A Dynamic Swarm for Visual Location Tracking

Marcel Kronfeld, Christian Weiss, and Andreas Zell

Computer Science Dept., University of Tübingen, Tübingen, Germany
{marcel.kronfeld|c.weiss|andreas.zell}@uni-tuebingen.de

Abstract. The visual localization problem in robotics poses a dynamically changing environment due to the movement of the robot compared to a static image set serving as environmental map. We develop a particle swarm method adapted to this task and apply elements from dynamic optimization research. We show that our algorithm is able to outperform a Particle Filter, which is a standard localization approach in robotics, in a scenario of two visual outdoor datasets, being computationally more effective and delivering a better localization result.

1 Introduction

Environments with uncertainty like noise or dynamic changes are especially challenging for optimization [1]. A possible application for dynamic optimization is self-localization of mobile robots (Fig. 1), which need to know their position to interact with their environment in any useful way. Visual localization is based on an image database and therefore computationally expensive [2]. Moreover, the robot may move between any two iterations of the localization method, making the problem highly dynamic.

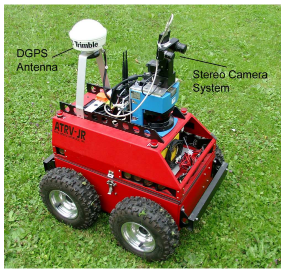


Fig. 1. RWI outdoor robot Arthur.

An up-to-date approach for localization in robotics is the so-called Particle Filter (PF), introduced in Sec. 3. A Particle Swarm Optimization method (PSO, [3]) for visual robot localization was described in [4] from the robotics point of view. This work goes into details of the PSO variant and analyzes parameter settings empirically. Closely related is the work of Vahdat et al. [5], who employed DE and PSO for global robot localization using laser sensors in an indoor environment. Their case treats a larger search space, where standard DE and PSO approaches showed to be superior to a Particle Filter. They did not, however, go into dynamics and thus handled only an initial phase of localization. From the robotics view, there have been several approaches extending Particle Filters with evolutionary concepts, e.g., [6]. Particle swarm algorithms have been extended to dynamic problems, e.g., by boosting diversity or creating sub-swarms to track optima [7].

2 Particle Swarm Optimization

The PSO technique takes as basic idea the flocking behavior of animals. It searches the solution space by assigning velocities $\boldsymbol{v}$ and a neighborhood relationship to the individuals $\boldsymbol{x}$. An individual is in each generation attracted to the best location in its history ($\boldsymbol{p}^h$) and to the best location found by its neighborhood ($\boldsymbol{p}^n$). The classical formulation is given in Eqs. 1 and 2.

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

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2}$$

The neighborhood type may range between small (local), randomized, and large (global), differing especially in the rate of information distribution. The parameters $\phi_{1/2}$ control the impact of the attractors, while $r_{1/2}$ are uniform random samples in $[0,1]$ used as stochastic components. The inertness factor ω controlling the influence of former velocities is usually set < 1 for convergence.

The PSO concept seems to match problems with dynamically changing target functions, and dynamic adaptations of PSO have proven to be successful in this domain [8, 7]. Yet, most work in the direction considers problems where the time between environmental changes is rather long, whereas, for mobile location tracking, the frequency of change is usually very high. When exploiting the rich visual data, the PSO method still offers a promising approach.

3 Visual Localization and SIFT

Visual localization addresses the question of how a mobile system, which has access to a visual representation of its environment (map), can find its position relative to this map and keep track of it during motion. As internal odometry, e.g., sensors measuring the speed of the wheels, is prone to errors, external information is necessary to obtain an initial position estimation and to achieve robust location tracking over time. Vision is a major source of information for humans, so it is appealing to use it for robot localization as well.

Visual localization may be divided into two stages. Firstly, for training, images are collected with position information, e.g. GPS tags, and stored in a database - the map. For later localization, a mobile robot moves through the same environment, takes pictures online, and relates them to the map. Visual data is very sensitive to changes, e.g. in light conditions, so localization requires a robust method for image comparison, which is apt to be time-consuming.

If the size of the reference image set is small, localization is possible by just comparing a new image to every image in the database and choose the best match as position estimation. This gets, of course, infeasible in larger scenarios. Using a probabilistic approach such as a Particle Filter, the set of tested images is reduced to the most probable ones.

A Particle Filter (PF, [9]) is a sequential Monte-Carlo-method for hidden state estimation, which approximates a probability density function p using a finite set of weighted "particles" $\boldsymbol{x}^i$ (not originally related to the particles in PSO). A particle can be seen as a hypothesis on the state of the system at a time, and the particle weights w^i express the *importance* of particles. The PF seeks to deduce from the collected sensor readings $s_{1:t}$ a belief in the current state $\boldsymbol{x}_t$ of the robot: $p(\boldsymbol{x}_t|s_{1:t}) \approx \sum_{i=1}^{n} w_t^i \delta(\boldsymbol{x}_{1:t} - \boldsymbol{x}_{1:t}^i)$ (with Dirac δ).

In a PF iteration, a new proposal distribution is first predicted by advancing the particles using a transition model, e.g., from odometry readings. Then, the particles are reweighed incorporating new external information, judging how probable each hypothesis is. By resampling the particles using the updated weights, the PF concentrates on more promising areas of the state space. For an application using visual data, the weighing may be traced back to an image similarity measure. If the state-space is large, a PF requires many particles, e.g. $n = 300$ in [2], and much more for less distinctive sensory such as laser scanners [9, 5]. PSO is, by contrast, known to be effective with relatively small swarms.

3.1 Interpretation in an Optimization Context

The aim of the PF method is, in terms of optimization: From a given sample set, produce a new set which contains samples of same or higher fitness with respect to the image similarity measure. As the system is mobile, beyond finding an optimum, we need to track it over time. Assuming that position $\boldsymbol{x}_t$ is typically close to $\boldsymbol{x}_{t-1}$ and associating particle velocities with the robot's speed, we argue that PSO together with a distinctive image similarity measure is effective for localization. For such a distinctive function of two images, $m : S \times S \to \mathbb{R}$, we formulate a target fitness function for the map M:

$$f_M(x, t) = m(img_M(x), s(t)) \cdot pen(d(x, p_M(img_M(x)))), \tag{3}$$

where $img_M : X \to M \subset S$ returns the associated training image of a position $\boldsymbol{x}$, i.e., the training image closest to $\boldsymbol{x}$. $s(t)$ is the test image at iteration t, $p_M : M \to X$ delivers the known position for an image in the map. The penalty function $pen : \mathbb{R} \to \mathbb{R}$ reduces the fitness for particles far away from the training data, because localization is feasible only where there is training information. This is done similarly to [2] in terms of a Gaussian function. The problem of tracking a position now corresponds to a dynamic optimization problem: find the optimum $\hat{\boldsymbol{x}}$ of f_M at a time and follow it ensuring a plausible path.

A popular method to find and describe image features is Lowe's Scale Invariant Feature Transform (SIFT) [10]. By using an image's scale-space and assigning oriented features to the SIFT key points, they can be found relatively robust under changing views. The SIFT-match function m_{SIFT} delivers a value in $[0, 1]$ indicating image similarity by calculating the ratio of single feature matches to all possible matches and is used in the function f_M stated above.

4 A Dynamic PSO for Localization

We base the our algorithm on the original PSO formulation [3] using the inertness parameter ω and a maximum velocity v_0. Due to the dynamic tracking requirement, the swarm is not to converge below a certain scale defined by the map. The fitness of an individual at time t is calculated using the SIFT similarity between the current test image and the training image closest to the individual's position (Eq. 3). As SIFT is relatively distinctive, we expect one strong main attractor most of the time and use the global neighborhood as swarm topology.

$$v_i(t+1) = \omega v_i(t) + \phi_0 r_0 \delta_i v_0 + \phi_2 r_2 (p_i^n(t) - x_i(t)), \tag{4}$$
$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{5}$$
$$x_i^q(t+1) = p_i^n(t) + \delta_i N(0, \widehat{q_d}). \tag{6}$$

In Eq. 4, we replaced the $\boldsymbol{p}^h$-component by a random term, because we expect a continuously changing environment where historically good positions lose relevance quickly. ϕ_0 is the weight of the random perturbation, while δ_i normalizes to the range of axis i. As the random perturbation is undirected, r_0 is sampled uniformly from $[-1, 1]$.

A fraction of $\widehat{q_r} = 10\%$ of particles $\boldsymbol{x}^q$ are treated as quantum particles [7] to ease the dynamic tracking, cf. Eq. 6. They are distributed around $\boldsymbol{p}^n$ with standard deviation $\widehat{q_d} = 0.15$. The inertia is usually below 1 to allow for convergence, yet it needs to be high to stress the correlation of robot motion, so we set it to 0.99. The trade-off between the ϕ-values remains important, balancing between exploration and exploitation. Random perturbation is necessary for diversity, but reduces accuracy if too dominant. For scenarios with constant velocity, preliminary tests show robustness towards settings of $\phi_0 \in (0, 2]$. We suggest $\phi_0 = 1$ as a default, while ϕ_2 is set to 0.6 where not stated otherwise. A fraction of $\widehat{h_r} = 10\%$ of particles are allowed a velocity $\widehat{v_0} = 3v_0$ for quick optimum recovery. The best individual at every iteration is taken as position estimation.

4.1 Self-adaptive Parameters

We introduce two self-adaptive mechanisms. By calculating the speed v_{sw} of the swarm's center, we dynamically hold the relation $v_0 \approx 2v_{sw}$. This enables the method to react to speed changes while providing robust tracking at any speed. Also, v_{sw} gives a good estimate of the robot's speed.

SIFT features offer robust image similarity information in outdoor areas, yet situations may still be ambiguous and the real position may get lost. To handle this, we include a mechanism to adapt the swarm diversity: If the 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 swarm lost track of the position. If this happens several times in a row, we start a recovery phase and boost particle diversity by increasing v_0, $\widehat{q_r}$ and decreasing ϕ_2 towards limit values. As soon as the particles' quality increases again, the initial values are gradually restored. Experiments show that the adaptive speed improves tracking and the recovery phase improves robustness, cf. Sec. 6.1. With increasing v_0 the method becomes more sensitive to the setting of ϕ_0. Therefore, we tested several values for ϕ_0 in a setting where recovery phases were important (Sec. 6).

5 Experimental Setting

In the experiments in [2], an RWI ATRV-JR outdoor robot (Fig. 1) collected images in a campus environment. One 320×240 pixel gray scale image per second was taken with a Videre Design SVS camera system at a constant velocity of about 0.6 m/s. The robot is equipped with a differential GPS (DGPS) system, from which ground truth data was read. 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 the robot moved on a smooth trajectory, some wrong GPS values were eliminated as outliers and gaps could be closed by interpolation.

Two different data sets S and C were produced, each consisting of three rounds around a building. A round is 260 m long and contains about 400 images. Three were collected under sunny conditions (S), three more on a cloudy day (C, cf. Fig. 2). The images contain buildings, streets, cars, as well as vegetation. There are also dynamic objects like cars and people passing by. The SIFT features of a round with GPS annotations make up the localization map.

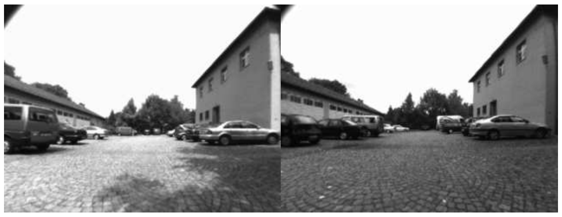


Fig. 2. Example images of the data sets, sunny (left) and cloudy (right).

One experimental run is defined by a training and a test round, the training round constituting the reference map M. At each iteration, an image $img(t)$ of the test round is presented to the algorithm and interpreted as current view of the robot. Where not stated otherwise, the images are presented in the order they where taken in. At each time step, the deviation of the estimated position $\boldsymbol{x}(t)$ to the real position of the test image $img(t)$ gives the online error, the average of which makes up the allover error of the run.

For a full experiment on two rounds, we repeat the localization k times with different starting positions, so k is the number of images in the test round. The average error over these runs give the performance in the experiment. To examine some parameter settings, we experiment on two exemplary rounds, while for the final results, we additionally loop over all the rounds in the data sets.

6 Results

Tab. 1 shows results for different settings of ϕ_0 in a sunny vs. cloudy scenario. They indicate that a setting of $\phi_0 \approx 1$ is favorable, keeping in mind that the random perturbation is also proportional to the maximum speed v_0 which is increased in recovery phases. A high ϕ_0-value increases the number of image comparisons, because the swarm diversity tends to be higher.

ϕ_0	0.005	0.02	0.08	0.32	0.64	0.96	1.28	1.60
Avg.err. (m)	2.64	2.67	2.61	2.50	2.43	2.41	2.44	2.59
Error variance	0.26	0.46	0.28	0.25	0.17	0.09	0.07	0.09

Table 1. Average error (m) varying ϕ_0 with recovery.

Method	PSO-30	PSO-60	PSO-80	PSO-100	PF-100	PF-300
Avg.err. (m)	3.34	2.51	2.42	2.39	3.95	3.39
Avg.comp./img.	16.87	23.68	27.36	30.60	40.8	62.4

Table 2. Varying the number of particles for the PSO-localization.

Recovery active / ϕ_0	$+/1$	$-/1$	$+/0.005$	$-/0.005$
Avg.err. (m)	2.87	3.55	2.91	3.46
Error variance	0.39	3.26	0.54	2.68

Table 3. Comparing localization with and without recovery for sunny vs. cloudy.

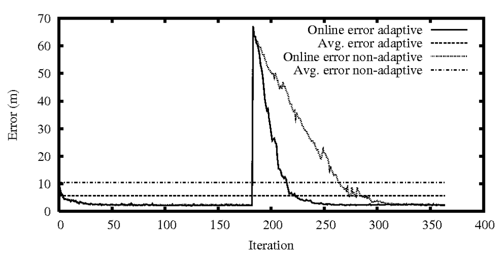
When comparing several swarm sizes for $\phi_0 = 1$ (Tab. 2), the localization performance increases with additional particles as expected. At the same time, the number of comparisons increases, and consequently the computation time. For comparison, the results for a PF with 100 and 300 particles are also shown (cf. Sec. 6.2). For robust localization, PSO uses 80 particles in further experiments.

For the sunny vs. cloudy (S vs. C) situation, the advantage of the adaptive recovery is compared to the performance without recovery in Tab. 3. It shows the averaged errors for 80 particles and $\phi_0 \in \{0.005, 1\}$. For a small ϕ_0, localization tends to be more exact in simple rounds but is more likely to lose the position. For robust localization, we favor setting $\phi_0 = 1$ and adaptive v_0 with recovery.

To demonstrate the effect of the v_0-adaptation, we run simulations with different virtual robot velocities. For $\frac{1}{4}/\frac{1}{2}$ of the original speed, we present the same test image 4/2 times in a row, while for 2/3/4 times the original speed, we present only the $2^{nd}/3^{rd}/4^{th}$ image of the test round, resulting in the virtual speed modified by the respective factor. Fig. 3 (right) shows localization results in the S vs. C case. For the non-adaptive experiment, v_0 is set to roughly twice the original speed, which works for slower speeds but clearly fails for high speeds.

6.1 Kidnapped-Robot Scenario

For localization, a "kidnapping" of the robot, meaning that it is moved by hand without getting informed, is one of the toughest challenges. The robot's position estimate suddenly becomes completely invalid and misleading. In our environment, we simulate kidnapping by adding $\frac{k}{2}$ to the test image index *modulo* k after $\frac{k}{2}$ iterations. Thus, the localization method is forced to jump to the opposite side of the round after converging for half of the run. In Fig. 3 (left), the simulated kidnapping causes an abrupt error of about $68m$ averaged. Yet, the adaptive method quickly finds and retracks the position.



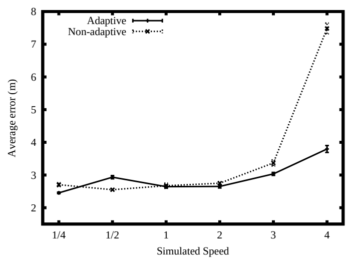


Fig. 3. Kidnapped-robot scenario (left).Varying the virtual speed of the robot (right).

6.2 Final Comparison to a Particle Filter

For the final comparison of the PSO localization method with a Particle Filter approach, we loop over all the rounds in the two data sets, but without testing a round against itself. This means that for S vs. S and C vs. C, there are six, for S vs. C there are nine experiments averaged (Tab. 4).

	PF-100		PF-300		PSO-80,$\phi_0 = 0.005$		PSO-80,$\phi_0 = 1$	
	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm	Avg.err.(m)	#Cm
S vs. S	3.16±0.89	40.0	2.15 ± 0.29	60.8	2.03± 0.42	21.0	2.16± 0.63	24.2
C vs. C	3.46±1.28	36.5	2.06 ± 0.56	55.3	1.45 ± 0.53	19.3	1.49± 0.35	22.7
S vs. C	3.93±0.66	40.4	3.27 ± 0.27	60.9	2.91 ± 0.73	22.8	2.87± 0.62	27.1

Table 4. Comparing the PF to PSO in avg. error and SIFT comparisons per image.

We compare two PSO variants with different scales of the random perturbation with a PF using 100 and 300 particles [2]. The PF-100 localization is rather inaccurate, producing errors of $3m - 4m$, and it requires nearly twice as many image comparisons as the PSO approach. The PF-300 nearly reaches the accuracy of the PSO, but requires about three times as many image comparisons, so the PSO-80 variant is clearly more effective. The difference between low and high random perturbation, depending on ϕ_0, lies mostly in robustness. Since the standard deviations in Tab. 4 refer to experiments with different rounds and not single runs, this is more clearly visible in Tab. 1.

A SIFT comparison of the considered data sets took about 0.015 s on average on our test system, a 2.4 GHz dual core AMD Opteron. An iteration of PF-300 therefore takes approx. $0.8-0.9$ s. Compared to that, PSO reduces the necessary comparisons by more than 50% and saves nearly half a second in every iteration.

7 Conclusions

Visual outdoor localization of mobile systems requires visual data processing and is therefore time-consuming. A typical localization approach from robotics, the Particle Filter, is successful especially with highly ambiguous sensory and non-Gaussian noise. Yet, sparse visual outdoor images, which occur if large areas (e.g. whole cities) are to be mapped in a short time, are relatively distinctive. We therefore employed a PSO heuristic with modifications appropriate to the high dynamics of a mobile robotic system. Using a current method to extract and compare visual features from images, SIFT, we formulated an optimization problem relating a test image sequence to a given map of images. By adding adaptive mechanisms, the robustness of the swarm method could be increased.

Test results using two data sets recorded under different wheather conditions showed that the PSO localization method requires considerably fewer particles and thus less computation time compared to a Particle Filter approach, but is still more accurate. It is able to adapt to different speeds and solves the difficult kidnapped-robot case. We will tackle larger scenarios and incorporate odometry readings from the robot, e.g., as an additional attractor in the PSO-formula, in future work.

References

1. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments – a survey. IEEE Transactions on Evolutionary Computation **9** (2005) 303–317
2. Weiss, C., Masselli, A., Tamimi, H., Zell, A.: Fast outdoor robot localization using integral invariants. In: Proc. of the 5th International Conference on Computer Vision Systems (ICVS), Bielefeld, Germany (March 2007)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE Int. Conf. on Neural Networks, Perth, Australia (1995)
4. Kronfeld, M., Weiss, C., Zell, A.: Swarm-supported outdoor localization with sparse visual data. In: 3rd Europ. Conf. on Mobile Robots. (2007) 259–264
5. Vahdat, A.R., NourAshrafoddin, N., Ghidary, S.S.: Mobile robot global localization using differential evolution and particle swarm optimization. In Srinivasan, D., Wang, L., eds.: 2007 IEEE Congress on Evolutionary Computation, Singapore, IEEE Computational Intelligence Society, IEEE Press (2007) 1527–1534
6. Moreno, L., Garrido, S., Muñoz, M.L.: Evolutionary filter for robust global localization. Robotics and Autonomous Systems **54**(7) (2006) 590–600
7. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: GECCO '06: Proc. of the 8th annual conf. on Genetic and evolutionary computation, New York, NY, USA, ACM Press (2006) 51–58
8. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation. Volume 1. (2001) 94–100
9. Fox, D., Thrun, S., Burgard, W., Dellaert, F.: Particle Filters for Mobile Robot Localization. In: Sequential Monte Carlo Methods in Practice. Springer (2000) 401 – 428
10. Lowe, D.: Distinctive image features from scale-invariant keypoints. Int. Journal of Computer Vision **60**(2) (2004) 91–110