Optimizing Particle Filter Parameters for Self-localization

Armin Burchardt[1], Tim Laue[2], and Thomas Röfer[2]

[1] Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
armin@informatik.uni-bremen.de
[2] Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. Particle filter-based approaches have proven to be capable of efficiently solving the self-localization problem in RoboCup scenarios and are therefore applied by many participating teams. Nevertheless, they require a proper parametrization – for sensor models and dynamic models as well as for the configuration of the algorithm – to operate reliably. In this paper, we present an approach for optimizing all relevant parameters by using the Particle Swarm Optimization algorithm. The approach has been applied to the self-localization component of a Standard Platform League team and shown to be capable of finding a parameter set that leads to more precise position estimates than the previously used hand-tuned parametrization.

1 Introduction

In recent years, the application of particle filters to robot localization, the so-called Monte-Carlo Localization (MCL) approach [4], gained immense popularity and is also a commonly used approach in different RoboCup scenarios since it is able to efficiently cope with non-linear dynamics, multi-modal probability distributions, and the *kidnapped robot problem*. Especially in soccer leagues with local directed vision systems such as the Standard Platform League (SPL) and the Humanoid KidSize League it can be considered as being the de facto standard approach for self-localization. At RoboCup 2009, it was used by all top three teams of the KidSize League – *Darmstadt Dribblers* [5], *FUmanoid* [13], and *CIT Brains* [7] – as well as by two of the top three teams of the SPL – *B-Human* [15] and *Nao Devils Dortmund* [3]. Only the runner-up *Northern Bites* [8] used an EKF-based approach [16]. In other RoboCup Soccer leagues, self-localization is not a topic of research – the Small Size League provides a global view on the whole relevant environment [17] – or can be solved in a different way – the Middle Size League allows omni-directional vision systems that make iterative matching approaches as [11] an efficient solution.

Nevertheless, successfully applying particle filters demands a proper configuration of a variety of different parameters. This affects the sensor models as well

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 145–156, 2011.

as different parameters that control the sensor resetting [12] that is needed to recover from kidnapping situations. The major contribution of this paper is an optimization approach that is able to optimize all relevant parameters in order to obtain more precise position estimates. In general, adequate sensor models can be learned as described in [16] but in this case, they have to be optimized together with the algorithm's parameters due to the strong interdependencies between sensor models and sensor resetting. Our work is based on the Particle Swarm Optimization (PSO) approach by [9] that is able to provide results faster than evolutionary algorithms [1]. PSO has already been applied successfully to gait optimization in the RoboCup context, e. g. by [14]. Since the proposed optimization procedure is computationally intensive, a minor contribution of this work is a modification of the original PSO algorithm that leads to a faster convergence near an optimum, in case of noisy problems. Please note that due to the use of stochastic factors in the algorithm and the noisy evaluation, the particles may converge to a local optimum and a better parameter set may remain undiscovered.

This paper is organized as follows: Section 2 describes the particle filter-based approach and all parameters that need to be optimized. The optimization process as well as the applied Particle Swarm Optimization algorithm are presented in Sect. 3. The conducted experiments and their results are described in Sect. 4.

2 Particle Filter-Based Self-localization

The implementation to become optimized is using an adapted version of the Augmented MCL approach by [6] that extends the standard MCL approach by an adaptive sensor resetting component. This section provides a brief overview of the algorithm and describes in detail all parameters that need to become optimized for obtaining more accurate position estimates.

2.1 Basic Localization Algorithm

Using the Monte-Carlo Localization approach, a set of samples representing the belief of the robot is sequentially updated with the estimate of the current action and weighted by applying likelihood functions of the observed field features. Resampling the samples according to their weightings is used to reduce the number of required samples to approximate arbitrarily probability densities.

In scenarios that include the possibility of robot kidnappings, such as most RoboCup robot soccer leagues, a mechanism for position recovery is needed. In general, the sensor resetting approach by [12] is applied; it generates new samples based on recent sensor measurements and can easily be integrated into the MCL resampling step (cf. line 13 of Algorithm 1). The Augmented MCL approach includes a mechanism for controlling the number of new samples by keeping track of the sample set's overall weighting over time. Recent changes can be interpreted as a delocalization that requires a resetting. In detail, the probability for inserting new samples is computed by using two averaged weightings w_{slow} and w_{fast}

Algorithm 1. Augmented MCL, derived from [16, p.258]

function Augmented_MCL(X_{t-1}, t_y, z_t)**:**

static $w_{\text{slow}}, w_{\text{fast}}$
$\bar{X}_t = \emptyset, X_t = \emptyset, w_{\text{avg}} = 0$
for $m = 1$ to M **do**
  $x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = measurement\_model(z_t, x_t^{[m]})$
  $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M} w_t^{[m]}$
end for
$w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$
$w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$
for $m = 1$ to M **do**
  with probability $\max\{0,\ 1 - {}^{w_{\text{fast}}}/_{w_{\text{slow}}}\}$
    add random pose to X_t
  else
    draw i with probability $\propto w_t^{[i]} + resampling\_threshold \cdot w_{\text{avg}}$
    add $x_t^{[i]}$ to X_t
  endwith
end for
return X_t

of the mean sample weighting w_{avg}. The ratio is calculated using two scalars α_{fast} and α_{slow} that control the reactiveness of this mechanism. These two parameters are subject to the optimization process. One additional parameter is *resampling_threshold* which prevents the effect of particle depletion which might occur easily in case of small sample sets.

The complete algorithm is shown in Algorithm 1.

2.2 Motion Model

The pose of each sample is updated in every cycle using odometry data. Since the data is error-prone a random offset is added (Algorithm 2) depending on the motion speed. In addition to a minimum white noise, factors $(\lambda_x, \lambda_y, \lambda_\theta)$ are used to scale the level of the noise in respect to the motion speed $(\Delta_x, \Delta_y, \Delta_\theta)$:

$$\lambda_x = \max\{\Delta_x \text{Noise}_{\text{TranslationMajorDirWeight}}, \Delta_y \text{Noise}_{\text{TranslationMinorDirWeight}}, \text{Noise}_{\text{Translation}}\} \quad (1)$$

$$\lambda_y = \max\{\Delta_x \text{Noise}_{\text{TranslationMinorDirWeight}}, \Delta_y \text{Noise}_{\text{TranslationMajorDirWeight}}, \text{Noise}_{\text{Translation}}\} \quad (2)$$

$$\lambda_\theta = \max\{|\Delta_x + \Delta_y| \text{Noise}_{\text{RotationMovedDistWeight}}, \Delta_\theta \text{Noise}_{\text{RotationMovedAngleWeight}}, \text{Noise}_{\text{Rotation}}\} \quad (3)$$

All six $Noise_*$ parameters are subject to the optimization process.

Algorithm 2. Noisy update of the state hypotheses (x_t, y_t, θ_t) with action Δ. The noise is scaled by the λ_* argument to the random function *sample*.

function *sample_motion_model*$(\Delta, \langle x_{t-1}, y_{t-1}, \theta_{t-1}\rangle)$:

1: $\begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \leftarrow \begin{pmatrix} x_{t-1}^{[m]} \\ y_{t-1}^{[m]} \end{pmatrix} + R_{t-1}^{[m]} \begin{pmatrix} \Delta_x + sample(\lambda_x) \\ \Delta_y + sample(\lambda_y) \end{pmatrix}$

2: $\theta_t^{[m]} \leftarrow \theta_{t-1}^{[m]} + \Delta_\theta + sample(\lambda_\theta)$

3: **return** $\langle x_t, y_t, \theta_t \rangle$

2.3 Measurement Models

To calculate a weighting for each sample used in the particle filter, a set of measurement models is applied to the features extracted from the camera images. The current implementation of the vision system is able to perceive goal posts, lines, line crossings, and the center circle of a standard SPL field. A detailed description of the system is given in [10].

The implemented sensor models are configured via standard deviations for the likelihood of errors in measured sensor data. An example of the perception of a goal post is depicted in Figure 1: The angle α between the upper and lower post points (p_{upper}) and (p_{lower}) is used to calculate the distance d. Estimation errors in respect to α are rated by the sensor model using the density function of a normal distribution with the standard deviation $\sigma_{\text{GoalpostSizeDistance}}$. In Fig. 1b, the observed angle β to a goal post is compared to the assumed measurement. The difference is used to rate the likelihood of sample s_i based on the standard deviation $\sigma_{\text{GoalpostSizeAngle}}$.

An overview of all standard deviations which are subject to the optimization process used by the measurement models is provided in Tab. 1.

2.4 Sensor Resetting

For the generation of new samples, only perceptions of goal posts are used since this is the only kind of feature providing a unique global direction. To avoid the placement of all new samples on the same spot within one execution cycle, a measurement used for resetting becomes distorted by an uncorrelated random error scaled by $\sigma_{\text{GoalpostSampleBearingDistance}}$ or $\sigma_{\text{GoalpostSampleSizeDistance}}$, depending on the kind of measurement provided by the vision system.

3 Particle Swarm Optimization

To optimize the parameters in Table 1, the Particle Swarm Optimization [9] approach is used as it is known to quickly find parameters in the neighborhood around the minimum of a function (cf. [1]). This section briefly explains the general approach, an extension to PSO that had to be added for this scenario, the applied benchmark function, and the setup for efficient distributed optimization.

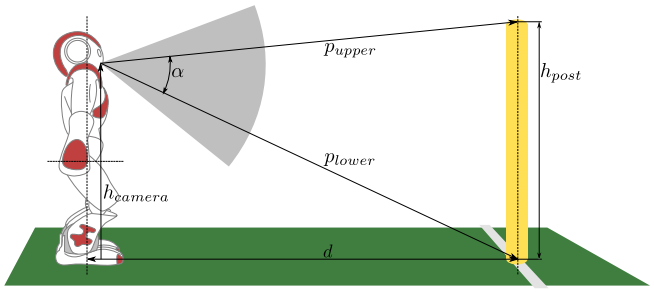
3.1 General Approach

Optimization in this context means to find a parameter set x with the lowest localization error $f(x)$:

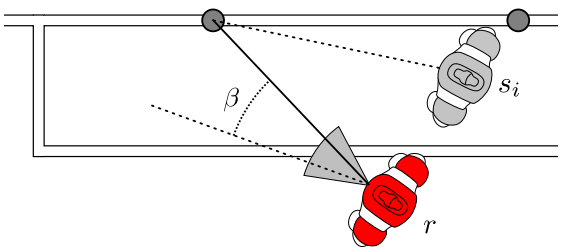
$$x = \underset{x_i}{\operatorname{argmin}} f(x_i)$$

In order to optimize parameters for a self-localization scenario, every parameter set needs to be tested against the same sequence of observations. To assure this condition, the whole optimization process is carried out offline using recorded log data. To compute $f(x)$, the data is augmented with reference results obtained from a global tracking system as described in Sect. 4.2.

The algorithm uses a set of particles P to find results in the given search space (Algorithm 3). Each particle p has a memory $p_{bestpos}$ for the best parameter set found in the previous iterations. The first iteration is randomly initialized within given limits (Table 2). Neighborhoods are used to exchange information between the particles. The velocity p_{vel} is updated with a momentum to direct a particle's position towards a randomized combination of the best position found by itself and the best position known in its neighborhood.



(a)



(b)

Fig. 1. a) Sensor model for distance estimation by observed goal post height α b) Measurement error β for hypothetical field position s_i, the real robot position is r

Table 1. Parameters for *Augmented MCL* algorithm (upper) and implemented sensor models (lower part). All parameters have been determined empirically by human experts.

parameter	value	parameter	value
α_{slow}	0.0059	α_{fast}	0.0060
resampling_threshold	4.0		
$\sigma_{\text{FieldLines}}$	$512\,^{\text{rad}}/_{1024}$	σ_{Corners}	$512\,^{\text{rad}}/_{1024}$
$\sigma_{\text{GoalpostAngle}}$	0.02 rad	$\sigma_{\text{GoalpostBearingDistance}}$	0.4 rad
$\sigma_{\text{GoalpostSizeDistance}}$	0.2 rad	$\sigma_{\text{GoalpostSampleBearingDistance}}$	150.0 mm
$\sigma_{\text{GoalpostSampleSizeDistance}}$	150.0 mm	$\sigma_{\text{CenterCircleAngle}}$	0.2 rad
$\sigma_{\text{CenterCircleDistance}}$	0.4 rad	$\text{Noise}_{\text{Translation}}$	25.0 mm
$\text{Noise}_{\text{TranslationMajorDirWeight}}$	2.0	$\text{Noise}_{\text{TranslationMinorDirWeight}}$	1.0
$\text{Noise}_{\text{Rotation}}$	0.1 rad	$\text{Noise}_{\text{RotationMovedAngleWeight}}$	1.0
$\text{Noise}_{\text{RotationMovedDistWeight}}$	0.002		

3.2 Modification for Noisy Data

Noisy results are apparent in an environment where a simulation or real-world experiments are used to evaluate the performance of a specific parameter set. Since PSO has a memory for the best result found so far, it becomes irritated by outliers. A modification (cf. Algorithm 4) has been made to the PSO algorithm (cf. Algorithm 3) in line 4 to lessen this effect. After the evaluation of a parameter set, the result is compared with the particle's best result as in the original implementation. If the memory is not updated with a better position, the previous rating is degraded by a configurable factor κ. This allows the algorithm to escape from a local minimum over time.

Algorithm 3. *Particle Swarm Optimizer* (cf. [2, Page 80ff.]).

function PSO():

1: **while** continueSearch **do**
2: **for** $p \in P$ **do**
3: $p_{result} \leftarrow f(p_{pos})$
4: **if** $p_{bestresult} > p_{result}$ **then**
5: $p_{result} \leftarrow p_{bestresult}$; $p_{bestpos} \leftarrow p_{pos}$
6: **end if**
7: **end for**
8: **for** $p \in P$ **do**
9: $p_{neighbestpos} \leftarrow$ *FindBestNeighbour(p)*
10: $p_{vel} \leftarrow c_1 r_1 p_{vel} + c_{max} r_2 (p_{bestpos} - p_{pos}) + c_{max} r_3 (p_{neighbestpos} - p_{pos})$
11: $p_{pos} \leftarrow p_{pos} + p_{vel}$
12: **end for**
13: **end while**

Algorithm 4. Modification to PSO.

1: **if** $p_{\text{bestresult}} > p_{\text{result}}$ **then**
2: $p_{\text{bestresult}} \leftarrow p_{\text{result}}$; $p_{bestpos} \leftarrow p_{pos}$
3: **else**
4: $p_{\text{bestresult}} \leftarrow p_{\text{bestresult}}(1 + \kappa)$
5: **end if**

3.3 Benchmark Function

To rate a parameter set x, a benchmark function is required to estimate an error $f(x)$. In this scenario, the Euclidean distance between the estimated field position $p(t, x)$ using parameter set x and the reference position $p'(t)$ is used for every data set corresponding to a captured camera image t (frame). The camera is able to capture images at a rate of 30 Hz, some frames are missing ground truth information due to a fallen robot or obscured marker and are therefore ignored for parameter benchmarking. The self-localization is conducted using the complete input data stored in a log file for one parameter set under test. The image processing is only conducted once while recording the log file and storing the extracted perceptions. We used the distance as the error per frame but it is also possible to use a linear combination of the square or logarithmic translational and rotational distances. The mean of the per frame errors is used as parameter rating for a single log file evaluation:

$$f(x) = \frac{\sum_{t=1}^{\text{frames}} \|p(t, x) - p'(t)\|}{\text{frames}}.$$

3.4 Distributed Computing

Since the evaluation of a previously recorded log file consisting of about 17000 frames is computationally expensive, the implementation of the optimizer supports distribution by using multiple processes controlled over TCP/IP network connections. The number of particles determines the maximum number of processes. In the current implementation, all parameters have to be evaluated before the velocity and position can be updated to generate a new set of particles.

4 Experiments and Results

The proposed approach has been applied to recorded game scenes with two Nao robots in each team. During the experiments, all perceived data of one robot became recorded and merged with reference data from a global tracking system.

4.1 Evaluation of PSO Modification

Before using the proposed PSO modification for optimization, it was successfully tested with *Griewank30D*, a function commonly used for optimization

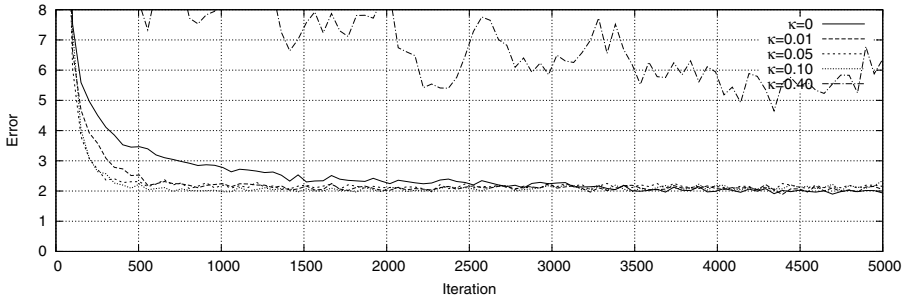


Fig. 2. Benchmark function $f_{\text{Griewank30D}}$ with added noise $\mathcal{N}_{0,1}$, search space is $[-300; 300]^{30}$

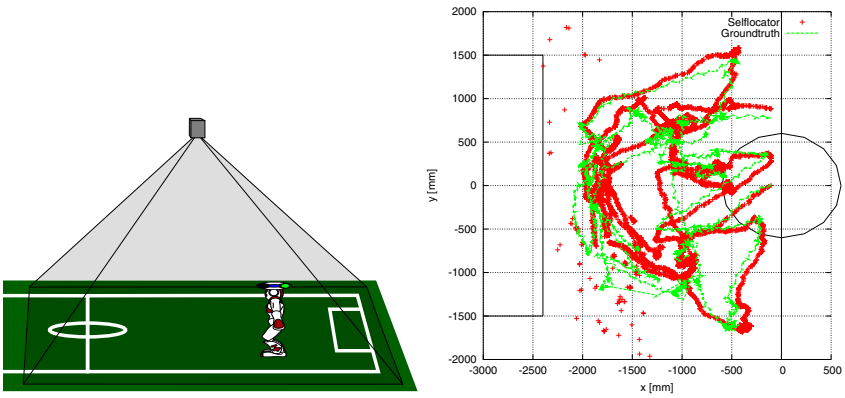
benchmarks (Figure 2). The slow degrading of old ratings ($\kappa = 0.01, \kappa = 0.05$) accelerates the optimization within the first 3000 frames in comparison to the non-modified PSO ($\kappa = 0$). The performance is significantly worse for high levels ($\kappa = 0.4$) of degradation.

4.2 Global Tracking System

To rate the performance of a parametrization the SSL-Vision application (cf. [17]) is used as a ground truth system. A camera is mounted above the field (Figure 3a) to keep track of a marker fixed on the head of the Nao. The SSL-Vision is calibrated to provide the projection of the marker position to the ground in a global coordinate system. This information is transmitted to the robot via WLAN, the offset caused by tilt of the robot body and different head joint angles is compensated using filtered sensor readings of the accelerometer, gyroscope, and measured joint angles. The result is used as ground truth position (Figure 3b).

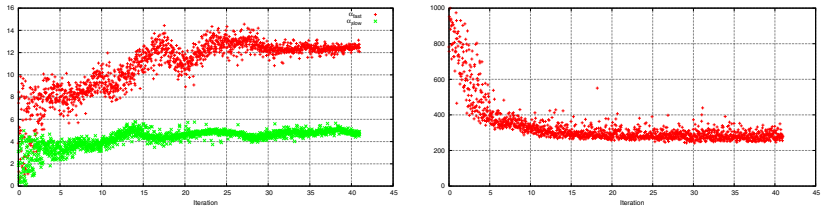
4.3 Optimization

The Particle Swarm Optimizer is configured as follows: number of particles: 40, full neighborhood, $c_1 = 0.69$, $c_{\max} = 1.43$, $\kappa = 0.1$. The values for c_1 and $c_{\max}$ are suggested in [2], a large neighborhood, the amount of particles and the value used for degradation of results (κ) were found to produce good results in previously conducted experiments. Modifications of these parameters may improve the performance of the optimizer but they are computationally expensive to evaluate. Two log files containing recognized field features like goal posts and field lines out of ten minutes of game play are used as training data. Each evaluation is repeated three times, the mean of these measurements is used as parameter rating. The optimization progress was canceled after 41 iterations due to a lack of progress. The best parameter set was selected by re-evaluating all particles of the last iteration for 16 times. The position of the particles in the search space are displayed in Figure 4a using the example of α_{slow} and α_{fast}, the rating of the particles in each iteration is shown in Figure 4b.



(a) (b)

Fig. 3. *a)*Schematic view of the ground truth setup used *b)*Comparison of ground truth data with self-localization



(a) (b)

Fig. 4. *a)* Particle position for α_{slow} and α_{fast} during optimization *b)* Result of evaluation of particles per iteration step

4.4 Interpretation of Results

By comparing the optimized parameter values in Table 2 to the original values in Table 1, big differences are apparent. The large values for α_{slow} and α_{fast} suggest that insertion of new samples by sensor resetting or randomized field positions is used to a greater extend. The small values were originated from a scenario with more landmarks used for sensor resetting and thus introducing more ambiguous state hypotheses.

The *resampling_threshold* was reduced to 0, indicating that the modification of the Augmented MCL in line 15 of Algorithm 1 in comparison to the Augmented MCL in [16] to support particles with low weightings had no significant benefit.

The σ_* values were mostly adjusted to larger values with the exception of $\sigma_{\text{GoalpostBearingDistance}}$ and $\sigma_{\text{CenterCircleDistance}}$. These higher standard

Table 2. Parameter set of best particle after 41 iterations

parameter	initial range	value
α_{slow}	$0.0001 - 5.0$	4.807
α_{fast}	$0.0001 - 10.0$	12.43
resampling_threshold	$0.0 - 100.0$	0.0
$\sigma_{\text{FieldLines}}$	$16 - 4000\,{}^{\text{rad}}/_{1024}$	$1883.0\,{}^{\text{rad}}/_{1024}$
σ_{Corners}	$16 - 4000\,{}^{\text{rad}}/_{1024}$	$747.8\,{}^{\text{rad}}/_{1024}$
$\sigma_{\text{GoalpostAngle}}$	$0.01 - 2.0$ rad	0.96 rad
$\sigma_{\text{GoalpostBearingDistance}}$	$0.01 - 2.0$ rad	0.32 rad
$\sigma_{\text{GoalpostSizeDistance}}$	$0.01 - 2.0$ rad	0.436 rad
$\sigma_{\text{GoalpostSampleBearingDistance}}$	$0.01 - 2000.0$ mm	1878.0 mm
$\sigma_{\text{GoalpostSampleSizeDistance}}$	$0.01 - 15000.0$ mm	8418.0 mm
$\sigma_{\text{CenterCircleAngle}}$	$0.01 - 2.0$ rad	0.402 rad
$\sigma_{\text{CenterCircleDistance}}$	$0.01 - 2.0$ rad	0.08 rad
$\text{Noise}_{\text{Translation}}$	$0.0 - 100.0$ mm	115.2 mm
$\text{Noise}_{\text{TranslationMajorDirWeight}}$	$0.0 - 20.0$	5.11
$\text{Noise}_{\text{TranslationMinorDirWeight}}$	$0.0 - 20.0$	1.13
$\text{Noise}_{\text{Rotation}}$	$0.0 - 1.0$ rad	0.072 rad
$\text{Noise}_{\text{RotationMovedAngleWeight}}$	$0.0 - 10.0$	2.7
$\text{Noise}_{\text{RotationMovedDistWeight}}$	$0.0 - 0.1$	0.0

deviations can be interpreted as a certain compensation for the elimination of *resampling_threshold* as they lead to more homogeneous weightings within a sample set.

The uncertainty introduced in the motion update is slightly reduced for the minimum rotational noise and nearly unchanged concerning the factor for rotational noise by moved distance. The ratios for random translational errors by moved distance ($\text{Noise}_{\text{TranslationMajorDirWeight}}$, $\text{Noise}_{\text{TranslationMinorDirWeight}}$) and the velocity independent minimum error $\text{Noise}_{\text{Translation}}$ are increased.

After the optimization, the overall precision of the self-localization has been increased. Comparisons are shown in Table 3 and Figure 5.

Table 3. The performance of the self-localization is compared by reprocessing data from a log file not used for optimization 100 times

log file	min	max	mean error
log file A (original)	242.7 mm	332.2 mm	292.3 mm (±6.1 %)
log file A (optimized)	160.0 mm	222.2 mm	188.9 mm (±8.4 %)
log file B (original)	283.5 mm	364.7 mm	307.7 mm (±6.0 %)
log file B (optimized)	154.2 mm	231.0 mm	198.5 mm (±8.9 %)

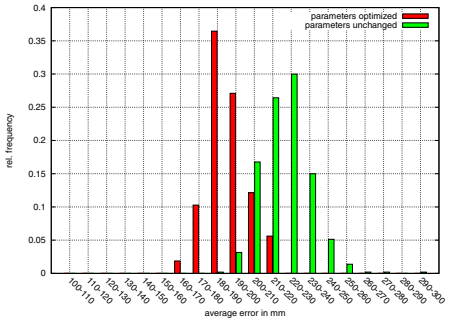


Fig. 5. Histogram of mean errors used for the evaluation of a parameter set (log file was used in training data set)

5 Conclusions and Future Work

In this paper, we presented an approach for optimizing the parameters of a robot's self-localization component. The approach has been successfully applied to a Standard Platform League game scenario. The results did not only lead to a more precise position estimate but also provided insights into the necessity of certain parameters of the applied localization algorithm.

Due to computational constraints set by the robot hardware, the number of samples (which is currently set to 100) used in the particle filter has not been subject to the conducted optimization. As the runtime of the self-localization scales linearly with the number of samples, including this parameter would also demand a more complex evaluation function for the optimizer, trading off computing time versus precision.

In the future, more experiments have to be conducted to study the effects of different error functions (i. e. weighted mean of translational and rotational error). Another interesting extension might also be the inclusion of more variables, e. g. those controlling the image processing modules.

References

1. Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
2. Clerc, M.: Particle Swarm Optimization. ISTE, London (2006)
3. Czarnetzki, S., Kerner, S.: Nao Devils Dortmund - team description for robocup 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 58–68. Springer, Heidelberg (2010)
4. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte-Carlo localization: Efficient position estimation for mobile robots. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, FL, USA, pp. 343–349 (1999)

5. Friedmann, M., Petersen, K., Petters, S., Radkhah, K., Scholz, D., Thomas, D., von Stryk, O.: Darmstadt Dribblers - team description for humanoid kidsize league of robocup 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
6. Gutmann, J.S., Fox, D.: An experimental comparison of localization methods continued. In: Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland, vol. 1, pp. 454–459 (2002)
7. Hayashibara, Y., Minakata, H., Seike, Y., Ogura, S., Ichizawa, K., Machi, K., Takamatsu, K., Yoshizawa, S., Yamada, Y., Horiuchi, T., Fukuta, M., Fujita, S., Irie, K., Kaminaga, H., Kawakami, T., Nishizaki, M.S.Y., Sakamoto, H.: CIT BRains (kid size league). In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
8. Hermans, T., Strom, J., Slavov, G., Morrison, J., Lawrence, A., Krob, E., Chown, E.: Northern Bites 2009 team report (2009)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, NJ, USA, vol. 4, pp. 1942–1948 (1995)
10. Laue, T., de Haas, T.J., Burchardt, A., Graf, C., Röfer, T., Härtl, A., Rieskamp, A.: Efficient and reliable sensor models for humanoid soccer robot self-localization. In: Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T. (eds.) Proceedings of the Fourth Workshop on Humanoid Soccer Robots in Conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots, Paris, France, pp. 22–29 (2009)
11. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: An efficient and accurate approach for robot self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 142–153. Springer, Heidelberg (2006)
12. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modeled mobile robots. In: Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, CA, USA, vol. 2, pp. 1225–1232 (2000)
13. Moballegh, H.R., Hohl, G., Landgraf, T., Fischer, B., Langner, T., Fassbender, T., Otte, S., Stoll, K., Tuchscherer, A., Steig, D., Heinrich, S., Mielke, S., Seifert, D., Kukulski, M., Kahlert, B., Rojas, R.: FUmanoid team description 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
14. Niehaus, C., Röfer, T., Laue, T.: Gait optimization on a humanoid robot using particle swarm optimization. In: Pagello, E., Zhou, C., Menegatti, E., Behnke, S. (eds.) Proceedings of the Second Workshop on Humanoid Soccer Robots in Conjunction with the 2007 IEEE-RAS International Conference on Humanoid Robots, Pittsburgh, PA, USA (2007)
15. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-Human team report and code release 2009 (2009), `http://www.b-human.de/download.php?file=coderelease09_doc`
16. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
17. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 425–436. Springer, Heidelberg (2010)