Monte Carlo Localization for Robocup 3D Soccer Simulation League

Luis Aguiar
Electronic Engineering Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: luis.guilherme@aluno.ita.br

Marcos Máximo
Computer Science Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: mmaximo@ita.br

Samuel Pinto
Electronic Engineering Division
Aeronautics Institute of Technology
Praça Marechal Eduardo Gomes, 50
Vila das Acácias, 12228-900
São José dos Campos, SP, Brazil
Email: sacepi@gmail.com

***Abstract*—This paper presents the application of the localization technique known as Monte Carlo Localization for the estimation of the global pose of virtual robots agents in the Robocup 3D Soccer Simulation League. It includes the overall theoretical background concerning this filtering algorithm, the stochastic modeling for this localization problem, an overview of this method's implementation, simulation results and performance evaluation.**

1. Introduction

Mobile robots have been recently accomplishing great technological advancements and visibility due to their wide range of applications in the real world. Latest efforts concerning humanoid robotics, one of the forefronts of robotics research, include the necessity for the robot to interact with the environment and therefore self-localize precisely. The task of estimating the robot's global position and orientation using its movement commands and sensor information is called localization, and that field has been proving to have great importance for performing sophisticated strategies and decision making algorithms in humanoid soccer.

RoboCup Soccer 3D Simulation League is a particularly interesting challenge concerning humanoid robot soccer. It consists of a simulation environment of a soccer match with two teams, each one composed by up to 11 simulated NAO Robots from Aldebaran Robotics, the official robot used for RoboCup Standard Platform League since 2008. RoboCup Soccer 3D competition allows the possibility for great enhancements in the design and implementation of multi-agent high-level behaviors at the same time it provides a solid low level platform for a realistic physical simulation of the game. Hence, through this combination, it is a opportunity to research, implement and test advanced algorithms in the robot soccer challenge before moving to the next step and apply these techniques in real robots.

This work is based on the the development of a localization system for the team ITAndroids Soccer3D. ITAndroids is a robotics research group at Technological Institute of Aeronautics which was refounded in mid-2011 after years of inactivity. To motivate the research, the group participates in national and international competitions of RoboCup's Soccer 2D, Soccer 3D and Humanoid Kid Size Leagues, as also of IEEE's Humanoid Robot Racing and IEEE's Very Small Size Soccer (VSS) categories. ITAndroids Soccer3D team has achieved some significant results in latest editions of RoboCup (top 12 in 2013 and 2015) and Latin America Robotics Competition (LARC) (1st place in 2012; 2nd place in 2013, 2014 and 2015).

For approaching this localization problem for this competition, it was used an extensively widespread probabilistic technique called Monte Carlo Localization. This technique, known for a good estimation performance, draw particles which represent the posterior non-parametric distribution of the global agent pose. This model merges the information from the robot's locomotion with sensor measurements, and particularly, in this case, sensoring is based on landmarks observations. Moreover, the localization needs to deal with Soccer 3D's simulator frequently teleporting the agent when certain situations occur, which is an instance of the problem known as robot kidnapping

The purpose of this paper is to present the implementation of the mentioned localization technique in the described problem, as well as assessing its performance based on the Root Square Error (RSE), calculated during the simulation, and the computational time. In this work, Section 1 gives a presentation and overview of the problem, Section 2 describes the applied algorithm with some theoretical background and Section 3 provides the stochastic model of the RoboCup Soccer 3D problem. In Section 4 follows some simulation results, in Section 5 we can see the main conclusions, future works and final considerations, and finally, Section 6 presents the people and entities who contributed to this work.

2. Probabilistic Localization

The key idea about localization from a probabilistic approach is representing robot's pose x_t at time step t by its probability distribution function (pdf) throughout the space, and by using mathematical techniques it is possible to reach greater precision in estimating robot's real position. The

probabilistic problem is usually modeled as a first order Markov Chain as described below:

$$x_{t+1} = f(x_t, \epsilon_t) + u_t \tag{1}$$

$$z_t = h(x_t) + v_t \tag{2}$$

Equation (1) represents the movement model of our agent, where f is its propagation function, ϵ_k is the control input and u_t a random noise associated with the process. At the same time, equation (2) is the measurement model of the system, given sensor observation z_t and another additive random noise associated v_t.

Our purpose, therefore, is to inductively merge these models and obtain a final pose x_t, given by the pdf $p(x_t|z_{1:t})$. The approach is through sample representation, exploiting the duality between the distribution function and random samples generated by it. In particle filter, the algorithm used in this work, these samples are called particles, being of the same nature of the robot's state. $X_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$.

Algorithm 1 Monte Carlo Localization, derived from [[1]p.252][[2]p.10]

Initialization
Draw $x_0^{[m]} \sim p(x_0), m = 1, \ldots, M$
$w_0^{[m]} \leftarrow p(z_0|x_0)$
$w_0^{[m]} \leftarrow \frac{w_0^{[m]}}{\sum_{m=0}^{M} w_0^{[m]}}$
$X_0 \leftarrow X_0 \cup \langle x_0^{[m]}, w_0^{[m]} \rangle$
function PARTICLE FILTER(X_{t-1}, z_t)
    $\overline{X_t} = X_t \leftarrow \emptyset$
    for $m = 1$ to M **do**
        draw $x_t^{[m]} \sim p(x_t|x_{t-1}^{[m]})$
        calculate $w_t^{[m]} \leftarrow\propto p(z_t|x_t^{[m]})$
        $\overline{X_t} \leftarrow \overline{X_t} \cup \langle x_t^{[m]}, w_t^{[m]} \rangle$
    end for
    $X_t \leftarrow$ RESAMPLE($\overline{X_t}$)
    $\hat{x_t} \leftarrow \frac{1}{M} \sum_{m=0}^{M} x_t^{[m]}$
    return $\hat{x_t}, X_t$
end function

As can be also seen in [1] and in [3], the overall idea about the algorithm follows other traditional techniques. We assume that the particle set X_{t-1} represents the previous iteration distribution, then we draw new particles from the previous one sampling from the importance function $p(x_t|x_{t-1}^{[m]})$. This step is called **prediction phase**, as you can predict the actual position based on the motion model, however it is corrupted by the random process noise and in a long term prediction-only simulation, the estimation tends to run away from the real position.

In order to control this natural error, the key point of the technique is approximating a specific distribution sampling from another distribution, since the intended final pdf $p(x_t|z_{1:t})$ is originally unknown. Thus, starting from the set of particles $\overline{X_t}$, distributed according to the importance function $p(x_t|x_{t-1}^{[m]})$, if we introduce the weight $w_t^{[m]}$ as in Eq. (3), given by the ratio of the two distribution functions in $x_t^{[m]}$, Eq. (4) holds, where the function $g(x)$ can be any measurable function.

$$w_t^{[m]} = \frac{p(x_t^{[m]}|z_{1:t})}{p(x_t^{[m]}|x_{t-1}^{[m]})} \tag{3}$$

$$\left[\sum_{m=1}^{M} w_t^{[m]}\right]^{-1} \sum_{m=1}^{M} g(x_t^{[m]}) w_t^{[m]} \xrightarrow{M\to\infty} \int_A \mathrm{g(x_t)p(x_t|z_{1:t})\,dx_t} \tag{4}$$

In other words, it is possible to calculate any expectancy over the posterior distribution in any given area A through the particles from the prediction phase, as long as we have enough particles. Moreover, can be proven that $w_t^{[m]}$ can be given in a calculable form, present in (5). This step is called **filtering phase**, as the target function reaches the desirable distribution with estimate closer to the robot's real position.

$$w_t^{[m]} = \frac{w_{t-1}^{[m]} p(z_t|x_t^{[m]})}{\sum_{m=1}^{M} w_t^{[m]}} \tag{5}$$

In the resample step we draw, from the predict set of particles, M particles according to each weight and allowing repetition, compounding the final set X_t. The purpose is to decrease particles' variance, reducing the number of particles with low weight too far from the actual position, otherwise it would be necessary many more particles to reach the same approximation performance. Since this procedure incorporates the weights into the sampling process, it changes the particles distribution to the desired posterior one. The weights can be reseted to $\frac{1}{M}$, and the final pose estimate can be computed by the average of particles positions.

The resample algorithm used can be found in [1] and in [4] and it's advantage is the fact that, instead of independently resampling, the algorithm consists of a sequential stochastic process, covering all the particle space and sorting particles according to a single random number. The main advantage of this resampling algorithm is its computational complexity of O(M), essential for a practical performance and the computational viability of the particle filter.

Despite self-localizing properly, Algorithm (1) is limited dealing with practical issues, such as false positives, sensor errors and the eventual lost of the robot, in the case of the kidnapping problem. As proposed in [1](p. 256-261) and in [5], an augmented Monte Carlo Localization (Algorithm 2) is able to deal with these restrictions.

The main idea is to compute experimentally the observation likelihood and store short-term and long-term values of this probability. It can be made by the parameters α_{slow} and α_{fast}, where $\alpha_{slow} << \alpha_{fast}$, which represent the decay rates for calculating the observation qualities. Then, each particle is resetted with probability $max(0.0, 1.0 - \frac{w_{fast}}{w_{slow}})$, suggesting that if the short-term observation quality is worse than the long-term one, particles must be resetted according to this difference, otherwise no further action is needed.

Algorithm 2 Augmented Monte Carlo Localization

Initialization
draw uniform particles
$w_{slow} = w_{fast} \leftarrow -1.0, w_t^{[m]} \leftarrow 1/M, \forall m$
$X_0 \leftarrow X_0 \cup \langle x_0^{[m]}, w_0^{[m]} \rangle$
function UPDATE($X_{t-1}, Z_t, w_{slow}, w_{fast}$)
  $w_{obs} \leftarrow 0, \overline{X_t} = X_t = \emptyset$
  for $m = 1$ to M **do**
    $x_t^{[m]} \sim p(x_t^{[m]} | x_{t-1}^{[m]})$
    $p_{obs} \leftarrow p(Z_t, x_t^{[m]})$
    $w_{obs} \leftarrow w_{obs} + p_{obs}$
    $w_t^{[m]} \leftarrow w_{t-1}^{[m]} p_{obs}$
  end for
  $w_t^{[m]} \leftarrow \frac{w_t^{[m]}}{\sum_{m=0}^{M} w_t^{[m]}}$
  $\overline{X_t} \leftarrow \overline{X_t} \cup \langle x_t^{[m]}, w_t^{[m]} \rangle, \forall m$
  $w_{slow} \leftarrow (1 - \alpha_{slow}) w_{slow} + \alpha_{slow} w_{obs}$
  $w_{fast} \leftarrow (1 - \alpha_{fast}) w_{fast} + \alpha_{fast} w_{obs}$
  $p_{reset} \leftarrow \text{MAX}(0.0, 1.0 - \frac{w_{fast}}{w_{slow}})$
  $X_t \leftarrow \text{RESAMPLE}(\overline{X_t})$
  for $m = 1$ to M **do**
    if RANDOM() $< p_{reset}$ **then**
      $(\overline{z_1}, \overline{z_2}) \leftarrow$SAMPLETWOLANDMARKS($Z_t$)
      SENSORPARTICLERESET($\overline{z_1}, \overline{z_2}, x_t^{[m]}$)
    end if
  end for
  $\hat{x_t} \leftarrow \frac{1}{M} \sum_{m=0}^{M} x_t^{[m]}$
  return $\hat{x_t}, X_t, w_{slow}, w_{fast}$
end function

That action implies that, when occurring a big shift in the measurement likelihood, is better to change particles location based on sensor information solely (see [6]) than continuously losing particles variety in the resample step. Moreover, the smoothed calculation of w_{fast} and w_{slow} reduce the error caused by false positives and momentary sensor failures (see Algorithm 2).

3. Stochastic Modeling for 3D Soccer Simulation

In RoboCup 3D Soccer Simulation, for the global localization problem, the robot's pose is given by its two dimensional coordinates in the field, x e y, and by the horizontal angle its torso is heading, therefore, it is of the form $x_t^{[m]} = [x, y, \psi]$.

The motion model is given by the information provided by the walking engine in the control layer, in the form of 3-dimensional displacement of the center of the torso, given in the local torso coordinates frame of the previous iteration. In that way, locomotion information is a vector $\Delta d_t = [\Delta x_t, \Delta y_t, \Delta \psi_t]$. However, it presents a significant difference from the real movement due to slip, mostly in high speed circumstances. For solving this issue, we multiply each element of the vector Δd_t by a constant parameter, as can be seen in 6.

$$d'_t = \begin{bmatrix} \Delta x'_t \\ \Delta y'_t \\ \Delta \psi'_t \end{bmatrix} = \begin{bmatrix} \alpha \Delta x_t \\ \beta \Delta y_t \\ \gamma \Delta \psi_t \end{bmatrix} \tag{6}$$

This new vector represents the actual displacement used in the movement model, the parameters α, β and γ were manually tuned comparing the real position evolution and its estimation through the walk algorithm. Therefore, we reach the following model for the position update of each particle.

$$\begin{bmatrix} x_t^{[m]} \\ y_t^{[m]} \\ \psi_t^{[m]} \end{bmatrix} = \begin{bmatrix} x_{t-1}^{[m]} + \Delta x'_t cos(\psi_{t-1}) - \Delta y'_t sin(\psi_{t-1}) \\ y_{t-1}^{[m]} + \Delta x'_t sin(\psi_{t-1}) + \Delta y'_t cos(\psi_{t-1}) \\ \psi_{t-1}^{[m]} + \Delta \psi'_t \end{bmatrix} + \begin{bmatrix} u_x \\ u_y \\ u_\psi \end{bmatrix} \tag{7}$$

Where $u_x \sim N(0, \sigma_x^2)$, $u_y \sim N(0, \sigma_y^2)$ and $u_\psi \sim N(0, \sigma_\psi^2)$ are Gaussian noises associated with the process. The values σ_x^2, σ_y^2 and σ_ψ^2 were also manually tweaked comparing the real evolution and the predict one.

The observation model is given by landmarks detections (flags and goal posts). In Robocup Soccer 3D, this information is provided by the server in the form of perceptors, strings which contain the distance to the center of the objects described as a vector in spherical coordinates, which is the 3-dimensional distance and also the horizontal and latitudinal angle in relation to the camera. In the perception layer, this information is decodified and the distances passed from the camera's referential to the same two-dimensional system of the robot's pose, thus giving the horizontal distance and bearing angle from the object. The perceptors documentation can be found in [7].

Landmarks observations are also corrupted by Gaussian noise, in both horizontal distance and angle, described by the variances σ_d^2 and σ_ψ^2. We also assume that these measurements, and also from different landmarks, are independent of each other. Therefore, a suitable rule for updating particles' weights follows in (8).

$$w_t^{[m]} = w_{t-1}^{[m]} \prod_j \exp\left[-\frac{(d_j - \hat{d_j}^{[m]})^2}{2\sigma_d^2} \right] \exp\left[-\frac{(\psi_j - \hat{\psi_j}^{[m]})^2}{2\sigma_\psi^2} \right] \tag{8}$$

In this model, d_j and ψ_j are the measured horizontal distance and horizontal angle between the robot and the j-th landmark, and $\hat{d_j}^{[m]}$ and $\hat{\psi_j}^{[m]}$ are the calculated expected horizontal distance and horizontal angle between the j-th landmark and the robot in the m-th particle position. Again, the variances were determined empirically, starting from the given ones in Simspark's documentation [7], these values were manually tuned comparing the measurement prediction and the real model.

Note that weights update, and therefore the resample and particles reset steps, are only performed when vision update is detected. Data from vision is reported by the server every 3 simulation cycles (0.06 sec), meanwhile the estimation process consider only the update from the motion model, happening each cycle.

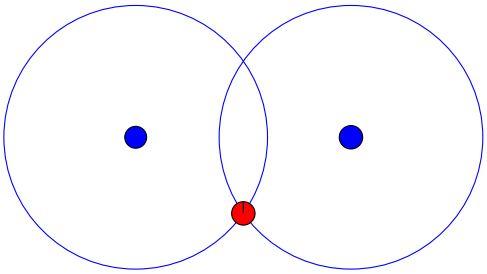

Figure 1. Sensor reset by landmark observation

Table 1. PARAMETERS

ObsQualities		Observation		Motion	
α_{fast}	0.1	σ_d	0.31	σ_x	0.01
α_{slow}	0.01	σ_ψ	0.03	σ_y	0.01
				σ_ψ	0.005

The particles' reset based on sensor information only happens when more than 1 landmark is detected, then two of them are randomly chosen. Observation noise is added in each measurement for both landmarks, in order to better spread the particles. In the cases of intersection of the circles surrounding each landmark by the calculated distances, the particle is resetted to the positions of these intersections (see Figure 1). Normally, only one intersection is within the field limits, and therefore the particle is resetted to this position, otherwise the reset position is drawn between those two. Note also that, as the landmark's pair is randomly chosen among the detected ones, an approximately equal number of particles is resetted for each pair of detected landmarks.

The parameters values used in this work are presented in Table 1. They were empirically determined by manual tweaking, however for future work we expect to determine them using experiments or optimization techniques.

4. Simulation and Results

In the results that are going to be presented, all distances and positions are measured in m and angles in rad. Two simulations were prepared to assess the presented algorithm performance. In the first one there is only horizontal linear velocity, $v_x = 0.2m/s$, calculating the root square error of robot position and orientation in the field, see Figures 3 and 4, and in the second simulation there is only angular velocity, $v_\psi = 0.2rad/s$, thus calculating the RSE in this case, see Figures 5 and 6.

We can observe the moment when the robot passed the middle field line during simulation 1 and suffered the *kidnapping* (10 s), having its position resetted. In that instant, the error sharply increased, however is immediately controlled again. This event only happens in the first simulation for the x coordinate, and also for the y coordinate in lesser extend.

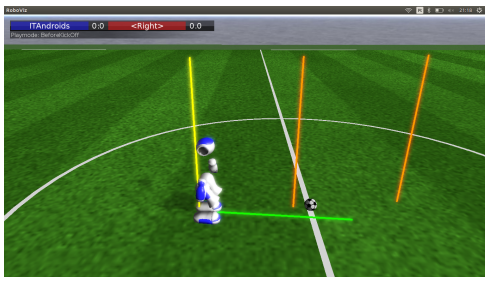

Figure 2. Localization simulation test

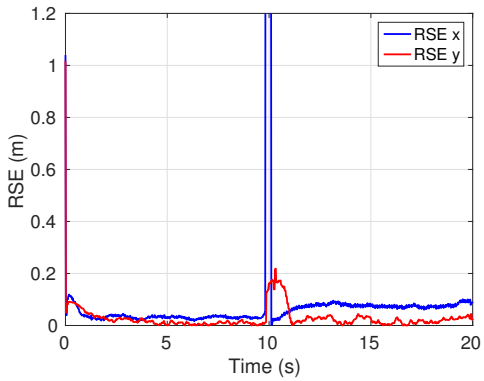

Figure 3. Position RSE - Simulation 1

Table 2. RESULTS- SIMULATION 1

Mean x Position Error	0.0881 m
Mean y Position Error	0.0283 m
Mean Orientation Error	0.0055 rad
Mean Processing Time	4.9×10^{-5} s

The initial state was generated by a uniform distribution of particles throughout the field. The simulation consists of 1000 iterations for the given walking command, corresponding each to a server simulation cycle period of 0.02s. The algorithm makes use of constant 400 particles, moreover the computed mean error and mean computational time follow in Tables 2 and 3. The simulations were run in a Intel Core i7-4720HQ processor, with 2.6Ghz of clock.

Figure 2 illustrates the procedure, showing yellow and green bars which indicate, respectively, the estimated robot position and orientation.

In figures 3 to 6 presented, we expected small oscillations above the zero, with the exception of figure 1, in which happens a kidnapping and therefore a single big oscillation in x. We can also observe that all root mean square errors are in the same order of magnitude, however, in coordinates in which there is velocity, or substantial displacements, the error is naturally bigger, since we have more moving steps (each cycle) than filtering steps (each 3 cycles). Besides bigger mean errors, we also expected bigger stress in oscillations for these cases.

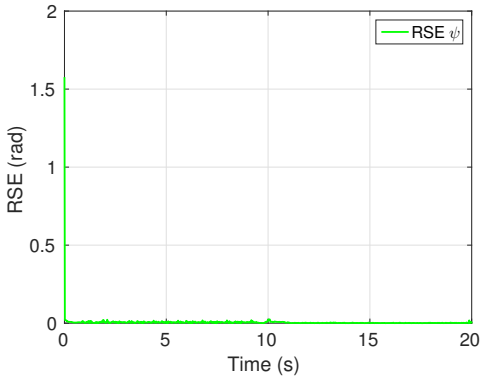

Figure 4. Orientation RSE - Simulation 1

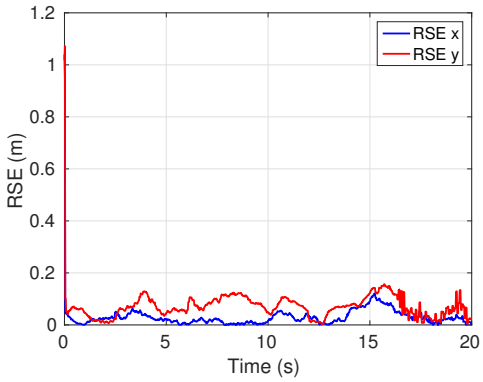

Figure 5. Position RSE - Simulation 2

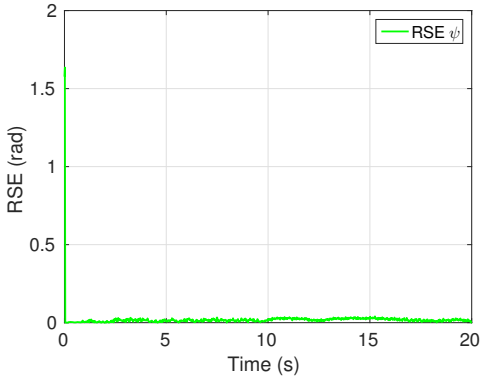

Figure 6. Orientation RSE - Simulation 2

Table 3. RESULTS - SIMULATION 2

Mean x Position Error	$0.0302\ m$
Mean y Position Error	$0.0714\ m$
Mean Orientation Error	$0.0196\ rad$
Mean Processing Time	4.7×10^{-5}

5. Conclusion

Analysing the applied algorithm and the results shown in Sec. 5, we might claim that the Monte Carlo Localization turned to be a quite simple algorithm to implement in this particular case, when compared to other non-linear extensions of the Kalman Filter, and also presented very efficient results, concerning both estimating precision and computational time, even though the fame for this last shortcoming. Furthermore, its augmented version with sensor reset was very successful in dealing with the kidnapping problem.

Future plans for ITAndroids team include the enhancement of this localization work, for instance, by proposing a method for evaluating the covariances of the model described in Sec. 3. Moreover, we seek the development of a tracking system for adversary robots and the ball.

One of the reasons for the development of this specific technique in RoboCup Soccer 3D League was the goal to lately integrate this work with our RoboCup Humanoid Kid-Size project. Therefore, we should also aim to perform some adaptations, since for the Kid-Size there are ambiguous landmarks in the field and then it will be necessary to extract robot's pose from multimodal distributions, for example.

Acknowledgment

Thanks to ITAndroids for the knowledge provided, and for the whole work in a big project, which involves contributions from all members in a big team. Thanks also to Radix for the financial aid which sponsor our project.

References

[1] Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, pages 96-112, 250-261 (2005)

[2] Maskell, S., Gordon, N. 2001. A tutorial on Particle Filters for On-line Nonlinear/Non-Gaussian Bayesian Tracking. In QinetiQ Ltd 2001

[3] Dellaert, F., Fox, D., Burgard, W., Thrun, S. 1999. Monte Carlo localization for mobile robots. In ICRA.

[4] Carpenter, J.; Clifford, P.; and Fernhead, P. 1997. An improved particle filter for non-linear problems. TR, Dept. of Statistics, Univ. of Oxford.

[5] Gutmann, J.-S., and Fox, D. 2002. An experimental comparison of localization methods continued. In Proc. of IROS.

[6] Lenser, S., and Veloso, M. 2000. Sensor resetting localization for poorly modeled mobile robots. In Proc. of ICRA.

[7] Simspark Documentation. [Online]. Available: http://simspark.sourceforge.net/wiki/index.php/Perceptors. Consulted on January 2016

[8] Burchardt, A., Laue, T., and Röfer, T. 2010. Optimizing particle filter parameters for self-localization. In RoboCup 2010: Robot Soccer World Cup XIV, Lecture Notes in AI