21

An Improved Real-Time Particle Filter for Robot Localization

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

1. Introduction

Robot localization is the problem of estimating robot coordinates with respect to an external reference frame. In the common formulation of the localization problem, the robot is given a map of its environment, and to localize itself relative to this map it needs to consult its sensor data. The effectiveness of a solution to the localization problem in an unstructured environment strongly depends on how it copes with the uncertainty affecting robot perception.

The *probabilistic robotics* paradigm provides statistical techniques for representing information and making decision, along with a unifying mathematical framework for probabilistic algorithms based on Bayes rule (Thrun et al., 2005). For this reason, bayesian filtering has become the prevailing approach in recent works on localization (Elinas & Little, 2005; Sridharan et al., 2005; Hester & Stone, 2008).

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 state random variables. Bayesian filters differ in the representation of the probability density function (PDF) of state. For example, the resulting estimation of Gaussian filters (Kalman Filter, Extended Kalman Filter) (Leonard & Durrant-Whyte, 1991; Arras et al., 2002) is expressed in the form of a continuous parametric function, while the state posterior is decomposed in discrete elements for nonparametric filters.

The main nonparametric algorithm is called *Particle Filter* (PF) (Fox et al., 1999) and relies on importance sampling (Doucet et al., 2001). With importance sampling, the probability density of the robot pose is approximated by a set of samples drawn from a proposal distribution, and an importance weight measures the distance of each sample from the correct estimation.

The nonparametric approach has the advantage of providing a better approximation of the posterior when a parametric model does not exist or changes during iteration, e.g. in initialization or when environment symmetries determine a multi-modal PDF. Even if techniques like Multi-Hypothesis Tracking (Arras et al., 2002) attempt to manage multi-modal distributions, particle filters are more efficient and can represent all kinds of PDFs, including uniform distributions. Moreover, particle filters limit errors due to the linearization of model equations that can lead to poor performance and divergence of the

Source: Robotics, Automation and Control, Book edited by: Pavla Pecherková, Miroslav Flídr and Jindřich Duník,
ISBN 978-953-7619-18-3, pp. 494, October 2008, I-Tech, Vienna, Austria

filter for highly nonlinear problems. Unfortunately, particle filters suffer from computational complexity due to the large number of discrete samples of the posterior: for each sample a pose update, a correction and a resample step are performed. Since localization can be performed slowly with respect to the usual movement and tasks of the robot, it would be conceivable to perform localization over a large time interval. Therefore, there have been attempts to adapt the number of samples (Fox, 2003). However, during an excessive time interval uncertainty increases and many useful observations are dropped; a proper interval to complete a particle filter iteration should be approximately equal to the rate of incoming data. A trade-off must therefore be reached between time constraints imposed by the need of collecting sensor data incoming with a given rate and the number of samples determining the accuracy of the representation of localization hypotheses. Performance depends both on the number of integrated observations and on the number of samples.

The *Real-Time Particle Filter* (RTPF) (Kwok et al., 2004) is a variant of a standard particle filter addressing this problem. 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.

While RTPF represents a remarkable step toward a viable particle filter-based localizer, there are a few issues to be addressed in developing an effective implementation. RTPF convergence is prone to bias problem and to some numerical instability in the computation of the mixture weights arising from the need to perform a numerical gradient descent. Furthermore, even adopting RTPF as the basic architecture, the design of a flexible and customizable particle filter remains a challenging task. For example, life cycle of samples extends beyond a single iteration and covers an estimation window in which mixture posterior computation is completed. This extended life cycle of samples impacts over software design. Moreover, RTPF addresses observations management and derived constraints. A good implementation should be adaptable to a variety of sensors.

In this chapter, we describe the application of RTPF to robot localization and provide three additional contributions: a formal analysis for the evolution of mixture of posterior in RTPF, a novel solution for the computation of mixture weights yielding improved stability and convergence, and a discussion of the design issues arising in developing a RTPF-based robot localization system.

The algorithm described in (Kwok et al., 2004) computes mixture weights by minimizing the Kullback-Leibler (KL) distance between the mixture distribution and the theoretically-correct one. Unfortunately, this criterion tends to promote partition sets of the estimation window that provide a poor representation of the distribution of robot state. In particular, we show that KL criterion favours sets with low *effective sample size* (Liu, 1996) and leads to a bias in the estimation. As an alternative way to compute mixture weights, we define a *weights matrix*, whose elements are related to the effective sample size. The mixture weight vector is then computed as an eigenvector of this matrix. This solution is more robust and less prone to numerical instability. Finally, we propose the design of a library that takes care

of efficient life cycle of samples and control data, which is different between RTPF and standard particle filter, and supports multiple motion and sensor models. This flexibility is achieved by applying generic programming techniques and a policy pattern. Moreover, differing from other particle filter implementations (e.g., CARMEN (Montemerlo et al., 2003)), the library is independent from specific control frameworks and toolkits.
The remaining of the chapter is organized as follows. Section 2 contains an overview of RTPF with the original algorithm to compute mixture weights. Section 3 provides a formal description of the bias problem and a novel approach in the computation of mixture weights based on the effective number of samples. This approach simplifies RTPF and tries to avoid spurious numeric convergence of gradient descent methods. Section 4 illustrates design issues connected to RTPF and describes a localization library implementing a highly configurable particle filter localizer. Section 5 presents simulation and experimental results which are reported and compared with the original RTPF performance. Finally, section 6 gives conclusion remarks.

2. 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 (Kwok et al., 2004). These approaches can affect filter convergence, as either they loose valuable sensor information, or they result in inefficient choices in algorithm parameters.

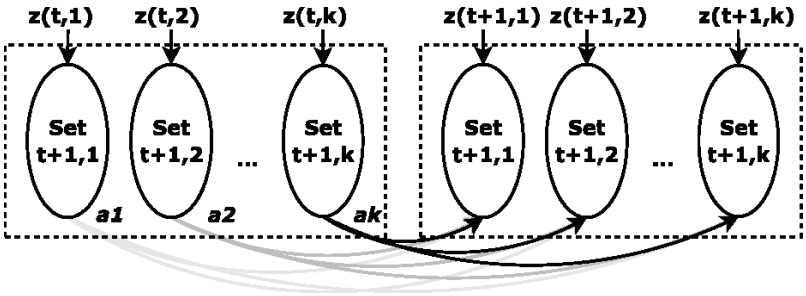

Fig. 1 RTPF operation: samples are distributed in sets, associated with the observations. The distribution is a mixture of the sample sets based on weights α_i (shown as *ai* in figure).

An advanced approach dealing with such situations is the Real-Time Particle Filters (RTPF) (Kwok et al., 2003; Kwok et al., 2004) which is 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 Fig. 1. At the end of each estimation window, 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:

$$Bel_{opt}(x_{tk}) \propto \int \prod_{i=1}^{k} p(z_{ti}|x_{ti}) p(x_{ti}|x_{ti-1},u_{ti}) Bel(x_{t0}) dx_{t0} \ldots dx_{tk-1} \quad (1)$$

where $Bel(x_{t0})$ is the belief generated in the previous estimation window, and z_{ti}, u_{ti}, x_{ti} 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:

$$Bel_{i}(x_{tk}) \propto \int p(z_{ti}|x_{ti}) \prod_{j=1}^{k} p(x_{tj}|x_{tj-1},u_{tj}) Bel(x_{t0}) dx_{t0} \ldots dx_{tk-1} \quad (2)$$

containing only observation-free trajectories, since the only feedback is based on the observation z_{ti}, 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:

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

An open problem is how to define the optimal mixture weights minimizing the difference between the $Bel_{opt}(x_{tk})$ and $Bel_{mix}(x_{tk} \mid \alpha)$. In (Kwok et al., 2004), the authors propose to minimize their Kullback-Leibler distance (KLD). This measure of the difference between probability distributions is largely used in information theory (Cover & Thomas, 1991) and can be expressed as:

$$J(\alpha) \propto \int Bel_{mix}(x_{tk} \mid \alpha) \log \frac{Bel_{mix}(x_{tk} \mid \alpha)}{Bel_{opt}(x_{tk})} dx_{t_{k-1}} \quad (4)$$

To optimize the weights of mixture approximation, a gradient descent method is proposed in (Kwok et al., 2004). 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:

$$\frac{\partial J}{\partial \alpha_i} \cong 1 + \sum_{s=1}^{Np} w_{ti}(x_{ti}^{(s)}) \frac{\sum_{j=1}^{k} \alpha_j w_{tj}(x_{tj}^{(s)})}{\prod_{j=1}^{k} w_{tj}(x_{tj}^{(s)})} \quad (5)$$

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 (Kwok et al., 2004) 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.

3. An enhanced RTPF

In this section we provide a formal investigation on the motivation of bias in RTPF estimation in (Kwok et al., 2004) and we propose a new solution for mixture weights computation.

3.1 A bias in RTPF

In RTPF, samples belonging to different partition sets are drawn from the same proposal, but their importance weights depend on different observation likelihood functions $p(z_{ti} \mid x_{ti})$, 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 $B(\eta_{x_{ti}}, r) \in \Re^d$, which is centered on expected value x_{ti} and includes a consistent portion of the distribution of x_{ti}. The probability that a sample falls in $B(\eta_{x_{ti}}, r)$ can be bound by r and the trace of the covariance matrix $\Sigma_{x_{ti}}$, since the following Chebychev-like inequality holds:

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

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

$$\pi(x_{ti}) = \int_{\Re^{d \times i}} \prod_{j=1}^{i} p(x_{tj} \mid x_{tj-1}, u_{tj}) dx_{t0} \ldots dx_{tk-1} \tag{7}$$

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

$$r_{t_i,\varepsilon} = \sqrt{\frac{\mathrm{tr}(\Sigma_{x_{ti}})}{1-\varepsilon}} \tag{8}$$

Parameter $r_{t_i,\varepsilon}$ provides a rough estimation for dispersion because only for unimodal PDF the ball $B(\eta_{x_{ti}}, r_{t_i,\varepsilon})$ (briefly B hereafter) limits a region around a local maximum. Furthermore, it is often the case that x_{ti} 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 the covariance matrix. If G_{ti} is the Jacobian of motion model computed in (η_{xti}, u_{ti}), with $G_{ti} G_{ti}^T \geq I$ (hypotheses verified by a standard model like (Thrun et al., 2005)), and Σ_{wti} is the covariance matrix of additive noise, then

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

Thus, we conclude that $\mathrm{tr}(\Sigma_{x_{t_i}}) \leq \mathrm{tr}(\Sigma_{x_{t_{i+1}}})$ 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 $w_{t_i}(x) = p(z_{t_i} \mid x)$ is usually more concentrated than the proposal, sometimes peaked as shown in (Grisetti et al., 2007). We assume that, given a proper $\delta > 0$, region

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

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, $w_{t_i}(x) < \lambda(x)$

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

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 $\pi(x_{t_i} \mid x_{t_i} \in B)$ 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 (Liu, 1996) is a measure of the efficiency of a set of samples in the representation of a target posterior:

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

$$= \frac{\left(\sum_{s=1}^{N} \mathrm{w}_{\mathrm{t_j}}(\mathrm{x}_{\mathrm{t_j}}^{(\mathrm{s})})\right)^{2}}{\sum_{s=1}^{N} \mathrm{w}_{\mathrm{t_j}}^{2}(\mathrm{x}_{\mathrm{t_j}}^{(\mathrm{s})})} \tag{13}$$

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:

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

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

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):

$$n_{eff_{ti}} \cong N\frac{(M\cdot H_{L}+\delta\cdot H_{B\setminus L})^{2}}{M^{2}\cdot H_{L}+\delta^{2}\cdot H_{B\setminus L}} \cong \mathrm{N}\left(H_{B\setminus L}+2\frac{M}{\delta}H_{L}+\frac{M^{2}H_{L}^{2}}{\delta^{2}H_{B\setminus L}}\right) \tag{16}$$

The approximation given by (16) follows from the assumption that $H_L / H_{B\setminus L} << (\delta / M)^2$. When dispersion is large, the 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 $B\setminus L$: $H_L \propto 1/r_{ti,\varepsilon}^{d}$ 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 (16). From difference between effective sample size and KLD reduction, the bias in estimation follows.

The solution proposed in (Kwok et al., 2004) 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 (Cover & Thomas, 1991), Jensen inequality holds

$$KL(\sum_{j=1}^{M}Bel_{mix}(C_{j}),\sum_{j=1}^{M}Bel_{opt}(C_{j}))\leq\sum_{j=1}^{M}KL(Bel_{mix}(C_{j}),Bel_{opt}(C_{j})) \tag{17}$$

Gradient estimation based on the second term of inequality (17) is better than the previous one based on the first term which underestimates the distance, 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.

3.2 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 the 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 , due to resampling and propagation of samples. This map should depend on sample weights. Let wi_j be the weight of the $i-th$ sample (or trajectory) of the $j-th$ partition set. Then, the weight partition matrix is given by

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1k} \\ \vdots & & \vdots \\ w_{N_p 1} & \cdots & w_{N_p k} \end{bmatrix} \tag{18}$$

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.

- *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 $t = W\alpha$.
- *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

$$\hat{\alpha} = W^T t = W^T W \alpha \tag{19}$$

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 (19). 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 that is left unchanged by map (19) except for scale. Since (19) depends on square of sample weights, the 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.

Fig. 2 illustrates the differences in mixture weights computed according to the original algorithm (RTPF-Grad) and the proposed variant (RTPF-Eig) with an example. When RTPT-

Eig is used to compute mixture weights, the weights of the last partition sets in the estimation window decrease with the effective sample size of the sets, while they increase with RTPF-Grad. Thus, the proposed criterion takes into account the effectiveness of representation provided by partition sets.

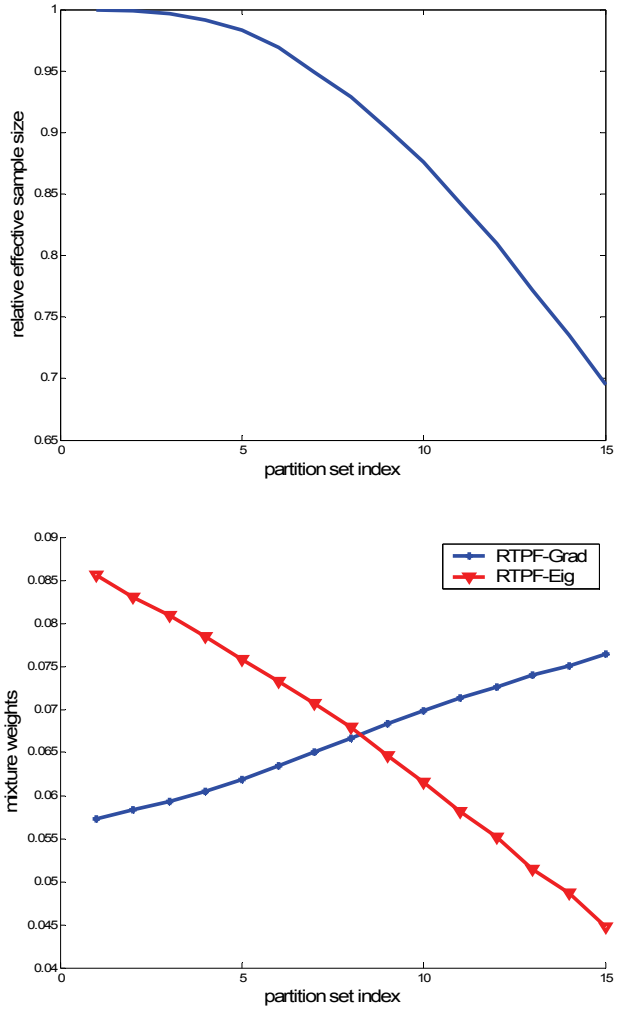

Fig. 2 Effective sample size (top) and mixture weights computed according to the original algorithm and to the proposed variant (bottom) in an estimation window of 15 partitions.

4. Complexity of RTPF implementation

As pointed out in the previous section, updating the whole set of samples can be quite demanding from a computational perspective. Together with advanced algorithms, able to

maximize the use of sensor data, the developer should pay great attention to the implementation issues. Inefficiencies due to poor analysis in object creation/destruction cycles and abstraction levels supporting the polymorphic behaviour can introduce drain of computing time preventing the successful application of the conceived algorithm.
This section describes a library designed to efficiently support the implementation of particle filter localization algorithms. The library aims at providing an efficient yet open infrastructure allowing the users to exploit the provided genericity to integrate their own algorithms. The library has been studied to be easily included in the majority of control systems for autonomous mobile robots. In a functional layer, or controller, with the basic computational threads for robot action and perception, the localization task can be simply configured as a computationally demanding, low priority thread.

4.1 Design of the library

Based on the functional analysis of the localization problem, four main components have been identified: the localizer, the dynamic model of the system, the sensor data model, and the maps. The interaction among these components enables the implementation of the prediction, matching, and resampling phases of particle filter algorithms. Three classes storing basic data information are required: system state, control command, and sensor data. The main component is the localizer implemented by the *Localizer* class, managing the localization cycle through the coordination of the classes composing the library. Listing 1 shows a simplified interface of the *Localizer* class, including just two *update()* methods.
In the prediction phase, *update()* uses the control command executed by the robot (*Control* class) to perform the computation on the *SystemModel* class, representing the dynamic model of the system. In the correction phase, the second *update()* method uses the perception of the environment *(SensorData* class) to update the density function through the evaluation of the weight. In the following, we describe only implementation strategies to face the main sources of computational overhead arising in the implementation of abstraction layers and object creation/destruction cycles.
The main goal of the library is to provide an open infrastructure aimed at integrating developer choices without their hard-wiring inside the code. Often this goal is achieved with the strategy pattern (Gamma et al., 1996). With this pattern the algorithms are implemented separately as subclasses of an abstract strategy class. While this implementation avoids the hard-wiring of user choices inside the library code, it causes inefficiency due to the abstraction levels introduced to support the polymorphic behaviour. This remark, together with the observation that the choices are immutable at runtime, suggested the use of static polymorphism in the implementation of the current version of the library. Static polymorphism is a more effective flavor of polymorphism based on the use of templates. Templates were originally conceived to support generic programming, as they are functions or classes that are written for one or more types not yet specified. Each template parameter models one degree of variability of the problem domain. This parameter must be fixed at compile time allowing the compiler to generate the proper code. This static polymorphism guarantees type checking and improves code optimization. Exploitation of templates to perform code generation is also known as generic programming (Alexandrescu, 2001).

```
template< State,
          SensorData,
          Control,
          SampleCounter,
          SampleManager,
          Fusion >
class Localizer : public SampleManager<State>
{
    Fusion<SensorData> fusion_;

  public:
    template< ... >      // StateConverter
    Localizer(Pdf &pdf, StateConverter<...> &converter, tods::TimeInterval &period);

    ~Localizer();

    template< ... >      // SystemModel Parameters
    void update(Control &u, SystemModel<...> &sys);

    template< ... >      // SensorModel Parameters
    void update(SensorData &z, SensorModel<...> &sen);
};
```

Listing 1 The Localizer class.

To reduce the second source of computational overhead, we focused our attention on the creation/destruction of the objects within the localization cycle to reduce their dynamic allocation. Fig. 3 presents the life cycle of the objects created and destroyed during a single localization cycle of the RTPF algorithm. During each step, new samples and new objects for representation of odometric and sensor data are created to be immediately destroyed. Note that the samples and controls are created in different iterations of the localizer in the same estimation window and survive after the end of this window: that is a rather different way of handling objects from a standard particle filter implementation. Thus a management

policy and proper data structures for storage are required in a flexible implementation. Therefore observations, even if they are created and used in an iteration and their handling is quite straightforward, exhibit a variety of sensor models given by the range of sensors. To support such a variety, the library provides generic sensor data.

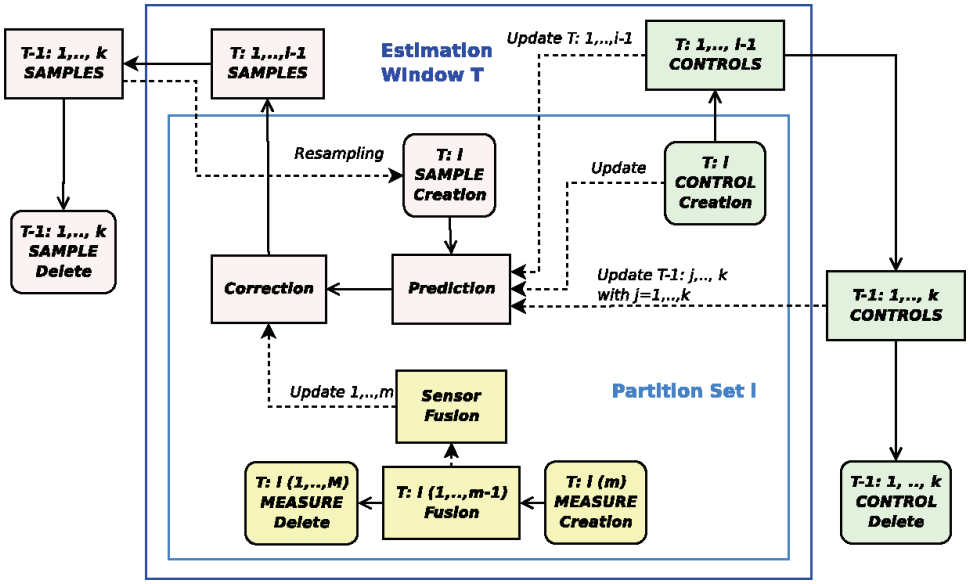

Fig. 3 Life cycle for particle, measurement, and control objects within a single step in a real-time particle filter.

5. Results

In this section performance of RTPF is evaluated both in simulated environment and using experimental data collected by robot sensor. An important purpose of this section is the comparison of the two RTPF versions differing in the method for computing mixture weights.

5.1 Simulation

Several tests were performed in the environments shown in Fig. 4 and Fig. 5. They correspond to the main ground floor hallway in the Computer Engineering Department of the University of Parma (Fig. 4) and to the hallway of the Department of Computer Science and Engineering of the University of Washington (Fig. 5, map adapted from (Kwok et al., 2004)). These environments allow verification of RTPF correctness while coping with several symmetric features, which may cause ambiguities in the choice of correct localization hypotheses. The environment of Fig. 5 had been exploited in (Kwok et al., 2004) 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 (Thrun et al., 2005). 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 quality. 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 Fig. 4 and Fig. 5 along assigned trajectories. Simulated trajectories, labeled as Path 1 and Path 2 in Fig. 4 and Fig. 5, correspond to lengths of approximately 5 to 8 m.

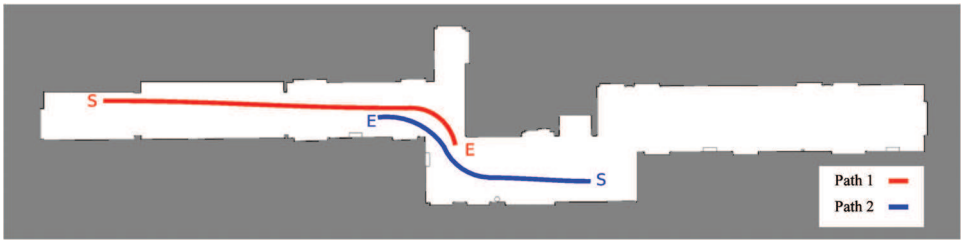

Fig. 4 Map 1 – Hallway and simulated paths in the Computer Engineering Department, University of Parma.

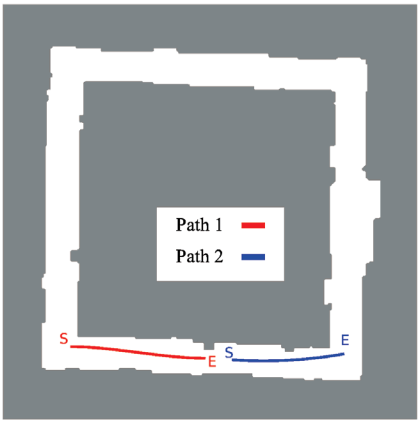

Fig. 5 Map 2 – Hallway and simulated paths in the Department of Computer Science and Engineering, University of Washington.

Localization algorithms investigated are RTPFs in the two versions: the original steepest descent-based one (RTPF-Grad) and the proposed one 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 Fig. 6 and Fig. 7, 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 Fig. 6 and Fig. 7.

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 trend in Fig. 6 and Fig. 7 does not correspond to any of the simulated trials; rather, it is the result of averaging trials with quick convergence and trials where the correct hypothesis could only be recovered after several more iterations.

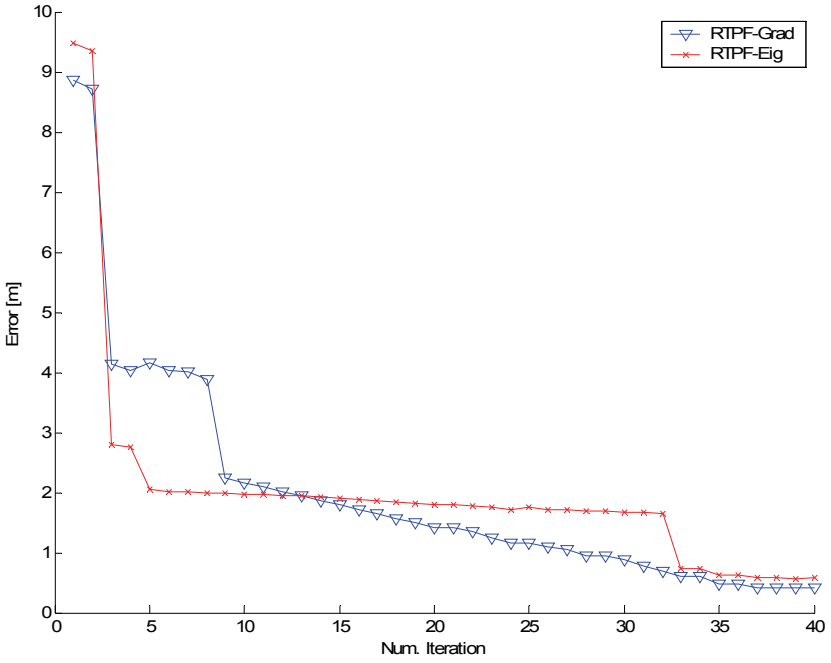

Fig. 6 Performance of the two RTPF versions in the simulated environment of Map 1. The x-axis represents the iterations of the algorithm. The y-axis shows the average error distance of the estimated pose from the actual robot pose.

Fig. 8 provides an alternative view of the same data, as curves show the percentage of simulation trials converging to the correct hypothesis (i.e. with localization error less than 1.5 m) at each iteration. For both environments, convergence is reached with only few

iterations in some simulation runs. In other 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.

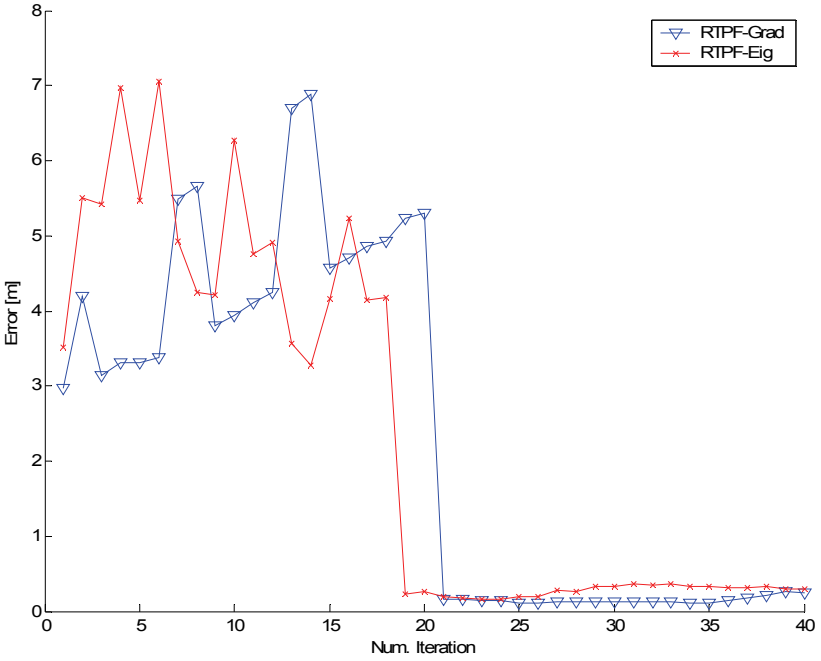

Fig. 7 Performance of the two RTPF versions in the simulated environmentof Map 2. The x-axis represents the iterations of the algorithm. The y-axis shows the average error distance of the estimated pose from the actual robot pose.

5.2 Experiments

Real experiments took place in the environment of Fig. 4 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. In the real environment localization was always successful, i.e. it always converged to the hypothesis closer to the actual pose in less than 10 iterations (remarkably faster than in simulation). Localization error after convergence was measured below 50 cm, comparable or better than in simulation.
To assess the consistency of the localizer's output on a larger set of experiments, 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

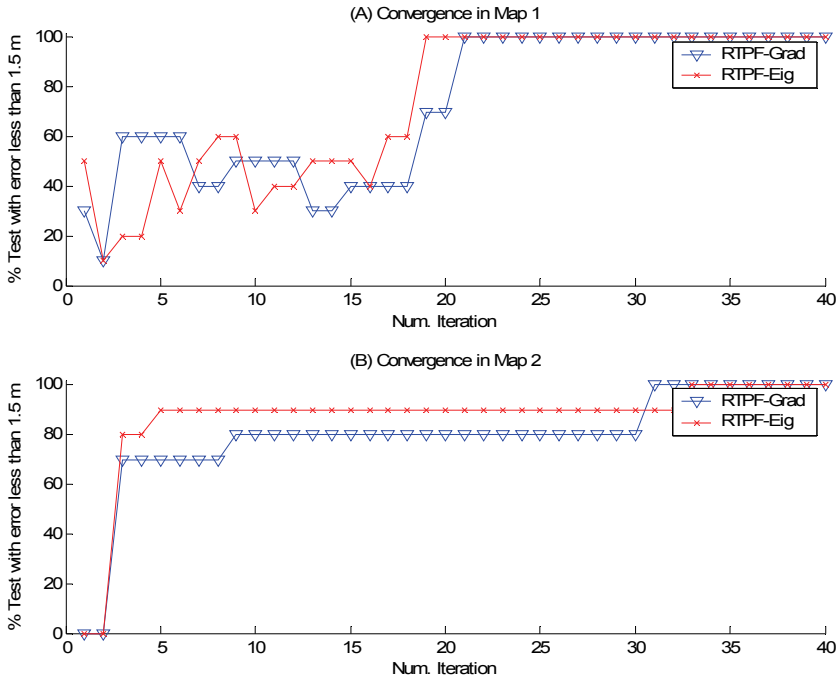

Fig. 8 Percentage of simulation trials converged to the correct hypothesis, i.e. with localization error less than 1.5 m, during iterations for Map 1 (a) and Map 2 (b).

landmarks were placed in the environment and on the mobile robot, and a vision system exploiting the ARToolKit framework (Kato & Billinghurst, 1999) was exploited to triangualate 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. 9 shows the results of 10 tests of RTPF-Eig over about 20 iterations. These results confirm that RTPF-Eig achieves localization to the correct hypothesis very fast in most experiments. 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.

6. Conclusion

In this chapter, we have described an enhanced Real-Time Particle Filter for mobile robot localization incorporating several formal and practical improvements. We have presented a formal discussion of computation of mixture weights in RTPFs, along with a new approach overcoming potential problems associated with the existing technique. The method proposed in this chapter 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. We have also

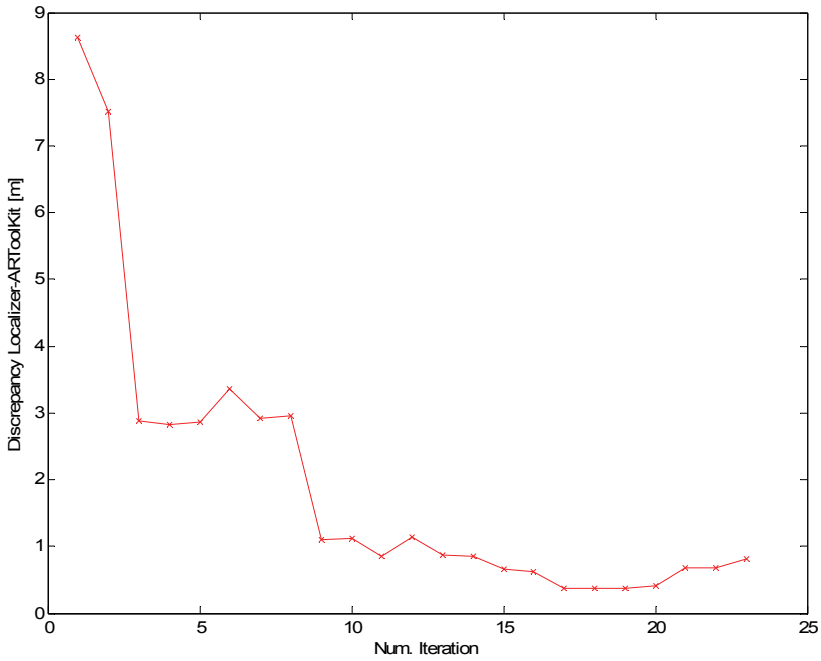

Fig. 9 Performance of RTPF-Eig using real data collected in the hallway of Map 1.

described a library efficiently supporting implementation of particle filter algorithms and independent from any specific robot control architecture. The library takes advantage from generic programming and from a carefully designed object lifecycle model to minimize overhead while providing flexibility.

The proposed approach has been implemented in a RTPF for localization with a Nomad 200 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.

Of course, further experimental work is required to assess the relative merit of the improved RTPF over the original approach. Nonetheless, the research described in this paper shows how a through theoretical understanding of the problem and of the algorithmic solution should be combined with a careful software implementation to attain the potential of probabilistic localization methods.

7. Acknowledgement

This research has been partially supported by Laboratory LARER of Regione Emilia-Romagna, Italy.

8. References

Alexandrescu, A. (2001). Modern C++ Design: Generic Programming and Design Pattern Applied. *Addison-Wesley*.

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

Cover, T. M. & Thomas, J. A. (1991). Elements of Information Theory. *Wiley*.

Doucet, A., de Freitas, J., & Gordon, N. (2001). Sequential Monte Carlo Methods in Practice. Springer.

Elinas, P. and Little, J. (2005). sMCL: Monte-carlo localization for mobile robots with stereo vision. *Proc. of Robotics: Science and Systems*.

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

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

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). Design Patterns: elements of reusable object-oriented software. *Addison-Wesley*.

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

Hester, T., & Stone, P. (2008). Negative Information and Line Observations for Monte Carlo Localization. *IEEE Int. Conf. on Robotics and Automation*, pages 2764–2769.

Kato, H. and Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. *Proc. of the Int. Workshop on Augmented Reality*.

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

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

Leonard, J. J. & Durrant-Whyte, H. F. (1991). Mobile Robot Localization by Traking Geometric Beacons. *IEEE Int. Conf. on Robotics and Automation*.

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

Montemerlo, M., Roy, N., & Thrun, S. (2003). Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale invariant visual landmark. *Int. J. of Robotics Research*, 21(8):735–758.

Sridharan, M., Kuhlmann, G., & Stone, P. (2005). Practical vision-based Monte Carlo localization on a legged robot. *IEEE Int. Conf. on Robotics and Automation*, pages 3366–3371.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabibilistic Robotics. *MIT Press*, Cambridge, MA.

Robotics Automation and Control

Edited by Pavla Pecherkova, Miroslav Flidr and Jindrich Dunik

ISBN 978-953-7619-18-3
Hard cover, 494 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

This book was conceived as a gathering place of new ideas from academia, industry, research and practice in the fields of robotics, automation and control. The aim of the book was to point out interactions among various fields of interests in spite of diversity and narrow specializations which prevail in the current research. The common denominator of all included chapters appears to be a synergy of various specializations. This synergy yields deeper understanding of the treated problems. Each new approach applied to a particular problem can enrich and inspire improvements of already established approaches to the problem.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821