Swarm-supported Outdoor Localization with Sparse Visual Data

Marcel Kronfeld [*], Christian Weiss, Andreas Zell

University of Tübingen, Department of Computer Science, Tübingen, Germany

Abstract

The localization of mobile systems with video data is a challenging field in robotic vision research. Apart from support technologies like GPS, a self-sufficient visual system is desirable. We introduce a new heuristic approach to outdoor localization in a scenario with sparse visual data and without odometry readings. Localization is interpreted as an optimization problem, and a swarm-based optimization method is adapted and applied, remaining independent of the specific visual feature type. The new method obtains similar or better localization results in our experiments while requiring only two-thirds of the number of image comparisons, indicating an all-over speed-up by 25%.

Key words: Outdoor robotics, robot vision, visual localization, swarm intelligence, particle swarm optimization

1. Introduction

Localization with mobile robots may be achieved through a number of ways. Besides environmentally installed support architectures such as 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 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, where a typical PF localization approach was employed using an image similarity measure, the *Scale Invariant Feature Transform* (SIFT) [1]. Video data can easily be produced, but are extremely memory-consuming if densely collected. Also, sparse visual data can be maintained and updated easier, e.g., by driving through the streets of a city environment and taking 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. SIFT is a popular method for visual applications, 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 [2, 3]. 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 [4]. Database tuning can be done offline, while the latter method may still reduce the number of online image comparisons.

Besides typical proprioceptive odometry, vision can be used to obtain odometry estimates, e.g., from

[*] Corresponding author.
Email address: marcel.kronfeld@uni-tuebingen.de (Marcel Kronfeld).

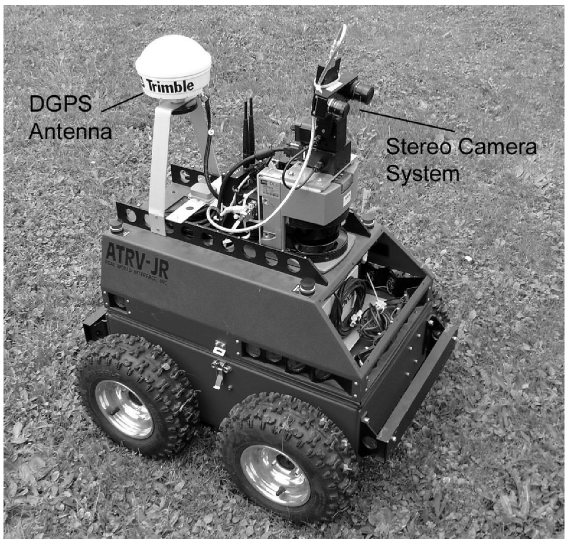


Fig. 1. RWI outdoor robot "Arthur".

feature tracking [5, 6] or optical flow [7]. Yet, these techniques are relatively expensive with respect to computation time and usually require a specific local visual feature type. Since, in large-scale scenarios, very diverse platforms may be employed, our approach remains independent of the specific feature detector. Therefore, we only presume an image similarity measure of a certain distinctness comparable to SIFT, and we do not employ optical flow.

A wide range of feature types have been used for visual robot localization. Besides global features, e.g., PCA-based [8] or *Integral Invariant* features [9], local features such as SIFT have been successfully employed in indoor [10, 11] and in outdoor environments, over longer time scales, and in spite of occlusion [12, 13, 14]. Since local approaches extract a diverse set of local descriptors from an image, they are usually more robust, but also more time-consuming than global feature methods. In [15], Tamimi presents a wide overview of visual features for robot localization.

Particle filters have been used for robot localization for some time [16], and numerous instances have been proposed [17, 18, 19]. For visual tracking, Soto employed a PF with adaptive particle count [20], while Zhou et al. presented an approach in which motion velocities and the noise level are adapted dynamically [21]. The *Kernel Particle Filter* (KPF) has been proposed by Chang et al. in [22] and lately been successfully combined with a heuristic called *Genetic Evolution* by Wang et al. [23]. A number of evolutionary extensions to the more general Markov chain Monte Carlo method are presented in [24]. Heuristically augmented particle filters have also been proposed by Treptow [25] 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 [26] 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., [27]. Random perturbation within particle swarms has been proposed by Liu et al. [28]. Vahdat et al. [29] employed PSO for indoor robot localization with laser sensors without considering the dynamic tracking phase.

The rest of the paper is organized as follows: Section 2 gives a short introduction to particle filters and points out some of their properties with respect to localization. In Sec. 3, the basic particle swarm optimization method is explained, and Sec. 4 suggests a formulation of the localization problem in the context of optimization. Section 5 details our adaptations to PSO for the localization approach, also called Dynamic PSO (DPSO). Section 6 introduces the experimental setup, the results are presented in Sec. 7, and we conclude with Sec. 8.

2. Particle Filter-based Localization

Particle filters essentially represent the probability density function (pdf) of the estimated system state by a set of "particles", each of which encodes a single possible state. The particles are iteratively propagated using control inputs (the *motion model*) and associated with *importance weights* representing their individual probability given new measurements (the *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 velocity) 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 reweighing the particles according to the similarity to new test images.

Theoretically, if the number of particles is very large, the particle filter estimate will approach the optimal Bayesian state estimate [30], which is optimal with respect to the system models. In practice, however, the number of particles is limited due to the computational costs, 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 [31], it is inevitable that after some iterations most of the particles grow "impossible" in that their importance weights tend to zero, and much of the computation 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. 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 – known as the *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. Large numbers of particles are required to cover the state space and allow for good localization, while comparing visual images usually requires computationally expensive operations, so a reduction of the particle count is essential for quick localization. Then again, using visual features allows for making some helpful assumptions and provides a high information density, which we exploit by interpreting localization as a dynamic optimization problem.

3. Particle Swarms

One branch of heuristic methods called Particle Swarm Optimization (PSO) is especially useful in dynamically changing domains [27, 32]. PSO takes as basic idea the flocking behaviour of 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 $\boldsymbol{x}(t)$ and is assigned a travel velocity $\boldsymbol{v}(t)$. The individuals are arranged in a logical topology, by which a neighborhood N_I of other individuals is defined for each I. For the iteration at time t, the velocity vector of an individual is then attracted to the best location $\boldsymbol{p}^h$ in the individual's history $H_I = \bigcup_{t'=0}^{t}\{\boldsymbol{x}(t')\}$ on the one hand, and to the best location $\boldsymbol{p}^n$ found by its neighbors in N_I on the other hand, see Eqs. 1 and 2, defined by components of $\boldsymbol{x}$ and $\boldsymbol{v}$. The parameters ϕ_1 and ϕ_2 control the impact of the attractors $\boldsymbol{p}^h$ and $\boldsymbol{p}^n$, 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. 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 in 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_0 by ensuring that $\|\boldsymbol{v}\| \leq v_0$. A number of extensions have been introduced to improve PSO for dynamic optimization problems, of which we use the following:

– Invalidation of $\boldsymbol{p}^h$ at changes of the environment. When expecting the hardest case, which is continuous movement, $\boldsymbol{p}^h$ 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 particles 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 desirable for visual localization.

4. Visual Localization as Optimization Problem

Let S be the set of all possible camera images. Presuming a training set $M \subset S$ of images corresponding to known positions as a 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 fitness value of a particle with position $\boldsymbol{x} \in X$ at time t may be expressed using a similarity measure of two arbitrary images, $m : S \times S \to [0, 1]$:

$$f_M(\boldsymbol{x}, t) = m(i_M(\boldsymbol{x}), s(t)) \cdot \zeta(\mathrm{dist}(\boldsymbol{x}, p_M(i_M(\boldsymbol{x})))) \tag{3}$$

Herein, m compares the nearest training image corresponding to the particle, $i_M : X \to M$, and the current test image $s(t) \in S$. $p_M : M \to X$ delivers the known position for an image which is part of the map. The penalty function $\zeta : \mathbb{R} \to \mathbb{R}$ 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 [4] in terms of a Gaussian function. The problem of tracking a position now corresponds to a dynamic optimization problem: find the optimum $\boldsymbol{x}^*$ of f_M at a time and follow it ensuring a plausible path.

In an approach with sparse visual outdoor data, a distinct similarity measure is desirable and can be implemented using SIFT [1]. The SIFT match function compares sets of local SIFT features of images A and B by calculating the ratio of *single feature matches* to all possible matches. A singe feature match for a feature $a \in A$ is detected by looking at the two closest features of a within image B, $b_{a,1}, b_{a,2} \in B$, by a distance measure d in feature space: $d(a, b_{a,1}) \leq d(a, b_{a,2})$ and $\forall b \in B \setminus \{b_{a,1}, b_{a,2}\} : d(a, b_{a,2}) \leq d(a, b)$. Feature a is said to *match* feature $b \in B$ if b is its closest neighbor $(b = b_{a,1})$ and the distance ratio with respect to the second closest neighbor $b_{a,2}$ is above a threshold δ, typically $\delta \in [0.6, 0.8]$. Thus, for the set of single feature matches of A in B, M_{AB}, it holds that

$$a \in M_{AB} \Leftrightarrow \frac{d(a, b_{a,1})}{d(a, b_{a,2})} < \delta.$$

Specifically, we use a more robust symmetric variant and define the set of symmetric feature matches M^*_{AB} for pairs (a, b) that fulfil $a = a_{b,1} \wedge b = b_{a,1}$, meaning that they are reciprocal nearest neighbors:

$$(a, b) \in M^*_{AB} \Leftrightarrow a \in M_{AB} \wedge b \in M_{BA}$$

The match function m can now be expressed as $m(A, B) = \frac{|M^*_{AB}|}{min(|A|,|B|)}$, where $|A|$ stands for the number of features extracted from image A, so $min(|A|, |B|)$ is the maximum size of M^*_{AB}. The scalar product, $d(a, b) = \langle a, b\rangle$, is used as the distance measure on feature vectors, since it is a metric for normed vectors and approximates the Euclidian distance for small angles $\angle(a, b)$. δ is set to 0.6.

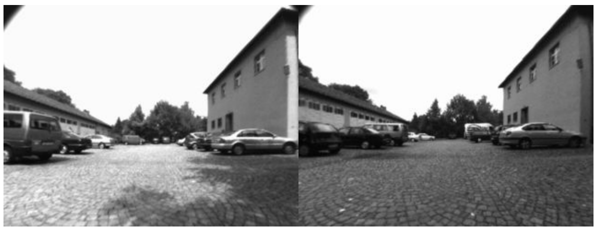


Fig. 2. Example images of the datasets, sunny (left) and cloudy (right).

Since the target function f depends on the current view and thus on the robot location, the value of a particle at a fixed position $\boldsymbol{x}$ changes substantially when the robot moves. The problem of tracking a position therefore corresponds to a dynamic optimization problem, aiming for finding the optimum $\boldsymbol{x}^*(t')$ of f at time t' and then following it while ensuring a plausible path. A dynamic optimization method 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. In optimization, however, the particle set is not assumed to correspond to the statistical solution distribution at a single instant, but rather every particle is seen as a possible solution in the light of the fitness function. Due to that and the assumption of a distinctive similarity measure on sparse visual data, we just pick out the best particle from the swarm as position estimate.

5. Adapting PSO to Visual Localization

In scenarios with sparse visual data, the particle filter approach is time-consuming, 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. The dataset did not contain odometry, so the velocity of particles could also be used to estimate the robot's speed.

Basic algorithm

The fitness of an individual at time t depends on the camera 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 position of the training image (Eq. 3). The resampling is replaced by the PSO formula, adapted to the dynamic localization case in the following way:

$$v_i(t+1) = \omega v_i(t) + \phi_0 r_0 \delta_i v_0 + \phi_2 r_2 (p_i^n(t) - x_i(t)), \quad (4)$$
$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (5)$$

Instead of the $\boldsymbol{p}^h$-component, a random term is used therein, because we expect a continuously changing environment where historically good positions quickly lose their relevance. 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 velocity of a particle, respectively. The maximum velocity v_0 is expressed relative to the range and also serves as a scaling factor to keep ϕ_0 in similar dimension as ϕ_1 and ϕ_2. Effectively, the main attractor $\boldsymbol{p}^n$ is thereby turned into an area of attraction around $\boldsymbol{p}^n$ 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 $\boldsymbol{p}^n$. The formula is similar to the standard 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, and a fraction of particles $\widehat{h_r} = 10\%$ is allowed a velocity three times v_0 to ease quick optimum tracking.

The inertia is usually set $\omega < 1$ to allow for convergence. For dynamic tracking, however, it needs to be large to stress correlation of movement, so we set it to 0.99. The best ϕ-settings in PSO for a problem class are often established by a parameter grid search. 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. We found $\phi_2 = 0.6$ to perform well, while the method is pretty robust towards settings of $\phi_0 \in [0.005, 1.5]$, where smaller values allow closer convergence but increase the danger of losing the track. We suggest $\phi_0 = 0.3$ as a default. 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_{sw} of the swarm's center of mass and holding the relation $v_0 \approx 2v_{sw}$. This is done by looking at the Exponential Moving Average (EMA, $\alpha = \frac{1}{8}$) [1] of v_{sw} and adapting v_0 by 10% after an iteration t if $\frac{EMA_{v_{sw}}(t)}{v_0}$ does not lie in [0.4,0.6]. This enables the method to react to speed changes while providing robust tracking at any speed. When tracking the location, v_{sw} also gives a good estimate of the robot's speed in absence of or in addition to odometry.

SIFT features offer robust image similarity information in outdoor areas, still some situations are ambiguous. To counter this and to cope with the kidnapped-robot problem, we include a mechanism to dynamically adapt swarm diversity. To do this, we assume the image similarity function to deliver a quasi-absolute measure of similarity. For local feature detectors such as SIFT, this can be delegated to the descriptor count: If the best match in the particle set is still bad, e.g., matching less than 5% of local features, it may be an ambiguous position or the localizer lost the real position. If this happens for several iterations in a row, we start a recovery phase and boost particle diversity by increasing v_0, $\widehat{q_r}$ and decreasing ϕ_2 towards predefined limit values. ϕ_2 may be reduced close to zero ($\phi_{2,min} = 0.01$), reducing the attraction of the best matching particle, which is, in this case, still bad. The velocity and $\widehat{q_r}$ should be allowed values large enough to quickly start exploring the problem space and overcome possible jumps due to kidnapping. Without further tuning, we preset $\widehat{q}_{r,max}$ to 20% and allow $v_{0,max} = 0.1$, meaning that in case of position loss, particles may cross the search space in about 10 iterations. As soon as the best matches increase in quality again, the recovery phase ends and the parameters return to their initial values. Preliminary experiments showed that the adaption of v_0 improves tracking and the adaption of diversity improves robustness plus it solves the kidnapped-robot situation, see Sec. 7.1.

[1] For a sequence of values $(Y_0, Y_1, Y_2, \dots)$: $\mathrm{EMA}_Y(0) = Y_0$, $\mathrm{EMA}_Y(k+1) = (1-\alpha) \cdot \mathrm{EMA}_Y(k) + \alpha Y_k,\ k \in \mathbb{N}_0$

6. Experimental Scenario

In the experiments in [4], 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 $0.6\,\frac{m}{s}$, the positions of subsequent images are about $0.6\,m$ away from each other. The robot is also equipped with a GPS system, which we used to get ground truth data for the position of each image. Under ideal conditions, the accuracy of the GPS is below $0.5\,m$. However, due to occlusions and reflections 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 a smooth trajectory, we eliminated some wrong GPS values as outliers. As we also used a constant velocity, we closed gaps by linearly interpolating between the positions before and after the gap.

We recorded two different datasets, each consisting of three rounds around our institute building. One round is $260\,m$ long and contains about 360 to 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 buildings, streets, as well as some vegetation. Additionally, there are dynamic objects, namely cars and people passing by. We also traversed a parking lot, where different cars were parked on the two days. Exemplary images from both sets are shown in Fig. 2, whereas the layout of the rounds is plotted in Fig. 3, in which, for clarity, only every 20^{th} image is marked. As in [4], 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.

With the two datasets "sunny" and "cloudy", three kinds of experiments were conducted. Using one round as environmental map and treating the second 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

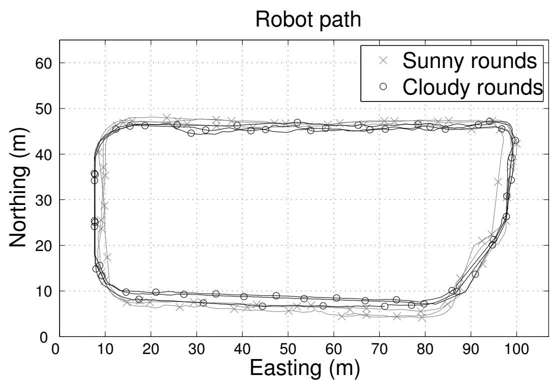


Fig. 3. GPS round data.

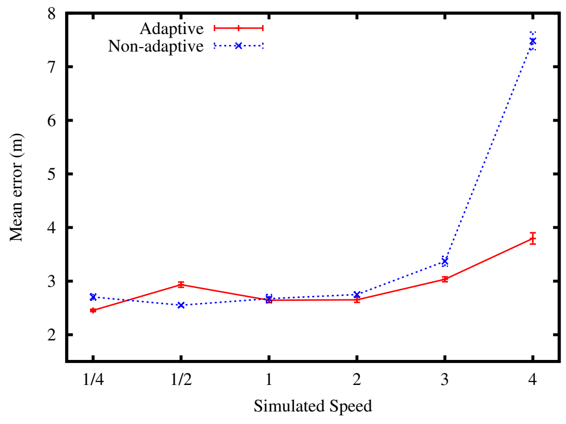


Fig. 4. Localization error at different simulated speeds.

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 starting image for localization.

7. Results

7.1. *Adaptivity of DPSO*

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. Where not stated otherwise, DPSO is used with a swarm size of 80 particles. We varied the simulated speed of the mobile system in Fig. 4. Here, the simulated speed $\bar{v} = 1$ corresponds to the original data collection speed of $v_R \approx 0.6\,\frac{m}{s}$. For speeds k times the original one we use every k-th

Table 1
Varying the number of particles for DPSO and PF, see Fig. 8.

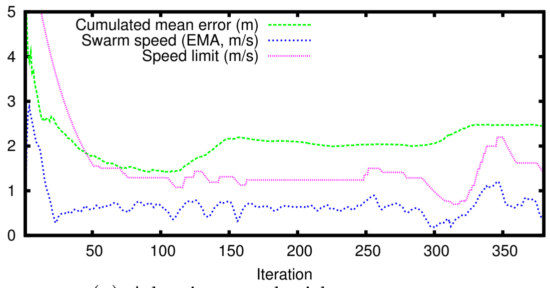
Method	DPSO-40	DPSO-60	DPSO-80	DPSO-100	DPSO-120	PF-g-100	PF-g-300	PF-l-100	PF-l-300
Mean err. (m)	2.84	2.60	2.54	2.50	2.46	3.95	3.39	2.80	2.38
Avg. comp./image	17.9	22.3	25.9	29.1	32.0	40.8	62.4	42.2	69.6

image for localization only, resulting in the simulation hurrying around the loop at higher speed. In analogy, a simulated 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, we set the maximum speed parameter manually to roughly fit the original speed case. For higher speeds, the non-adaptive method clearly fails without manual tuning of the speed limit (Fig. 4).

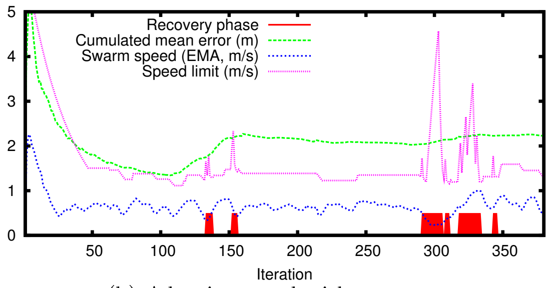
Figure 5 shows two exemplary runs within the mentioned scenario illustrating a recovery phase. The graph in Fig. 5 (a) displays the cumulated error, momentary speed limit and the exponential moving average of the swarm speed during the run. After the initial convergence phase, the swarm speed swings around the actually driven speed of $v_R \approx 0.6\frac{m}{s}$. There are two especially ambiguous situations around iterations 130 and 300, in which the particle swarm is prone to lose the track. The more difficult one, around iteration 300, causes a clear drop in swarm speed and a raise of the error value. The real position is lost for some iterations and recovered around step 320, which requires the swarm to hurry after it with increased speed before resettling.

In Fig. 5 (b), the very same scenario is simulated with the recovery mode activated. In the difficult situation around iteration 300, due to ambiguous image similarity values, a recovery phase is started and the diversity increased. In effect, the error ascent is minor and the velocity swing shorter and of smaller amplitude. Figure 7 compares the course of the mean distances of particles within the swarm for the different runs. The increase in diversity during recovery is visible around iteration 300. It witnesses that the decreasing swarm speed is also due to the swarm diverging, while for the run without recovery, the swarm size remains similar, and while slowing down, it has a higher probability of losing the track.

In Tab. 2, the self-adaptive diversity mechanism comes again into play when a kidnapped-robot scenario is simulated. To do this, the virtual position after half a 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



(a) Adaptive speed without recovery.



(b) Adaptive speed with recovery.

Fig. 5. Examples of cumulated error, absolut speed and speed limit for a single run.

Table 2
Comparing adaptive DPSO in the standard and kidnapped case.

Condition	Standard case, mean err. (m)	Kidnapped case, mean err. (m)
Non-ad.	2.56 ± 0.71	10.54 ± 22.1
Adaptive	2.50 ± 0.35	5.58 ± 14.2

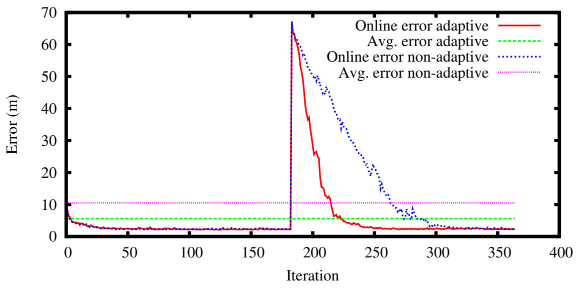


Fig. 6. DPSO in the kidnapped-robot scenario.

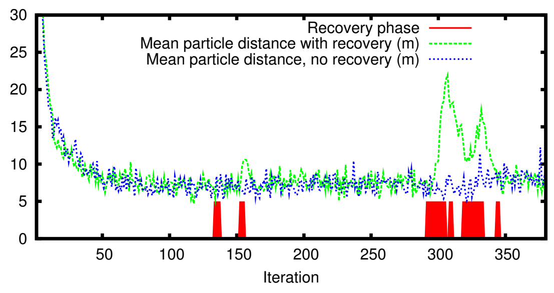


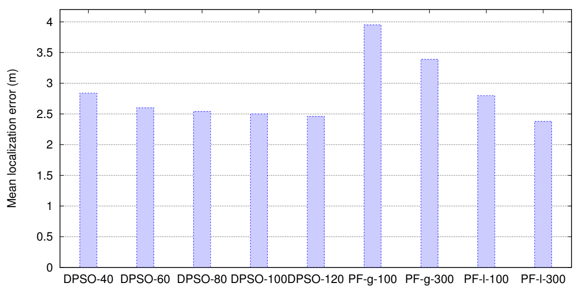
Fig. 7. Illustrating swarm diversity in the example run.

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 68 m at that instant. Of course, the kidnapping reduces performance, but the adaptive method is clearly able to re-trace the position. The mean online localization error of the experiment is plotted in Fig. 6. The performance of the non-adaptive variant and the mean localization errors are also shown.

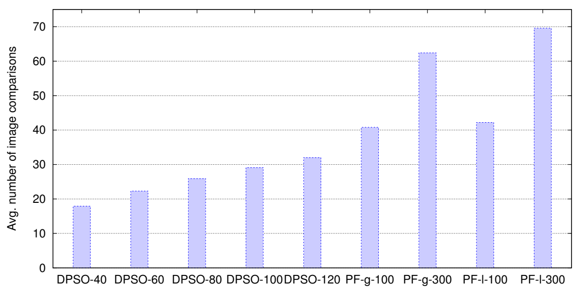
Table 1 and Fig. 8 show results for different numbers of particles, demonstrating that even low population sizes are able to perform well. The largest population tested here still requires much fewer SIFT comparisons than the particle filter, cf. Section 7.2. Smaller populations, admittedly, have more problems recovering in a kidnapping case. We therefore use a swarm size of 80 for the final comparison with the PF.

7.2. *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 Sec. 5. As our dataset did not contain odometry, the particle filter was first employed with a Gaussian motion model with a standard deviation of 4 m (PF-g), while the sampling weights are calculated in analogy to Eq. 3. An alternative directed motion model was formulated for the PF by asssuming a linear motion between two iterations (PF-l): Each particle is assigned a velocity vector which is rotated and scaled using zero-mean Gaussian distributions of standard deviation σ_{rot} for the angles and standard deviation σ_{acc} for acceleration. In Eqs. 7-8, the update step is described formally. Notice that, therein, $\boldsymbol{v}$ and $\boldsymbol{x}$ are 2D vectors, and $\boldsymbol{v}$ is first rotated and then scaled by a log-normally chosen factor modelling change in speed.



(a) Mean localization error.



(b) Avg. number of comparisons per test image.

Fig. 8. Comparison of localization error and comparisons per image with varying particle counts.

$$\boldsymbol{v}(t+1) = e^{N(0,\sigma_{acc})} \cdot \mathrm{rot}(\boldsymbol{v}(t), N(0,\sigma_{rot})), \qquad (7)$$
$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \boldsymbol{v}(t+1). \qquad (8)$$

A grid search over the motion-model parameters σ_{rot}, σ_{acc}, and the initial speed $v_{init} = |\boldsymbol{v}(0)|$ was performed on PF-l with 300 particles (PF-l-300) for the more difficult scenario sunny (round 1) vs. cloudy (round 1) already used in Sec. 7.1. The results are given in Tab. 3, where error values below 2.60 m are accentuated to gain a better overview. The values tested were $\sigma_{rot} \in \{20°, 30°, 40°, 50°, 60°\}$, $\sigma_{acc} \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, and $v_{init} \in \{v_R,\ 2\,v_R,\ 3\,v_R\}$, where v_R again corresponds to the actually driven speed.

Interestingly, small deviations in rotation perform poorly compared to large ones, and higher initial speeds are advantageous. The best results are achieved for $\sigma_{rot} = 60°$, $\sigma_{acc} = 0.001$, and $v_{init} = 3v_R$, which corresponds to about 2 $\frac{m}{s}$. These settings imply a high diversity of particles due to the high velocity and large rotations, where single particle states do not approximate the real motion. The smaller the angle distributions, the closer the PF-l may converge to the position for short periods, and the easier it fails in more ambiguous situations.

On average, there are 60 to 80 image comparisons performed per iteration for the tested PF-l-300

Table 3
Grid search for the PF-l-300 with linear motion model; displayed are error values in m averaged over 365 runs (cf. Sec. 7.2).

	σ_{rot}	20°			30°			40°			50°			60°		
σ_{acc}	v_{init}	v_R	$2v_R$	$3v_R$	v_R	$2v_R$	$3v_R$	v_R	$2v_R$	$3v_R$	v_R	$2v_R$	$3v_R$	v_R	$2v_R$	$3v_R$
10^{-2}		5.43	4.22	4.31	4.79	3.55	3.60	4.49	3.34	3.42	4.37	3.22	3.21	4.23	3.12	3.11
10^{-3}		5.70	3.56	3.22	5.16	2.81	2.61	4.77	2.65	**2.56**	4.54	**2.59**	**2.53**	4.51	**2.58**	***2.51***
10^{-4}		5.62	3.45	3.13	5.18	2.72	2.61	4.70	2.60	**2.55**	4.55	**2.54**	**2.54**	4.44	**2.54**	**2.52**
10^{-5}		5.63	3.43	3.14	5.14	2.72	2.61	4.68	2.60	**2.56**	4.50	**2.56**	**2.54**	4.36	**2.54**	**2.53**

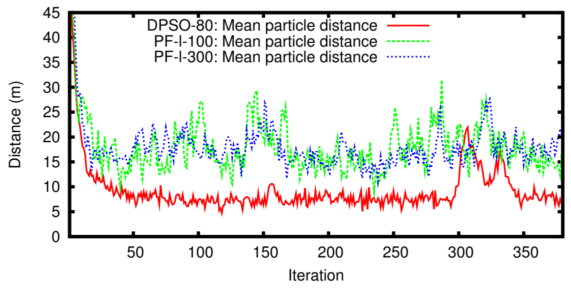
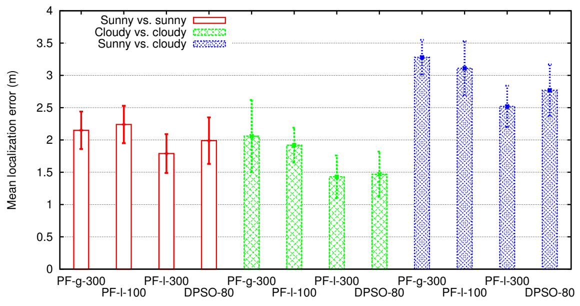
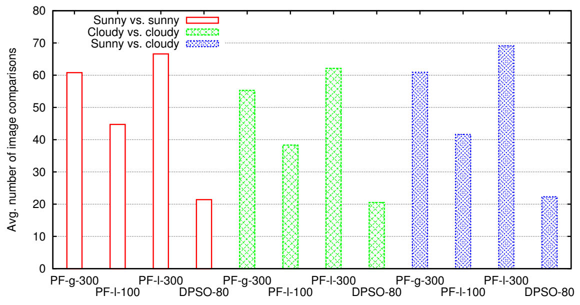


Fig. 9. Comparing swarm diversity of DPSO with PF-l-300.



(a) Mean localization error.



(b) Avg. number of comparisons per test image.

Fig. 10. Final comparison of particle filter to DPSO.

configurations, and 72.15 for the best configuration tested. When the particle count for the PF-l with the advantageous settings is decreased, the number of necessary comparions drops only slowly, since the high speed keeps the diversity permanently high so that the particles are fairly distributed and many training images need to be looked at. This is illustrated in Fig. 9, where the mean particle distances per iteration of DPSO-80, PF-l-100 and PF-l-300 are plotted for the exemplary scenario.

Table 4 and Fig. 10 summarize the mean errors of DPSO using 80 particles compared to the PF using 100 and 300 particles in all three scenarios. The particle filter with simple Gaussian motion and 100 particles (PF-g-100) performs considerably worse than the other methods and is omitted. Note that the indicated standard deviations refer to full rounds and are therefore higher than the deviations within specific pairs of training and test rounds.

Considering the localization accuracy, both the PF-l-300 and the DPSO-80 variants outperform the PF-g methods as well as the PF-l with 100 particles. This is supported by Student's t-test on a significance level below 0.5% in the three scenarios. The PF-l-300 achieves a slightly smaller localization error than DPSO-80, however it requires a large number of image comparisons to do so. The DPSO method needs only half of the comparisons of the PF-l-100 while yielding better localization, and it needs less than a third compared to the PF-l-300.

This can be projected onto the real system runtime, since the SIFT comparison is the most expensive operation of the localization procedure. 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. 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. The image comparisons for an iteration of the PF-l-300 thus take about 0.9 to 1 s and a reduction by two-thirds implies saving about 0.6 s per test image. With regard to [4], where SIFT comparisons took about 40% of the computation time using the PF-g-300 method for localization, an allover speed-up by 25% can be expected.

Table 4
Comparison of particle filter to DPSO

	PF-g-300		PF-l-100		PF-l-300		DPSO-80	
Experiment	Mean err. (m)	Avg. comp. per img.	Mean err. (m)	Avg. comp. per img.	Mean err. (m)	Avg. comp. per img.	Mean err. (m)	Avg. comp. per img.
Sunny vs. sunny	2.15 ± 0.29	60.8	2.24 ± 0.29	44.72	1.79 ± 0.30	66.6	1.99 ± 0.36	21.4
Cloudy vs. cloudy	2.06 ± 0.56	55.3	1.92 ± 0.27	38.31	1.43 ± 0.33	62.2	1.47 ± 0.35	20.5
Sunny vs. cloudy	3.28 ± 0.27	60.9	3.11 ± 0.42	41.62	2.52 ± 0.32	69.1	2.77 ± 0.40	22.3

8. 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 a customized motion model. This is a major advantage, e.g., for the augmentation of black-box systems with independent visual trackers.

For distinctive visual features such as SIFT, which provide 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. Moreover, dynamic speed adaptation makes the system robust with regard to manual parameter settings and the robot's velocity. According to a comparison to particle filter approaches, the swarm-based method reaches similar accuracy but requires substantially fewer of the costly image comparisons.

Our experiments were 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 extremely ambiguous environments such as forests, the swarm approach may lose 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 issue in such scenarios.

The use of visual sensory and robust features is the basis for our localization method. The specific method of image comparison, however, is not fixed, and the general algorithm can be used with any viable feature extraction method by just switching the similarity function. SIFT is a popular choice and serves well for comparisons to further approaches, which could be using iterative SIFT [33], SURF [34], additional geometric constraints or hybrid feature sets. An analysis of the swarm-supported localization with hybrid features in larger scenarios will therefore be considered in future work.

References

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

[2] O. Booij, Z. Zivkovic, B. Kröse, Sparse appearance based modeling for robot localization, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2006, pp. 1510–1515.

[3] O. Booij, B. Terwijn, Z. Zivkovic, B. Kröse, Navigation using an appearance based topological map, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2007, pp. 3927–3932.

[4] C. Weiss, A. Masselli, H. Tamimi, A. Zell, Fast outdoor robot localization using integral invariants, in: Proc. of the 5th Int. Conf. on Computer Vision Systems (ICVS), Bielefeld, Germany, 2007.

[5] S. Se, D. Lowe, J. Little, Vision-based mobile robot localization and mapping using scale-invariant features, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2001, pp. 2051–2058.

[6] D. Scaramzza, R. Siegwart, Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles, IEEE Trans. on Robotics 24 (2008) 1015–1026.

[7] S. S. Beauchemin, J. L. Barron, The computation of optical flow, ACM Comput. Surv. 27 (3) (1995) 433–466.

[8] M. Jogan, M. Artac, D. Skocaj, A. Leonardis, A framework for robust and incremental self-

localization, in: Proc. of the 3rd Int. Conf. on Computer Vision Systems (ICVS), Graz, Austria, 2003, pp. 460–469.

[9] J. Wolf, W. Burgard, H. Burkhardt, Robust vision-based localization by combining an image retrieval system with Monte Carlo localization, IEEE Trans. on Robotics 21 (2) (2005) 208–216.

[10] M. Bennewitz, C. Stachniss, W. Burgard, S. Behnke, Metric localization with scale-invariant visual features using a single camera, in: Proc. of European Robotics Symposium (EUROS-06), Vol. 22, Palermo, Italy, 2006, pp. 143–157.

[11] B. Kröse, O. Booij, Z. Zivkovic, A geometrically constrained image similarity measure for visual mapping, localization and navigation, in: Proc. of the European Conf. on Mobile Robots (ECMR), 2007, pp. 168–174.

[12] S. Se, T. Barfoot, P. Jasiobedzki, Visual motion estimation and terrain modelling for planetary rovers, in: Proc. of the Int. Symposium on Artificial Intelligence for Robotics and Automation in Space (iSAIRAS), Munich, Germany, 2005.

[13] C. Valgren, A. J. Lilienthal, SIFT, SURF and seasons: Long-term outdoor localization using local features, in: Proc. of the European Conf. on Mobile Robots (ECMR), 2007, pp. 253–258.

[14] H. Andreasson, A. Treptow, T. Duckett, Self-localization in non-stationary environments using omni-directional vision, in: IEEE Int. Conf. on Robotics and Automation (ICRA), 2005, pp. 3348–3353.

[15] H. Tamimi, Vision-based features for mobile robot localization, Ph.D. thesis, University of Tübingen, Tübingen, Germany (2006).

[16] S. Thrun, D. Fox, W. Burgard, F. Dellaert, Robust Monte Carlo localization for mobile robots, Artificial Intelligence 128 (1-2) (2000) 99–141.

[17] A. Doucet, N. de Freitas, K. P. Murphy, S. J. Russell, Rao-blackwellised particle filtering for dynamic bayesian networks, in: UAI '00: Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 176–183.

[18] S. Godsill, T. Clapp, Improvement strategies for monte carlo particle filters, in: A. Doucet, N. D. Freitas, N. Gordon (Eds.), Sequential Monte Carlo Methods in Practice, New York: Springer-Verlag, 2000.

[19] R. van der Merwe, N. de Freitas, A. Doucet, E. Wan, The unscented particle filter, in: Advances in Neural Information Processing Systems 13, 2001.

[20] A. Soto, Self adaptive particle filter, in: Int. Joint Conf. on Artificial Intelligence (IJCAI'05), 2005.

[21] S. Zhou, R. Chellappa, B. Moghaddam, Appearance tracking using adaptive models in a particle filter, in: Asian Conf. on Computer Vision, 2004.

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

[23] Q. Wang, J. Liu, Z. Wu, Object tracking using genetic evolution based kernel particle filter, in: 11th Int. Workshop on Combinatorial Image Analysis, IWCIA 2006, 2006, pp. 466–473.

[24] M. M. Drugan, D. Thierens, Evolutionary Markov chain Monte Carlo, Tech. rep., Institute of Information and Computing Sciences, Utrecht University (2003).

[25] A. Treptow, Optimization techniques for real-time visual object detection and tracking, Ph.D. thesis, University of Tübingen, Tübingen, Germany (2007).

[26] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE Int. Conf. on Neural Networks, Perth, Australia, 1995.

[27] X. Li, J. Branke, T. Blackwell, Particle swarm with speciation and adaptation in a dynamic environment, in: GECCO '06: Proc. of the 8th annual Conf. on Genetic and Evolutionary Computation, ACM Press, New York, NY, USA, 2006, pp. 51–58.

[28] H. Liu, A. Abraham, Fuzzy adaptive turbulent particle swarm optimization, in: HIS '05: Proc. of the Fifth Int. Conf. on Hybrid Intelligent Systems, IEEE Computer Society, Washington, DC, USA, 2005, pp. 445–450.

[29] A. R. Vahdat, N. NourAshrafoddin, S. S. Ghidary, Mobile robot global localization using differential evolution and particle swarm optimization, in: D. Srinivasan, L. Wang (Eds.), 2007 IEEE Congress on Evolutionary Computation, IEEE Computational Intelligence Society, IEEE Press, Singapore, 2007, pp. 1527–1534.

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

[31] A. Doucet, On sequential Monte Carlo sampling methods for bayesian filtering, Statistics and Computing 10 (2000) 197–208.

[32] T. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments., in: EvoWorkshops, 2004, pp. 489–500.

[33] H. Tamimi, H. Andreasson, A. Treptow, T. Duckett, Z. Andreas, Localization of mobile robots with omnidirectional vision using particle filter and iterative SIFT, Robotics and Autonomous Systems 54 (9) (2006) 758–765.

[34] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.