Improved Mixture Representation in Real-Time Particle Filters for Robot Localization

Dario Lodi Rizzini* Stefano Caselli*
*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma, Parma, Italy

***Abstract*— Monte Carlo methods have been successfully adopted for robot localization thanks to their flexibility in distribution representation. However, these techniques are computationally expensive and can hardly perform at the incoming sensor data rate, when computation resources are limited. The Real-Time Particle Filter (RTPF) is an algorithmic solution conceived to make execution of a particle filter iteration feasible within time constraints by means of a mixture representation for the set of samples. RTPF requires an optimal balance of the contribution of each set to the mixture, whose computation, unfortunately, is quite difficult. In this paper, we provide a formal discussion of mixture representation by considering the weight mixture. We illustrate a novel solution for computing the mixture parameters based on the notion of effective sample size. This solution is less prone to numerical instability. Finally, we compare the proposed approach with the original RTPF algorithm through simulation tests and experiments.**

***Index Terms*— Robot Localization, Real-Time Particle Filter, Mixture of posterior**

I. Introduction

Global localization is the problem of estimating robot pose, i.e. its position and orientation, with respect to an external reference frame. The robot has at its disposal a map of the environment, its motion information and its sensor observations. Bayesian filtering is a general probabilistic paradigm to arrange motion and sensor data in order to achieve a solution in the form of distribution of a random variable. While parametric bayesian filters represent state as a continuous function (see e.g. [8, 1]), in *Monte Carlo localization* (MCL) [4] the robot belief about its current position is given by a set of samples drawn from a proposal distribution and by importance weights that measure the discrepancy of each sample from the correct distribution. MCL inherits advantages of *sequential importance sampling with resampling* (SIR) techniques: it allows flexibility in representation of the posterior, which usually does not have a given parametric model, and limits linearization errors in motion and sensor model equations, which often lead to poor performance and divergence of filter.

Unfortunately, particle filter (PF) complexity and performance both depend on the number of samples: in global localization a high density of samples helps to discover and to converge towards the correct localization hypothesis. However, for each additional sample a prediction, a correction and a resampling step are performed. Furthermore, localization performance also depends on sensor information, which could be acquired at a rate higher than the filter update rate. Possible solutions to this mismatch between sensing rate and processing time include reduction of the number of samples, e.g. adapting the size of the mixture [3], or of the number of observations, i.e. discarding sensor data.

The *Real-Time Particle Filter* [6, 7] provides a tradeoff between time constraints related to sensor management and filter performance. Samples are partitioned into subsets among observations over an *estimation window*. The size of each partitioned subset is chosen so that a particle filter iteration can be performed before a new observation is acquired. The difference with standard PF with smaller sample set lies in the representation of the posterior as a mixture of samples: at the end of an estimation window the distribution consists of the samples from each subset of the window. Mixture weights determine how each partition set contributes to the posterior and are computed in order to minimize the approximation error of the mixture distribution. However, the original proposal for computation of mixture weights, based on minimization of Kullback-Leibler (KL) divergence [6, 7], is prone to bias problems and numerical instability arising from the need to perform a numerical gradient descent.

In this paper, we provide two main contributions: a formal analysis for the evolution of mixture of posterior in RTPF and a novel solution for the computation of mixture weights. Each partition set posterior consists of samples, which are drawn from motion model as proposal on the estimation windows and whose importance weight depends only on a single observation. Since the correction step is performed at different time instants for each partition set, differences among partition posteriors introduce a bias in estimation. We show that the bias is an outcome of prominence of partition set that minimizes KL-divergence and has poor *effective sample size* simultaneously. We then present an improved approach for the computation of mixture weights based on *effective sample size* of the partition sets.

The paper is organized as follows. After outlining RTPF in section II, a formal description of the bias problem and the proposed novel approach are illustrated in section III. Simulations and experiments are reported in section IV and compared with the original RTPF. Finally, conclusion remarks are given in section V.

II. Real-Time Particle Filters

In particle filters, updating the particles used to represent the probability density function (potentially a large number) usually requires a time which is a multiple of the cycle of sensor information arrival. Naive approaches, yet often adopted, include discarding observations arriving during the

update of the sample set, aggregating multiple observations into a single one, and halting the generation of new samples upon a new observation arrival [7]. These approaches can affect filter convergence, as either they loose valuable sensor information, or they result in inefficient choices in algorithm parameters.

An advanced approach dealing with such situations is the Real-Time Particle Filters (RTPF) [6, 7], which will be briefly described in the following. Consider k observations. The key idea of the Real-Time Particle Filter is to distribute the samples in sets, each one associated with one of the k observations. The distribution representing the system state within an estimation window will be defined as a *mixture* of the k sample sets as shown in Figure 1. At the end of each estimation window,

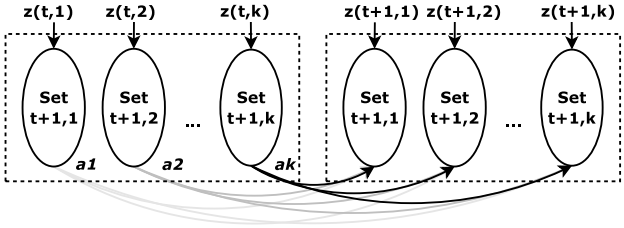

Fig. 1. RTPF operation: samples are distributed in sets, associated with the observations. The distribution is a mixture of the sample sets based on weights α_i (labeled a_i in figure).

the weights of the mixture belief are determined by RTPF based on the associated observations in order to minimize the approximation error relative to the optimal filter process. The *optimal belief* could be obtained with enough computational resources by computing the whole set of samples for each observation. Formally:

$$Bel_{opt}(x_{t_k}) \propto \int \dots \int \prod_{i=1}^{k} p(z_{t_i}|x_{t_i}) \cdot p(x_{t_i}|x_{t_{i-1}}, u_{t_{i-1}}) \cdot Bel(x_{t_0}) dx_{t_0} \cdots dx_{t_{k-1}} \quad (1)$$

where $Bel(x_{t_0})$ is the belief generated in the previous estimation window, and z_{t_i}, u_{t_i}, x_{t_i} are, respectively, the observation, the control information, and the state for the $i-th$ interval.

Within the RTPF framework, the *belief* for the $i-th$ set can be expressed, similarly, as:

$$Bel_i(x_{t_k}) \propto \int \dots \int p(z_{t_i}|x_{t_i}) \cdot \prod_{j=1}^{k} p(x_{t_j}|x_{t_{j-1}}, u_{t_{j-1}}) \cdot Bel(x_{t_0}) dx_{t_0} \dots dx_{t_{k-1}} \quad (2)$$

containing only observation-free trajectories, since the only feedback is based on the observation z_{t_i}, sensor data available at time t_i. The weighted sum of the k believes belonging to an estimation window results in an approximation of the optimal belief:

$$Bel_{mix}(x_{t_k}|\alpha) \propto \sum_{i=1}^{k} \alpha_i Bel_i(x_{t_k}) \quad (3)$$

An open problem is how to define the optimal mixture weights minimizing the difference between the $Bel_{opt}(x_{t_k})$ and $Bel_{mix}(x_{t_k}|\alpha)$. In [7], the authors propose to minimize their Kullback-Leibler distance (KLD). This measure of the difference between probability distributions is largely used in information theory [2] and can be expressed as:

$$J(\alpha) = \int Bel_{mix}(x_{t_k}|\alpha) \log \frac{Bel_{mix}(x_{t_k}|\alpha)}{Bel_{opt}(x_{t_k})} dx_{t_k} \quad (4)$$

To optimize the weights of mixture approximation, a gradient descent method is proposed in [7]. Since gradient computation is not possible without knowing the optimal belief, which requires the integration of all observations, the gradient is obtained by Monte Carlo approximation: believes Bel_i share the same trajectories over the estimation windows, so we can use the weights to evaluate both Bel_i (each weight corresponds to an observation) and Bel_{opt} (the weight of a trajectory is the product of the weights associated to this trajectory in each partition). Hence, the gradient is given by the following formula:

$$\frac{\partial J}{\partial \alpha_i} \simeq 1 + Bel_i \log \frac{\sum_{i=1}^{k} \alpha_i Bel_i}{Bel_{opt}} \quad (5)$$

where Bel_i is substituted by the sum of the weights of partition set $i-th$ and Bel_{opt} by the sum of the weights of each trajectory.

Unfortunately, (5) suffers from a *bias problem*, which [7] solve by clustering samples and computing separately the contribution of each cluster to the gradient (5). In the next section, an alternative solution is proposed.

III. An Enhanced RTPF

In this section we provide a formal investigation on the motivation of bias in RTPF estimation in [7], and we propose a new solution for mixture weights computation.

A. Bias in RTPF Mixture

In RTPF, samples belonging to different partition sets are drawn from the same proposal, but their importance weights depend on different observation likelihood functions $p(z_{t_i}|x_{t_i})$, which are computed in different time instants t_i. Hence, the first source of disparity among partition sets is the degree of proposal dispersion during the correction step. A suitable measure of proposal dispersion at iteration t_i is provided by the radius of the ball set $B(\eta_{x_{t_i}}, r) \subseteq \mathbb{R}^d$, which is centered on expected value $\eta_{x_{t_i}}$ and includes a consistent portion of the distribution of x_{t_i}. The probability that a sample falls in $B(\eta_{x_{t_i}}, r)$ can be bound by r and the trace of the covariance matrix $\Sigma_{x_{t_i}}$, since the following Chebychev-like inequality holds:

$$P\left(x_{t_i} \in B(\eta_{x_{t_i}}, r)\right) > 1 - \frac{tr(\Sigma_{x_{t_i}})}{r^2} \quad (6)$$

In the following, the probability of event given by $B(\eta_{x_{t_i}}, r)$ will refer to a proposal density function arrested in t_i:

$$\pi(x_{t_i}) = \int_{\mathbb{R}^{d \times i}} \prod_{j=1}^{i} p(x_{t_i}|x_{t_{i-1}}, u_{t_{i-1}}) \; dx_{t_0} \dots dx_{t_{i-1}} \quad (7)$$

Then, given $0 < \epsilon < 1$, a sample falls in a ball with at least probability ϵ when its radius is larger than the *dispersion radius*:

$$r_{t_i,\epsilon} = \sqrt{tr(\Sigma_{x_{t_i}})/(1-\epsilon)} \tag{8}$$

Parameter $r_{t_i,\epsilon}$ provides a rough estimation for dispersion because only for unimodal PDF the ball $B(\eta_{x_{t_i}}, r_{t_i,\epsilon})$ (briefly B hereafter) limits a region around a local maximum. Furthermore, it is often the case that x_{t_i} is a vector of heterogeneous random variables (e.g. cartesian coordinates and angular values), whose variances are mixed in the trace, with the result that bound (8) largely overestimates the region. However, the dispersion radius is a synthetic value and can be adapted to multimodal distributions after decomposition into a sum of unimodal hypotheses. Empirically, this decomposition is achieved by clustering on samples.

By applying command control and updating robot position, the dispersion radius increases together with the trace of covariance matrix. If G_{t_i} is the Jacobian of motion model computed in $(\eta_{x_{t_i}}, u_{t_i})$, with $G_{t_i}\ G_{t_i}^T \geq 0$ and $tr(G_{t_i}\ G_{t_i}^T) \geq 1$ (hypotheses verified by a standard model like [11]), and $\Sigma_{w_{t_i}}$ is the covariance matrix of additive noise, then

$$tr(\Sigma_{x_{t_{i+1}}}) \approx tr(G_{t_i}\ \Sigma_{x_{t_i}}\ G_{t_i}^T) + tr(\Sigma_{w_{t_i}}) \tag{9}$$

Thus, we conclude that $tr(\Sigma_{x_{t_i}}) \leq tr(\Sigma_{x_{t_{i+1}}})$ and that the dispersion radius increases over the estimation window. A more accurate estimation of how it increases could be obtained with further hypotheses on the motion model, e.g. Lipschitz continuity.

Since the proposal is more and more spread in the estimation window and correction is performed at different times for each partition, we want to investigate how the dispersion affects importance weights. Observation likelihood $w_{t_i}(x) = p(z_{t_i}|x)$ is usually more concentrated than the proposal, sometimes peaked as shown in [5]. We assume that, given a proper $\delta > 0$, region

$$L = \{x \in B \mid w_{t_i}(x) > \delta\} \tag{10}$$

covers a consistent portion of $w_{t_i}(x)$. Thus, observation likelihood is bound in L by $M = \sup_{x \in L}\ w_{t_i}(x) < \infty$ (envelope condition) and in $B \setminus L$ by δ. Hence, $w_{t_i}(x) \leq \lambda(x)$ over B, with

$$\lambda(x) = \begin{cases} M & x \in L \\ \delta & else \end{cases} \tag{11}$$

The bounding function $\lambda(x)$ and set L are defined on ball B, and in the following we will restrict the sampling domain to B using $\pi(x_{t_i}|x_{t_i} \in B)$ as proposal. This assumption allows us to consider the dispersion radius in the following discussion. Moreover, this approximation is not so rough when ϵ is close to 1.

The *effective sample size* [9] is a measure of the efficiency of a set of samples in the representation of a target posterior:

$$n_{eff_{t_i}} = \frac{1}{\sum_{s=1}^{N} \tilde{w}_{t_i}^2(x_{t_i}^{(s)})} \tag{12}$$

$$= \frac{\left(\sum_{s=1}^{N} w_{t_i}(x_{t_i}^{(s)})\right)^2}{\sum_{s=1}^{N} w_{t_i}^2(x_{t_i}^{(s)})} \tag{13}$$

The above expression is achieved by substituting normalized weights $\tilde{w}_{t_i}(x)$ with their expression. Maximizing the effective sample size is equivalent to minimizing the variance of the weights: it is easy to show with Jensen inequality that n_{eff} is bounded by the number of samples N, which is obtained when each weight is equal to 1 and the variance is small. Bounds on observation likelihood allow an approximation of expected values of weight and square weight:

$$E_\pi\left[w_{t_i}(x_{t_i})|x_{t_i} \in B\right] \leq M\ H_L + \delta\ H_{B\setminus L} \tag{14}$$

$$E_\pi\left[w_{t_i}^2(x_{t_i})|x_{t_i} \in B\right] \leq M^2\ H_L + \delta^2\ H_{B\setminus L} \tag{15}$$

where $H_L = E_\pi[I_L(x)]$ and $H_{B\setminus L} = E_\pi[I_{B\setminus L}(x)]$ are the *visit histograms* of bins L and $B \setminus L$ respectively; in our notation $I_D(x)$ is the indicator variable with value 1 when x falls in D, zero otherwise. Equations (14) and (15) can be used to approximate numerator and denominator of (13):

$$n_{eff_{t_i}} \approx N\ \frac{\left(M\ H_L + \delta\ H_{B\setminus L}\right)^2}{M^2\ H_L + \delta^2\ H_{B\setminus L}} \tag{16}$$

$$\approx N\ \left(H_{B\setminus L} + 2\ \frac{M}{\delta} H_L + \frac{M^2\ H_L^2}{\delta^2\ H_{B\setminus L}}\right) \tag{17}$$

The approximation given by (17) follows from the assumption that $H_L/H_{B\setminus L} << (\delta/M)^2$. When dispersion is large, proposal can be considered almost constant on region L and its visit histogram H_L decreases proportionally with the ratio of hypervolumes of L and $B \setminus L$: $H_L \propto 1/r_{t_i,\epsilon}^d$ in d-dimensional space. Thus, the last partition sets in the estimation window, i.e. those approximating better the distribution at the end of the estimation window, have a spread proposal and are represented by few effective samples, as shown by the trend of (17). From difference between effective sample size and KLD reduction, the bias in estimation follows.

The solution proposed in [7] mitigates the effects of bias by considering the multimodal structure of samples distribution in KL-distance gradient estimation. The estimation of gradient given by (5) ignores samples dispersion in different *bins*. Formally, gradient (5) is the result of underestimation of KL-divergence: call $Bel_{mix}(C_j)$ and $Bel_{opt}(C_j)$ the mixture and optimal histograms for cluster C_j respectively; from the convexity of KLD [2], Jensen inequality holds

$$KL(\sum_{j=1}^{M} Bel_{mix}(C_j) \parallel \sum_{j=1}^{M} Bel_{opt}(C_j)) \leq \sum_{j=1}^{M} KL\left(Bel_{mix}(C_j) \parallel Bel_{opt}(C_j)\right) \tag{18}$$

Gradient estimation based on the second term of inequality (18) is better than the previous one based on the first term, but no optimality can be claimed since bin subdivision is empirical and gradient descent approaches easily incur in local minima problems. Furthermore, even if cluster detection is usually performed in PF to group localization hypotheses and no additional computational load is required, sample management is not at all straightforward.

B. Alternative computation of Mixture Weights

This section proposes an alternative criterion to compute the values of the weights for the mixture belief. Instead of trying to reduce the Kullback-Leibler divergence, our approach considers mixture weights as the assigned measure of relative importance of partitions that is transformed by processing at the end of estimation window. RTPF prior distribution is the result of two main steps: resampling of samples and propagation of trajectories along previous estimation window. The effect of resampling is the concentration of previous estimation window samples in a unique distribution carrying information from each observation. Conversely, the trajectories update given by odometry and observation spreads the particles on partition sets.

Our attempt is to build a linear map modeling the change of relative importance, i.e. mixture weights α, due to resampling and propagation of samples. This map should depend on sample weights. Let w_{ij} be the weight of the $i-th$ sample (or trajectory) of the $j-th$ partition set. Then, the *weight partition matrix* is given by

$$W = \begin{bmatrix} w_{11} & \dots & w_{1k} \\ \dots & & \dots \\ w_{N_p 1} & \dots & w_{N_p k} \end{bmatrix} \tag{19}$$

The weights on a row of this matrix trace the history of a trajectory on the estimation window; a group of values along a column depicts a partition handling sensor data in a given time. Resampling and trajectory propagation steps can be shaped using matrix W and mixture weights α.

- *Resampling*. The effect of resampling is the concentration of each trajectory in a unique sample whose weight is the weighted mean of the weights of the trajectory. In formula, the vector of trajectory weights is given by $t = W \cdot \alpha$.
- *Propagation*. Projecting a sample along a trajectory is equivalent to the computation of the weight of the sample (i.e., the posterior) for each set, given the proper sensor information. Again, matrix W gives an estimation of the weight. Trajectories projection can thus be done with a simple matrix product

$$\hat{\alpha} = W^T \cdot t = W^T \; W \cdot \alpha \tag{20}$$

  Vector $\hat{\alpha}$ is a measure of the relative amount of importance of each partition set after resampling and propagation depending on the choice of coefficient α. Hence, $\hat{\alpha}$ is the new coefficient vector for the new mixture of believes.

Some remarks can be made about the matrix $V = W^T \; W$ in (20). First, since we assume $w_{ij} > 0$, V is a symmetric and positive semi-definite (SPSD) matrix. Moreover, each element j on the main diagonal is the inverse of the effective sample size of set j. The effective sample size is a measure of the efficiency of importance sampling on each of the partition sets. Therefore, the off-diagonal elements of V correspond to a sort of importance covariances among two partition sets. Thus we will refer to this matrix as *weights matrix*.

Hence, a criterion to compute the mixture weights consists of choosing the vector α that is left unchanged by map (20) except for scale. Since (20) depends on square of sample weights, resulting mixture weights reflects the importance of each partition set according to the effective sample size. The vector is thus obtained by searching for an eigenvector of matrix V. To achieve better stability we choose the eigenvector corresponding to the largest eigenvalue. The eigenvector can be computed using the power method or the inverse power method. This criterion can be interpreted as an effort to balance the effective number of samples keeping the proportion among different partition sets.

IV. Results

We report RTPF performance evaluation both in simulated environments and using experimental data collected by navigating a robot in a known environment. These results have been obtained exploiting the localization system described in [10]. Tests have compared the effectiveness of the two solutions previously described for computation of RTPF mixture weights by assessing their impact on localization performance.

A. Simulation

Several tests were performed in the environments shown in figures 2 and 3. They correspond to the main ground floor hallway in the Computer Engineering Department of the University of Parma (figure 2) and to the hallway of the Department of Computer Science and Engineering of the University of Washington (figure 3, map adapted from [7]). These environments allow verification of RTPF correctness

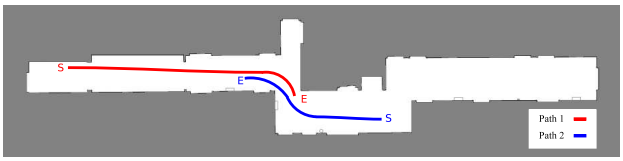

Fig. 2. Map 1 – Hallway and simulated paths in the Computer Engineering Department, University of Parma.

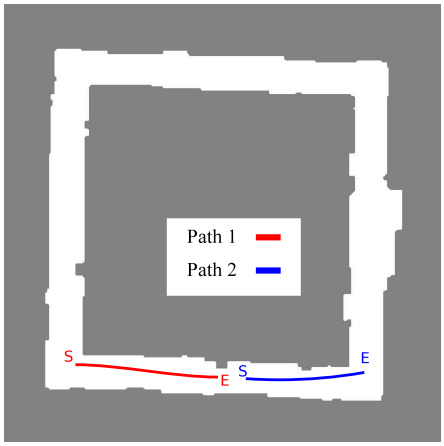

Fig. 3. Map 2 – Hallway and simulated paths in the Department of Computer Science and Engineering, University of Washington.

while coping with several symmetric features, which may

cause ambiguities in the choice of correct localization hypotheses. The environment of figure 3 had been exploited in [7] to verify RTPF correctness and has therefore been considered as a reference.

In simulation, the map is stored as a grid with a given resolution (0.20 m) and is used both to create simulated observations and to compute importance weights in correction steps. Data provided to the localizer consist of a sequence of laser scans and measurements: scanned ranges are obtained by ray tracing a beam on the discretized map. The measurement model is also based on ray tracing according to standard beam models for laser scanner [11]. In our tests we have used only three laser beams measuring distances to left, right and frontal obstacles; such poor sensor data stress the role of algorithm instead of sensor data. A gaussian additive noise was added to both range beams and robot movements representing environment inputs and robot state in simulation. Thus simulation tests are performed in an environment known in detail and are best suited for comparing performance between algorithms. The task of the robot is to achieve localization while moving in the environments of figures 2 and 3 along assigned trajectories. Simulated trajectories, labeled as Path 1 and Path 2 in figures 2 and 3, correspond to lengths of approximately 5 to 8 m.

Localization algorithms investigated are the original steepest descent-based one (RTPF-Grad) and the proposed RTPF based on the effective number of samples (RTPF-Eig). During these tests the partition set size was 1000 samples.

A summary of simulation results is reported in figures 4 and 5, where curves show the localization error for the two algorithms at each iteration by considering convergence to the maximal hypothesis. For both curves, each value is obtained by averaging the distances of the estimated pose from the real pose over 10 trials where localization eventually converged to the correct hypothesis within the maximum number of iterations (set to 40). For both algorithms there were also a few instances where localization did not converge to the correct hypothesis within the length of the path, although the correct hypothesis was the second best. These unsuccessful experiments were approximately 10% of all simulated localization trials. We did not verify whether the robot would eventually recover its correct pose in the environment with further navigation.

On the average, the two versions of the RTPF-based localizer converge to some few hypotheses after three iterations, and the common samples distribution is multi-modal. Hence, cluster search leads to few hypotheses with different weight. In our tests a hypothesis close to the correct robot pose always exists, and when this hypothesis prevails there is a sudden change in localization error, as shown in figures 4 and 5. Convergence is helped by recognizable features, e.g. the shape of scans, but when the environment is symmetric it can be difficult to reach, especially with limited or noisy sensoriality. Of course, the mean error in figures 4 and 5 does not correspond to any of the simulated trials; rather, it is the result of averaging trials with quick convergence and trials where the convergence requires many more iterations.

Figure 6 provides an alternative view of the same data, as curves show the percentage of simulation trials converging to

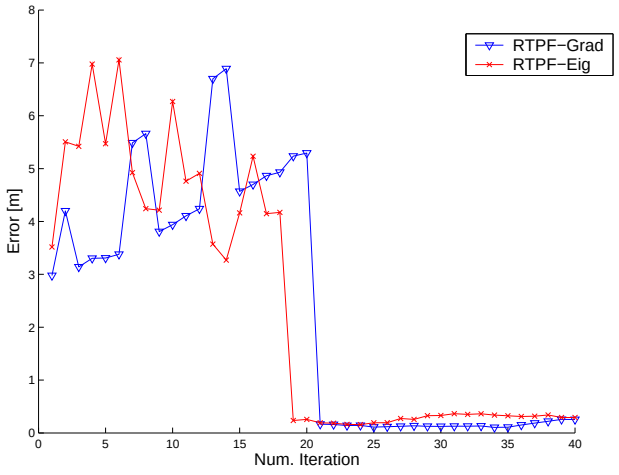

Fig. 4. Performance of the two RTPF versions in the simulated environment of Map 1. The *x*-axis represents the iterations of the algorithm. The *y*-axis shows the average error distance of the estimated pose from robot pose.

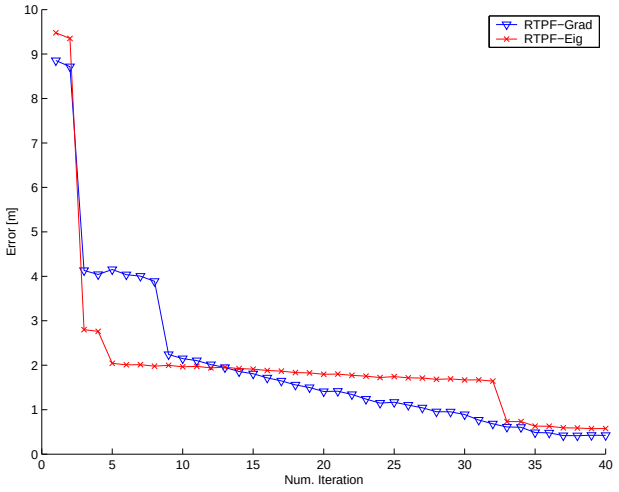

Fig. 5. Performance of the two RTPF versions in the simulated environment of Map 2. The *x*-axis represents the iterations of the algorithm. The *y*-axis shows the average error distance of the estimated pose from robot pose.

the correct hypothesis (i.e. with localization error less than 1.5 m) at each iteration. In a few simulations, the correct robot pose is recovered only after about 20 or 30 iterations, i.e. after sensing map features that increase the weight of the correct samples.

Empirically, for the examined environments RTPF-Eig seems to exhibit a slightly faster convergence, on the average, to the correct localization hypothesis, even though its average error at the last recorded iteration appears somewhat larger.

B. Experiments

Real experiments were run in the environment of figure 2 collecting data with a Nomad 200 mobile robot equipped with a Sick LMS 200 laser scanner. The robot moved along Path 1 for about 5 m, from the left end of the hallway in steps of about $15 - 20$ cm and reading three laser beams from each scan in the same way of the simulation tests.

To assess the consistency of the localizer's output in an

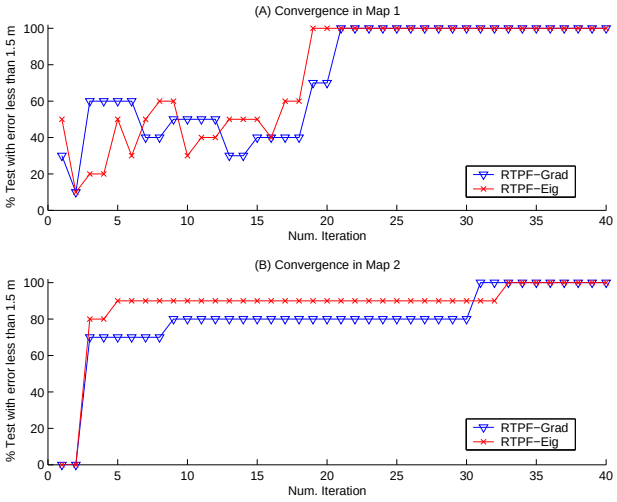

Fig. 6. Percentage of simulation trials converged to the correct hypothesis, i.e. with localization error less than 1.5 m, during iterations for Map 1 (a) and Map 2 (b).

automated way, we compared the robot pose computed by the localizer (using the RTPF-Eig algorithm) with the one provided by an independent localization methodology. To this purpose, some visual landmarks were placed in the environment and on the mobile robot, and a vision system was exploited to triangulate the robot position based on these landmarks. The vision system provided an independent, coarse estimate of the robot pose at any step, and hence allowed to establish convergence of the RTPF-based localizer. The two localization estimates were computed concurrently at each location and stored by the robot.

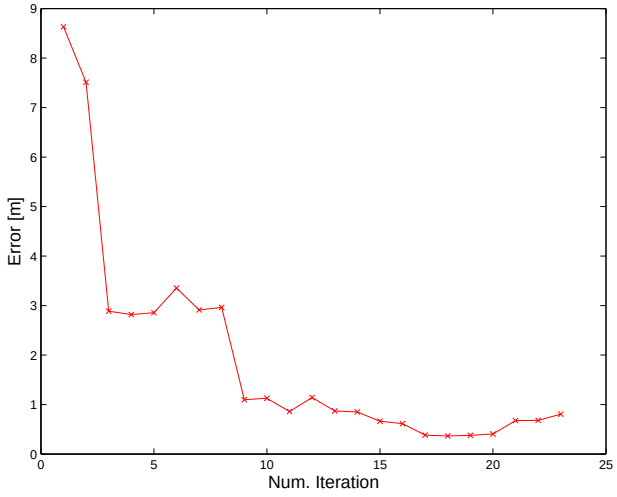

Fig. 7. Performance of RTPF-Eig using real data collected in the hallway of Map 1.

Figure 7 shows the results of 10 tests of RTPF-Eig over about 20 iterations. In these real experiments RTPF-Eig achieves localization to the correct hypothesis very fast in most cases. After convergence, the maximum distance between RTPF-based and vision based estimates is about 70 cm, due to the compound error of the two systems. In real experiments in Map 1 localization was always successful within the length of the path. Moreover, results in figure 7 show that localization to the correct hypothesis was always reached in less than 10 iterations.

V. Conclusion

In this paper, we have presented a formal discussion of computation of mixture weights in RTPFs, along with an improved approach overcoming potential problems associated with the existing technique. The method proposed in this paper computes mixture weights as the eigenvector of a matrix and thus avoids gradient descent, possibly prone to numerical instability. The method provides a balance of the effective sample size of partition sets on an estimation window.

The proposed approach has been implemented in a RTPF for localization with a mobile robot equipped with a laser range scanner, and evaluated in both simulation tests and real experiments. In two simulation environments, the new approach has achieved a localization performance similar to the original KLD-based algorithm, while avoiding the potential problems associated with gradient search methods. In real experiments with the mobile robot, the modified RTPF-based localization system has proven very effective, yielding correct localization within a small number of filter iterations.

In addition to the anecdotal evidence reported in this paper, further experimental work is required to assess the relative merit of the improved RTPF over the original approach. We also plan to investigate application of the modified RTPF to different estimation problems, beside localization.

Acknowledgments

This work was supported by ”LARER – Laboratorio di Automazione della Regione Emilia-Romagna.”

References

[1] K.O. Arras, H.F. Castellanos, and R. Siegwart. Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints. *IEEE Int. Conf. on Robotics and Automation*, 2:1371–1377, 2002.

[2] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[3] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *Int. J. of Robotics Research*, 22(12):985–1003, 2003.

[4] D. Fox, W. Burgard, and S. Thrun. Monte Carlo Localization: Efficient position estimation for mobile robots. *Proc. of the National Conference on Artificial Intelligence*, 1999.

[5] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Trans. on Robotics*, 23(1):34–46, February 2007.

[6] C. Kwok, D. Fox, and M. Meilă. Adaptive real-time particle filters for robot localization. *IEEE Int. Conf. on Robotics and Automation*, 2:2836–2841, 2003.

[7] C. Kwok, D. Fox, and M. Meilă. Real-time particle filters. *Proc. of the IEEE*, 92(3):469–484, March 2004.

[8] J.J. Leonard and H.F. Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. *IEEE Int. Conf. on Robotics and Automation*, 1991.

[9] J. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, June 1996.

[10] D. Lodi Rizzini, F. Monica, M. Reggiani, and S. Caselli. Addressing complexity issues in a real-time particle filter for robot localization. *Intl. Conf. on Informatics in Control, Automation and Robotics*, 2007.

[11] S. Thrun, W. Burgard, and D. Fox. *Probabibilistic Robotics*. MIT Press, Cambridge, MA, 2005.