Swarm-supported Outdoor Localization with Sparse Visual Data

Marcel Kronfeld* Christian Weiss* Andreas Zell*
*Department of Computer Science, University of Tübingen, Tübingen, Germany

***Abstract*— The localization of mobile systems with video data is a challenging field in robotic vision research. Apart from artificial environmental support technologies like GPS localization, a self-sufficient visual system is desirable. In this work, we introduce a new heuristic approach to outdoor localization in a scenario where no odometry readings are available. In an earlier work, we employed SIFT features and a common particle filter method in the scenario. A modification of Particle Swarm Optimization, a popular optimization technique especially in dynamically changing environments, is developed and fit to the localization problem, including self-adaptive mechanisms. The new method obtains similar or better localization results in our experiments, while requiring a fraction of SIFT comparisons of the standard method, indicating an all-over speed-up by 25%.**

I. Introduction

Localization with mobile robots may be achieved through a number of ways. Besides environmentally installed support architectures like radio beacons or GPS, the usage of universal visual features is an appealing approach to increase independence and robustness of mobile systems. Cameras may serve as relatively small, cheap, and yet powerful sensors for various surroundings and deliver a large amount of data, which, as biological organisms show, are highly valuable for orientation in natural environments. For visual localization, widely used techniques consist in combining a feature-based image similarity measure with a nonlinear particle filter (PF).

The paper at hand takes a closer look at a scenario with sparse visual data and without odometry, and a typical PF localization approach using the *Scale Invariant Feature Transform* (SIFT) [15]. Video data can easily be produced, but are very memory-consuming if densely collected. Also, sparse visual data can be easier maintained and updated, e.g. by driving through the streets of a city environment and taking relatively few, linearly ordered pictures.

The task of visual localization consists in finding a matching location by visual features in a database containing the environmental map. SIFT produces relatively reliable local image features based on structural interest points and may be used to recognize objects or locations in visual images with some robustness under changing illumination and direction of view. Visual applications employ SIFT in an increasing number, but comparing SIFT features is an expensive operation and often a bottleneck when the database is large. Some researchers compare a trial image to all images in the database and keep the database slim using pruning methods [5, 4]. Another approach uses a particle filter to concentrate the comparisons on a smaller subset of the features in the database, for which a match is expected with high probability [22]. Database tuning can be done offline, while the latter method may still reduce the number of online SIFT comparisons.

Localization with SIFT is also addressed e.g. in [2] for indoor and in [16] for outdoor environments. Different approaches for outdoor localization mainly use other types of local features like specialized histograms [6] or employ global features, e.g. PCA-based [11] or *Integral Invariant* features [23]. In [18], Tamimi presents a wide overview of visual features for robot localization.

Particle filters have been used for robot localization for some time [19], and numerous instances have been proposed [10], mainly differing in the resampling strategy. For visual tracking, Soto employed a PF with adaptive particle count [17], while Zhou et al. presented an approach in which motion velocities and noise level are adapted dynamically [24]. The *Kernel Particle Filter* (KPF) has been proposed by Chang et al. in [7] and lately been successfully combined with a heuristic called *Genetic Evolution* by Wang et al. [21]. A number of evolutionary extensions to the more general Markov chain Monte Carlo method are presented in [9]. Heuristically augmented particle filters have also been proposed by Treptow [20] and shown to be effective in reducing the particle count in real-time object tracking.

Considering swarm methods, the original Particle Swarm Optimization method (PSO) was proposed by Kennedy and Eberhart [12] and shown to be a valuable technique in different areas, of which optimization in dynamic environment is most related to our work, see e.g. [13]. Random perturbation within particle swarms has been proposed by Liu et al. [14].

The rest of the paper is organized as follows: Section II gives a short introduction to particle filters and points out some of their properties with respect to localization. In Section III, the basic Particle Swarm Optimization method is explained, and Section IV suggests a formulation of the localization problem in the context of optimization. Section V details our adaptations to PSO for the localization approach, also called Dynamic PSO (DPSO). Section VI introduces the experimental setup, the results are presented in Section VII. We finally conclude with Section VIII.

II. Particle Filter-based Localization

Particle filters essentially represent the probability density function (pdf) of the estimated system state with a set of so-called “particles”, each one encoding a single possible state. The particles are iteratively propagated using control input

(*motion model*) and weighted based on new measurements (*measurement model*). The weighted sum of the particles represents the estimated state. In visual localization, a particle represents a hypothesis on the system's position (and possibly its orientation) and is associated with a training image. The pdf then estimates the pose in the environment at a time. The estimation is improved iteratively by incorporating new test images and weighing the importance of the particles by the similarity to the test image.

Theoretically, if the number of particles is very large, the particle filter estimate approaches the optimal Bayesian state estimate [1], which is optimal with respect to the system models. In practice, however, the number of particles is restricted by computational effort, and thus often only relatively few particles can be used, a fact from which some problems arise.

As the variance of the importance weights can only increase over time [8], it is unavoidable that after some iterations most of the particles grow "impossible" in that their importance weights tend to zero, and much of the computational time is spent propagating highly improbable states. This may be avoided by using an importance resampling step, in which the particles are drawn anew from the currently estimated pdf at each iteration to keep particles within smaller regions of reasonable probability. Thereby, at iteration *t*, only those regions – and thus training images – are regarded which have a high probability given the information of $t-1$ earlier test images and the system models.

This diversity loss, however, abets localization failure in scenarios where jumps in the state space may occur – the so-called *kidnapped-robot problem*. To counter this, a common technique is to reinitialize a small ratio of particles randomly in each turn to keep the state space thinly covered (*random injection*). Other approaches try to detect kidnapping situations and handle them in a specialized way, facing the problems of possible misdetection and applying appropriate recovery.

In the scenario we are looking at, jumps in the state space are – to some extent – part of the underlying method, namely visual localization with sparse visual images of the environment. Also, no odometry information has been made available, which typically poses problems for particle filter approaches. Large numbers of particles are required to cover the state space and allow for good localization, while comparing visual images usually requires expensive computational operations, so that a reduction of the particle count is essential for quick localization. Then again, using visual features allows for making some helpful assumptions, such as their relatively high information density, which we try to exploit by interpreting localization as a dynamic optimization problem.

III. Particle Swarms

One branch of heuristic methods called Particle Swarm Optimization (PSO) is especially useful in dynamically changing domains [13, 3]. PSO takes as basic idea the flocking behaviour of animals like birds and searches for the solution using a population of potential solutions, called "particles" or "individuals". In a generational loop similar to Evolutionary Optimization, the individuals are iteratively updated using problem-specific knowledge to evaluate their current positions, resulting in a "fitness" value. Each individual I has a position $\vec{x}(t)$ and is assigned a travel velocity $\vec{v}(t)$. The individuals are arranged in a logical topology, by which for each I a neighborhood N_I of other individuals is defined. For the iteration at time t, the velocity vector of an individual is then attracted to the best location $\vec{p}^h$ in the individual's history $H_I = \bigcup_{t'=0}^{t} \{\vec{x}(t')\}$ on the one hand, and to the best location $\vec{p}^n$ found by its neighbors in N_I on the other hand, see Eqs. 1 and 2, defined by components of $\vec{x}$ and $\vec{v}$. The parameters ϕ_1 and ϕ_2 control the impact of the two attractors $\vec{p}^h$ and $\vec{p}^n$, while r_1 and r_2 are uniform random samples within the interval $[0,1]$ used as stochastic components. The factor ω is called *inertia* and controls the impact of the past velocity at a time. For this work, we use a simple star topology as neighborhood relation, implying that all particles are neighbors and are attracted to the currently "fittest" position known to the population.

$$v_i(t+1) = \omega v_i(t) + \phi_1 r_1 (p_i^h - x_i) + \phi_2 r_2 (p_i^n - x_i) \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

In typical PSO implementations, the velocity vector is limited to a maximum velocity v_{max} by ensuring that $\|\overrightarrow{v}\| \leq v_{max}$. A number of extensions have been introduced to improve PSO for dynamic optimization problems, of which we use the following:

- Invalidation of $\vec{p}^h$ at changes of the environment. When expecting the "worst case", which is continuous movement, $\vec{p}^h$ is replaced by a random perturbation term;
- "Quantum particles", similar to *random injection* used with PF. Quantum particles have no speed but are stochastically distributed over an area around the last position estimate within which movements are typically expected;
- "High-energy particles", which are allowed higher speeds than usual ones and have the same properties otherwise.

A heuristic approach reduces the system complexity, and PSO typically needs only few particles for good results, which is what we want to achieve for visual localization.

IV. Visual Localization as Optimization Problem

Presuming a training set of images corresponding to known positions as given world map, the goal of localization is to deduce a position estimate based on test images taken online with respect to the training set. Our formulation of visual localization as optimization problem is in analogy to the resampling criterion of a PF: If a particle has a high probability of fitting the measurement, it also has a high value or "fitness" in the sense of optimization. The target function value of a particle with position $\vec{x}$ at time t may be expressed by

$$f(\vec{x}, t) = m(pic_{tr}(\vec{x}), pic(t)) \cdot pen(d(\vec{x}, pic(\vec{x}))), \quad (3)$$

where m is the SIFT comparison function [15], comparing the nearest training image $pic_{tr}(\vec{x})$ corresponding to the particle to the current test image $pic(t)$. m delivers a value in $[0,1]$ indicating image similarity by calculating the ratio

of *single feature matches* to all possible matches. A single feature of image a is said to *match* a feature of image b if its euclidian distance to the closest neighbor in b divided by the distance to the second closest in b is above a threshold δ, typically $\delta \approx 0.6 - 0.8$. The penalty function pen reduces the fitness for particles far away from the training data, because localization is feasible only where there is training information available. This is done similarly to [22] in terms of a gaussian function.

The problem of tracking a position now corresponds to a dynamic optimization problem, aiming for finding the optimum $\vec{x}^*(t')$ of f at time t' and then following it while ensuring a plausible path. A dynamic optimization method therefore needs to predict potential future optima, while keeping them related to the current state. The velocity components assigned to PSO particles may be interpreted as multiple motion models with respect to a PF and allow for just that. The actual position estimate can be deduced from the swarm by calculating the weighted swarm center or by just picking out the *k* best particles and calculate their center.

V. Adapting PSO for Visual Localization

In scenarios with sparse visual data, the particle filter approach is not effective, mainly because it requires a relatively large number of particles compared to the number of available images. As PSO is known to perform well in dynamic environments, we adapted PSO to this class of localization problems. Our dataset did not contain odometry, so the velocity of particles could also be used to estimate the robot's speed.

The fitness of an individual at time *t* depends on the robot's view at that time and is calculated as the SIFT similarity between the current test image and the training image closest to the individual's position, modified by a penalty function if the individual is far away from the training image position (Eq. 3). The resampling is replaced by the PSO formula, adapted to the dynamic localization case in the following way:

$$v_i(t+1) = \omega v_i(t) + \phi_0 r_0 \delta_i v_0 + \phi_2 r_2 (p_i^n(t) - x_i(t)) \quad (4)$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (5)$$

The $\vec{p}^h$-component is herein replaced by a random term, because we expect a continuously changing environment, where historically good positions lose their relevance quickly. ϕ_0 is the weight of the random perturbation, the additional parameters δ_i and v_0 stand for the range of axis *i* and the maximum randomized velocity of a particle, respectively. The velocity v_0 is expressed relative to the range and also suits as scaling factor to keep ϕ_0 in similar dimension as ϕ_1 and ϕ_2. Effectively, the main attractor $\vec{p}^n$ is thereby turned into an area of attraction around $\vec{p}^n$ of dimensions $\rho_i = \frac{\phi_0}{\phi_2}\delta_i v_0$.

For the fraction $\widehat{q_r}$ of quantum particles, the update takes the following form:

$$x_i(t+1) = p_i^n(t) + \delta_i N(0, \widehat{q_d}). \quad (6)$$

The parameter $\widehat{q_d}$ defines the standard deviation of the quantum particles around $\vec{p}^n$. The formula is similar to the default gaussian motion model employed with the PF. The quantum ratio $\widehat{q_r}$ is set to 10% by default, $\widehat{q_d}$ to 0.15. The inertia is usually set < 1 to allow for convergence. For dynamic tracking, however, it needs to be high to stress correlation of movement, so we set it to 0.99. Standard settings for ϕ_0 and ϕ_2 were $\phi_0 = 0.3$ and $\phi_2 = 0.6$. The trade-off between the ϕ-values in Eq. 4 remains of similar importance as for standard particles, now trading between random exploration and local exploitation. Random perturbation is necessary for particle diversity, but reduces the overall tracking quality if too dominant. A fraction of particles $\widehat{h_r} = 10\%$ is allowed a velocity three times v_0, supporting quick optimum tracking. We simply use the best individual at a time as position estimate, following the typical methodology in heuristic optimization. The full algorithm is termed "Dynamic PSO" (DPSO) for the rest of this paper.

Self-adaptive parameters

We introduce two self-adaptive mechanisms, one of which dynamically adapts v_0 by calculating the speed v_{swarm} of the swarm's center of mass and holding the relation $v_0 \approx 2v_{swarm}$. This enables the method to react to speed changes while providing robust tracking at any speed. When tracking the location, v_{swarm} also gives a good estimate of the robot's speed in absence of or in addition to odometry.

The SIFT features offer robust image similarity information in outdoor areas, yet some situations still are ambiguous. To counter this and to ease the handling of the kidnapped-robot problem, we also include a mechanism to dynamically adapt the swarm diversity. We therefore exploit the property of SIFT delivering a quasi-absolute measure of similarity between images. If the best SIFT match of the best position guess is still bad, e.g. matching less than 5% of the features, it may be an ambiguous position or the localizer lost track of the real position. If this happens for several iterations in a row, we start a recovery phase and boost particle diversity by increasing ϕ_0, v_0 and the quantum ratio $\widehat{q_r}$ up to predefined maximum values. As soon as the best matches increase in quality again, the recovery phase ends and the parameters are decreased to the initial values. Experiments have shown that adapting v_0 improves tracking and adapting diversity improves robustness plus it solves the kidnapped-robot situation, cf. Section VII-A.

VI. Experimental Scenario

In the experiments in [22], we used images collected by our RWI ATRV-JR outdoor robot, Arthur (Fig. 1). We took one 320×240 pixel grayscale image per second with the left camera of the Videre Design SVS stereo camera system mounted on top of the robot. As we used a constant velocity of about 0.6 m/s, the positions of subsequent images are about 0.6 meters away from each other. The robot is also equipped with a differential GPS (DGPS) system, which we used to get ground truth data for the position of each image. Under ideal conditions, the accuracy of the DGPS is below 0.5 m. However, due to occlusion by trees and buildings, the GPS path sometimes significantly deviated from the real position or contained gaps. As we know that we moved the robot on

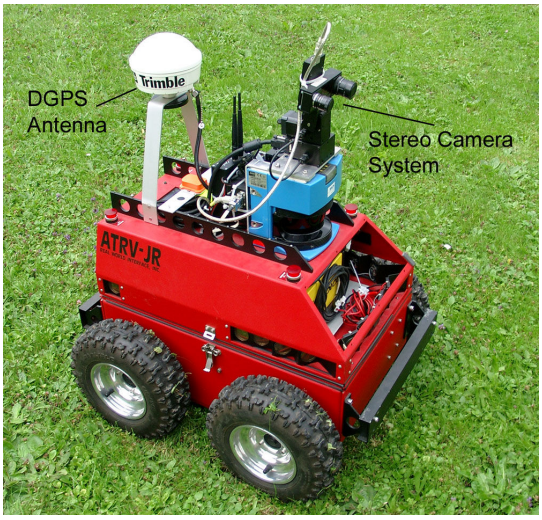

Fig. 1. Arthur, an RWI ATRV-JR outdoor robot.

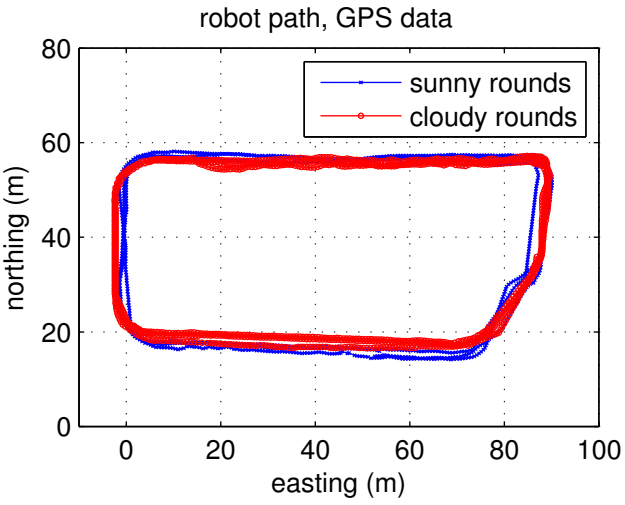

Fig. 2. GPS round data.

Fig. 3. Example images of the datasets, sunny (left) and cloudy (right).

a smooth trajectory, we eliminated some wrong GPS values manually. As we also used a constant velocity, we closed gaps by linearly interpolating between the positions before and after the gap. In our experiments, we used two different datasets, each consisting of three rounds around a big building. One round is 260 m long and contains about 400 images. The first three rounds were collected under sunny conditions. However, there are some short sections (about 5 to 10 s long) during which the sun was covered. Six weeks later, we collected the other three rounds on a cloudy day. The images contain artifical objects like buildings, streets and cars as well as some vegetation. Additionally, there are dynamic objects like cars and people passing by. We also traversed a parking lot, where different cars were parked on the two days. As in [22], we reduced the number of SIFT features by comparing each image to the two neighboring images in the series beforehand and discarding the "noisy" features which could not be discovered in either of the direct neighbors. This speeds up the SIFT comparison drastically, as about 50% to 80% of the features are left out, while not affecting later localization performance.

From the two datasets "sunny" and "cloudy", three kinds of experiments have been carried out. Using one round as environmental map and treating another as online data, we tested sunny vs. sunny, cloudy vs. cloudy, and sunny vs. cloudy. We did not test a round against itself, so there are six cases for the sunny and cloudy only experiments and nine for the sunny vs. cloudy experiment. We calculated the mean error for the three experiments distinctly. For each case, we repeated the localization run n times, where n is the number of test images. For each of these runs, we used a different test image as initial image for the localization.

VII. Results

A. Parameter Comparison

To demonstrate the effects of the settings for DPSO, we contrast localization rounds for a single localization case, namely localizing sunny (round 1) vs. cloudy (round 1), which is one of the difficult cases for the PF. Where not stated otherwise, DPSO is used with a swarm size of 80 particles. We varied the simulated speed of the mobile system, see Tab. I. Here, the standard speed 1 corresponds to the original data collection speed of about 0.6 m/s. For speeds k times the normal we use every k-th image for localization only, resulting in the simulation hurrying around the loop at higher speed. In analogy, a speed of $\frac{1}{2}$ or $\frac{1}{4}$ corresponds to localizing against each image twice or four times consecutively. In the non-adaptive version in Tab. I, we set the maximum speed parameter manually to roughly fit the standard speed case ($v_0 = 0.015$). The fact that repeating the same image several

Sim. Speed	1/4	1/2	1	2	3	4
Non-adaptive	2.48	2.51	2.54	2.73	5.89	9.59
Adaptive	2.57	2.59	2.50	2.68	3.20	3.75

TABLE I

AVERAGE ERROR (M) WHEN VARYING THE SIMULATED ROBOT'S SPEED

Condition	Standard case avg. err. (m)	Kidnapped case avg. err. (m)
Non-adaptive	2.56 $\pm$ 0.71	11.55 $\pm$ 7.23
Adaptive	2.50 $\pm$ 0.35	6.10 $\pm$ 2.34

TABLE II

COMPARING ADAPTIVE DPSO IN STANDARD AND KIDNAPPED CASE

Method	DPSO-40	DPSO-60	DPSO-80	DPSO-100	DPSO-120	PF-100	PF-300
Avg.err. (m)	2.84	2.60	2.54	2.50	2.46	3.95	3.39
Avg.comp./image	17.9	22.3	25.9	29.1	32.0	40.8	62.4

TABLE III
VARYING THE NUMBER OF PARTICLES FOR DPSO

Experiment	PF-100		PF-300		DPSO-80	
	Avg. err. (m)	Avg. comp. per img.	Avg. err. (m)	Avg. comp. per img.	Avg. err. (m)	Avg. comp. per img.
Sunny vs. sunny	3.1649±0.8907	40.0	2.1510 ± 0.2885	60.8	1.9862± 0.3612	21.4
Cloudy vs. cloudy	3.4555±1.2783	36.5	2.0556 ± 0.5561	55.3	1.4698 ± 0.3507	20.5
Sunny vs. cloudy	3.9260±0.6635	40.4	3.2723 ± 0.2749	60.9	2.7651 ± 0.4036	22.3

TABLE IV
COMPARISON OF PARTICLE FILTER TO DPSO

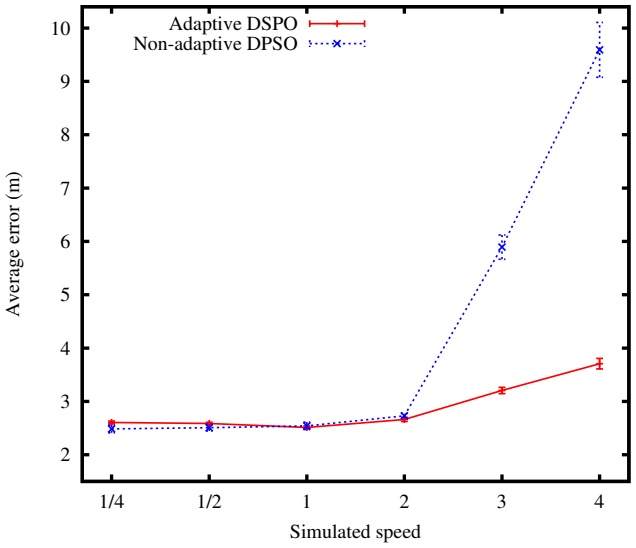

Fig. 4. Localization error at different simulated speeds

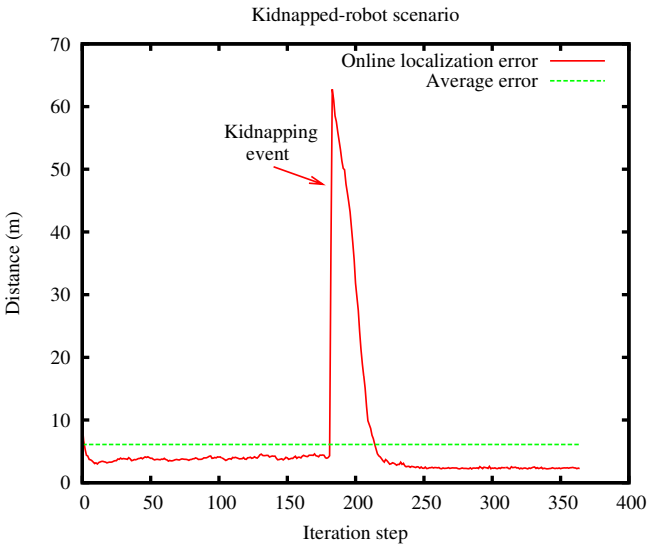

Fig. 5. DPSO in the kidnapped-robot scenario

times worsens the localization performance for the adaptive version seems counterintuitive. It can be explained when thinking about ambiguous views: Their interference is the higher the more often they are presented. The non-adaptive variant is less affected by this because it does not converge as closely. For higher speeds, however, the non-adaptive method clearly fails without manual tuning of the speed limit, cf. Fig. 4.

In Tab. II, the self-adaptive diversity mechanism comes into play when a kidnapped-robot scenario is simulated. To do this, the virtual position in each simulated round is set to the opposite of the round by adding *n/2* to the current test image index modulo *n*, where *n* is the number of test images. This means that the localization method is forced to jump to the opposite side of the playground after converging for half of the run. The simulated kidnapping takes place at iteration 182, which is $\lfloor \frac{n}{2} \rfloor$ of $n = 365$ and causes an abrupt localization error of about 63 m at that instant. Of course the kidnapping reduces performance, but the adaptive method is clearly able to find and retrack the position. The average online localization error of the experiment is plotted in Fig. 5. The averaged localization error of all *n* runs is also shown.

Tab. III shows results for different numbers of particles, demonstrating that even low population sizes are able to perform well. The largest population tested here still requires much less of the number of SIFT comparisons needed by the particle filter, cf. Section VII-B. Smaller populations, admittedly, have more problems recovering in a kidnapped-robot case. We therefore use a swarm size of 80 for the final comparison with PF as well.

B. Comparison to Particle Filter

In the final test runs for the comparison with a PF approach, we used the self-adaptive mechanisms and default settings described in Section V. As our dataset did not contain odometry, the particle filter was employed with a random gaussian motion model with a standard deviation of 4 m, while the sampling weight is calculated in analogy to Eq. 3.

From Tab. IV it can be seen that the DPSO approach with 80 particles is superior to both the PF using 100 and the PF using 300 particles. Concerning the localization error, PF-100 performs considerably worse than the other methods. The

average number of training images a test image is compared to is also significantly lower with DPSO. More precisely, it is decreased by about 35-45% compared to the PF-100 and by 60-65% compared to the PF-300 method. This can be directly projected onto the real system runtime, as SIFT comparison is the most expensive operation of the method.

The DPSO iteration with self-adaption can be done in two loops over the particle population and is therefore comparable to a PF with resampling, yet having a much smaller population. A SIFT comparison of the considered dataset took 0.015 s on average on our test system, a 2.4 GHz dual core AMD Opteron. An iteration of PF-300 therefore takes between 0.8 and 0.9 s on average and a reduction by 60% implies saving about 0.5 s per test image. With regard to [22], where SIFT comparisons took about 40% of the computation time using the PF-300 method for localization, an all-over speed-up by 25% can be expected.

VIII. Conclusion

In the work at hand, we have proposed a PSO-based method replacing a standard particle filter for localization with SIFT features on sparse outdoor visual data. As in standard PSO, the particles are attracted to the best global particle, allowing for fast convergence. The particles have a velocity component with high inertia, thus the dynamically changing position can be tracked without requiring an explicit motion model. This is a major advantage e.g. for the augmentation of black-box systems with independent visual trackers.

By taking advantage of the fact that SIFT provides a quasi-absolute measure of image similarity, a good guess can be made whether the position has been lost. In that case, particle diversity is boosted until a good position estimate is rediscovered. In addition to that, dynamic speed adaptation makes the system robust with regard to manual parameter settings and the robot's velocity.

Our experiments where based on visual images ordered linearly within an urban outdoor environment using GPS ground truth. Similar data can be obtained without much effort in large scale, whereby the plausibility of the GPS annotations must be verified. Given that, we are confident that the DPSO for localization scales well to large datasets. However, in highly ambiguous or dynamic environments like e.g. forests, we expect that the swarm approach loses accuracy compared to a particle filter, at least while employing the rather greedy star topology for the swarm. We intend to test a multi-swarm approach to counter this effect in ambiguous scenarios.

The use of visual sensory and robust features is the basis for our localization method. The exact kind of features, however, needs not to be predefined. SIFT is a popular choice and serves well for comparisons to further approaches, e.g. by using iterative SIFT, additional geometric constraints or hybrid feature sets. We plan to analyze the swarm-supported localization with hybrid features and in larger scenarios in the near future.

References

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174 – 188, 2002.

[2] M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single camera. In *Proceedings of European Robotics Symposium (EUROS-06)*, volume 22, pages 143–157, Palermo, Italy, March 2006.

[3] T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In *EvoWorkshops*, pages 489–500, 2004.

[4] O. Booij, B. Terwijn, Z. Zivkovic, and B. Kröse. Navigation using an appearance based topological map. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3927–3932, 2007.

[5] O. Booij, Z. Zivkovic, and B. Kröse. Sparse appearance based modeling for robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1510–1515, 2006.

[6] D. M. Bradley, R. Patel, N. Vandapel, and S. M. Thayer. Real-time image-based topological localization in large outdoor environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3062 – 3069, Edmonton, Canada, 2005.

[7] C. Chang and R. Ansari. Kernel particle filter for visual tracking. *IEEE Signal Processing Letters*, 12(3):242–245, March 2005.

[8] A. Doucet. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

[9] M. M. Drugan and D. Thierens. Evolutionary Markov chain Monte Carlo. Technical report, Institute of Information and Computing Sciences, Utrecht University, 2003.

[10] S. Godsill and T. Clapp. Improvement strategies for monte carlo particle filters. In A. Doucet, N. De Freitas and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag, 2000.

[11] M. Jogan, M. Artac, D. Skocaj, and A. Leonardis. A framework for robust and incremental self-localization. In *Proc. of the 3rd International Conference on Computer Vision Systems (ICVS)*, pages 460–469, Graz, Austria, 2003.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE Int. Conf. on Neural Networks*, Perth, Australia, 1995.

[13] X. Li, J. Branke, and T. Blackwell. Particle swarm with speciation and adaptation in a dynamic environment. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 51–58, New York, NY, USA, 2006. ACM Press.

[14] H. Liu and A. Abraham. Fuzzy adaptive turbulent particle swarm optimization. In *HIS '05: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, pages 445–450, Washington, DC, USA, 2005. IEEE Computer Society.

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[16] S. Se, T. Barfoot, and P. Jasiobedzki. Visual motion estimation and terrain modelling for planetary rovers. In *Proceedings of the International Symposium on Artificial Intelligence for Robotics and Automation in Space (iSAIRAS)*, Munich, Germany, 2005.

[17] A. Soto. Self adaptive particle filter. In *International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005.

[18] H. Tamimi. *Vision-based Features for Mobile Robot Localization*. PhD thesis, University of Tübingen, Tübingen, Germany, 2006.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[20] A. Treptow. *Optimization Techniques for Real-Time Visual Object Detection and Tracking*. PhD thesis, University of Tübingen, Tübingen, Germany, 2007.

[21] Q. Wang, J. Liu, and Z. Wu. Object tracking using genetic evolution based kernel particle filter. In *Combinatorial Image Analysis: 11th International Workshop, IWCIA 2006, Proceedings*, pages 466–473, 2006.

[22] C. Weiss, A. Masselli, H. Tamimi, and A. Zell. Fast outdoor robot localization using integral invariants. In *Proc. of the 5th International Conference on Computer Vision Systems (ICVS)*, Bielefeld, Germany, March 21 - 24 2007.

[23] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, 21(2):208–216, 2005.

[24] S. Zhou, R. Chellappa, and B. Moghaddam. Appearance tracking using adaptive models in a particle filter. In *Asian Conference on Computer Vision*, 2004.