and P. Skrzypczyński, "On the Application of RGB-D SLAM Systems for Practical Localization of the Mobile Robots", *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 11, no. 2, 2017, 57–66, 10.14313/JAMRIS_2-2017/17.

[21] M. Kraft, M. Nowicki, A. Schmidt, M. Fularz, and P. Skrzypczyński, "Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect", *Machine Vision and Applications*, vol. 28, no. 1, 2017, 61–74, 10.1007/s00138-016-0802-6.

[22] S.-K. Oh, H.-J. Jang, and W. Pedrycz, "A comparative experimental study of type-1/type-2 fuzzy cascade controller based on genetic algorithms and particle swarm optimization", *Expert Systems with Applications*, vol. 38, no. 9, 2011, 11217–11229, 10.1016/j.eswa.2011.02.169.

[23] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I the First 30 Years and Fundamentals", *IEEE Robotics Automation Magazine*, vol. 18, no. 4, 2011, 80–92, 10.1109/MRA.2011.943233.

[24] P. Skrzypczyński, "Mobile Robot Localization: Where We Are and What Are the Challenges?". In: R. Szewczyk, C. Zieliński, and M. Kaliczyńska, eds., *Automation 2017*, 2017, 249–267, 10.1007/978-3-319-54042-9_23.

[25] J. Sturm, W. Burgard, and D. A. Cremers, "Evaluating Egomotion and Structure-from-Motion Approaches Using the TUM RGB-D Benchmark". In: *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, 2012.

[26] J. Wietrzykowski and P. Skrzypczyński, "PlaneLoc: Probabilistic global localization in 3-D using local planar features", *Robotics and Autonomous Systems*, vol. 113, 2019, 160–173, 10.1016/j.robot.2019.01.008.

[27] Z. Xinchao, "A perturbed particle swarm algorithm for numerical optimization", *Applied Soft Computing*, vol. 10, no. 1, 2010, 119–124, 10.1016/j.asoc.2009.06.010.