Noname manuscript No.
(will be inserted by the editor)

Multirobot Localization in Highly Symmetrical Environments

Fabrizio Abrate, Basilio Bona, Marina Indri, Stefano Rosa, and Federico Tibaldi

the date of receipt and acceptance should be inserted later

Abstract The paper addresses and solves the problem of multirobot collaborative localization in highly symmetrical 2D environments, such as the ones encountered in logistic applications. Because of the environment symmetry, the most common localization algorithms may fail to provide a correct estimate of the position and orientation of the robot, if its initial position is not known, no specific landmark is introduced, and no absolute information (e.g., GPS) is available: the robot can estimate its position with respect to the walls of the corridor, but it could be critical to determine in which corridor it is actually moving. The proposed algorithm is based upon a particle filter cooperative Monte Carlo Localization (MCL) and implements a three-stage procedure for the global localization and the accurate position tracking of each robot of the team. Online simulations and experimental tests, which investigate different situations with respect to the number of robots involved and their initial positions, show how the proposed solution can lead to the global localization of each robot, with a precision sufficient to be used as starting point for the subsequent robot tracking.

Keywords multi-robot · localization · symmetric environments · logistic areas

1 Introduction

Multirobot collaboration is one of the most challenging and promising research areas in mobile robotics. A team of robots, suitably coordinated, can be used to execute complex tasks, as in surveillance, monitoring, and mapping, to cite only a few.

In these tasks the correct and reliable localization with respect to a known map is of capital importance, and represents one of the most fundamental problems in mobile robotics: a comprehensive study is reported in [24]. Potentially the multirobot case gives some interesting advantages, since the accuracy of the robots pose estimates can be improved by a

F. Abrate
Istituto Superiore Mario Boella, via P.C. Boggio 61 10129 Torino,
E-mail: fabrizio.abrate@ismb.it

B. Bona, M. Indri, S. Rosa, and F. Tibaldi
Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
E-mail: basilio.bona, marina.indri,stefano.rosa,federico.tibaldi@polito.it

cooperative localization, even if wireless communication and data sharing problems must be considered. Extended Kalman Filters (EKF) and Monte Carlo Localization (MCL) methods are the most common filtering algorithms applied to robot localization. The data association problem in the EKF approaches can be solved in various ways, for instance by multi modal distributions that approximate the position probability distribution, sometimes including iterations that propagate also an estimate of the posterior marginal densities of the unknown variances. Another possibility is to use visual feature descriptors (see for instance [7]). In [13] the global state of the environment is obtained by fusing the environment states of the vehicles; the poses of the detected vehicles are represented by a single system. The method proposed in [14] is based on a series of hierarchically distributed EKF filters. The team is decomposed in several groups, each with a group leader, and for each group an extended Kalman filter estimates the configurations of all the members. Then one filter is used to estimate the location of the group leaders. In [17] the authors present a study of the localization performances of heterogeneous robotic teams with arbitrary relative position measurement graphs. Interlaced Extended Kalman Filter is used in [18]. The authors of [20] propose an algorithm which can estimate both the robot poses and the unknown covariance parameters by an approximation of the posterior marginal densities of the unknown variances. In [22] the authors propose a decentralized estimation schema, called *collective localization*, which is used to fuse measurements collected from different sensors onboard of the different robots.

The MCL methods approximate the probability density to be estimated using a finite set of samples. The collaborative approach proposed in [9] uses a sample-based version of Markov localization. Other examples of collaborative MCL algorithms using different sensors exist. In [12] information on detected objects is used to improve self localization and is then propagated to other team members. The approach described in [19] improves the performance of low-cost sensors by exploiting multirobot communication. The authors of [21] exploit MCL collaborative localization for exploration tasks. In [10] the authors use a Fast Conjunctive Resampling Particle Filter (FCR-PF) for multirobot collaborative localization. In the algorithm proposed in [15] particles are exchanged between robots when they see each other and then evaluated on the basis of the measurement results of both robots.

Cooperative robust multirobot localization has also been proposed, in which unknown but bounded error models are employed for the sensor measurements (see e.g., [16], [23]).

Localization includes two distinct sub-problems: *position tracking* and *global localization*. In the first one, the robot pose is iteratively estimated while the robot moves starting from an initial condition, known with a given uncertainty, while the second one determines the absolute robot position with respect to a given environment map; this problem is the most challenging, since no information of initial pose – or a completely wrong estimation of the actual pose, as in the so-called *kidnapped robot* – can be available.

Many of the papers cited above use multirobot and/or mutual localization to improve the quality of self-localization estimates that single robots could achieve on the basis of their own sensors only, implicitly assuming that the measurements provided by such sensors would be suitable to obtain a sufficiently correct, even if not precise, global localization. Unfortunately, without some external absolute information, a correct global self-localization could be critical for a single robot when the environment is highly symmetrical.

Highly symmetrical environments are commonly encountered in large logistic spaces, like the ones considered in this paper, where a team of robots performs surveillance and monitoring tasks. A logistic space is similar to an indoor or outdoor warehouse, i.e., an area where logistic or transport companies receive, store and distribute large quantities of goods, as containers, cars, crates and other similar items. In order to achieve an efficient occupancy

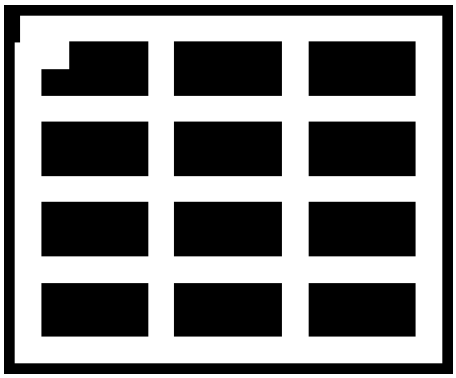

Fig. 1 The map of the environment.

of the area and facilitate the handling operations, free corridors among the stored goods often form a regular grid, as in Figure 1.

Some commercial systems are already available for logistic applications, in particular for warehouse automation but these systems do not exploit multirobot localization. Instead they rely on ad-hoc markers or perform single-robot SLAM and localization (see e.g., *Kiva Systems* [1] and *Seegrid* [4]).

In a fully symmetrical environment reliable global self-localization of each robot cannot be performed when its initial position is unknown, no specific landmarks are introduced to discriminate each corridor, and no absolute information (e.g., from GPS) is available or exchanged with the other robots. By using only its own sensors (odometry, laser scanner, sonar, etc.) a robot could estimate its position within the corridors, but may not be able to determine in which corridor is actually moving. In highly symmetrical environments a correct global self-localization could be possible only with long convergence times or thanks to ad-hoc path planning strategies.

This paper presents a solution for the global localization problem in highly symmetrical environments, along the lines of the cooperative MCL approach outlined in [9] (a detailed comparison with this approach is developed at the end of Section 2.2). The proposed solution does not use any absolute sensor data (e.g., GPS), that could be unavailable in some indoor areas, but relies only on minor asymmetries present in the maps (see for instance the superior left-hand corner in Figure 1), without using ad-hoc path planning strategies.

The idea is to exploit the Cartesian position measurements (in the global reference frame) that each robot receives from the other members of the team when it is in their field of view, to quickly propagate the information about the few asymmetries of the environment. The proposed algorithm is based on particle filters [24] and includes an original resampling algorithm (given by a modified version of the classical Kullback-Leibler Divergence solution), used each time a robot receives measurement informations from another member of the team.

Simulation and experimental tests investigate different cases involving a variable number of robots in the team, their initial positions, and some possible critical situations. The results show how the proposed algorithm, based upon a three-stage localization procedure, leads to the global localization of each robot, within a precision sufficient to be used as starting point for the subsequent robot tracking.

The paper is organized as follows: Section 2 describes the proposed localization algorithm, called **3SMCL**, Section 3 describes and reports the different simulation and experimental tests performed to demonstrate the effectiveness of the algorithm, and finally Section 4 draws some conclusions.

2 The Three-State Multirobot Collaborative Localization (3SMCL)

2.1 Preliminaries

The ***T**hree-**S**tate **M**ultirobot **C**ollaborative **L**ocalization* algorithm (**3SMCL**) allows each member of a group of robots moving in a highly symmetrical area to accurately localize itself and to correctly track its position over time. The **3SMCL** algorithm requires the robots to be endowed at least with a range sensor and a monocular camera. The camera is used to detect the positions of other robots when they are in the field of view; the measurement accuracy may be improved using a laser range finder if available. A binary occupancy grid map of the environment is assumed to be available.

The algorithm has been conceived as a finite state machine, with three states: 1) GL = *global localization*, 2) UN = *undecided*, and 3) PT = *position tracking*. We already proposed in [5] an approach to solve the multirobot localization problem using an algorithm based on two states only, but applied to a *completely* symmetric environment, when absolute heading measurements are only occasionally available.

At the beginning of the execution of the algorithm, each robot is in the GL state, as no information is available about its initial position, and hence it has to be *globally* localized with respect to the map. When the number of feasible hypotheses about its actual position becomes small (as detailed in the next Subsection), the robot enters the UN state. Finally, it switches to the PT state, when it is supposed to be correctly localized with a high degree of confidence. If a sudden increase in the localization error is detected, due for instance to kidnapping or failures in proprioceptive sensors (e.g., wheel encoders rupture) or changes in the map, the algorithm may switch again to the UN state and then to the GL state.

Let $\mathscr{R} = \{r_i : i = 1, \ldots, N_R\}$ be the set of robots deployed in the area; with t we indicate the time variable that clocks the whole localization algorithm. With $d_i(t)$ we indicate data coming from the i-th robot proprioceptive and exteroceptive sensors at time t. In particular we have that

$$d_i(t) = \begin{cases} o_i(t) & \text{if proprioceptive measurement} \\ z_i(t) & \text{if exteroceptive measurement} \end{cases}$$

The proprioceptive measurement $o_i(t)$ is used to perform dead-reckoning, while the exteroceptive measurement $z_i(t)$ contains the range measurements given by the range sensors.

Each robot is able (a) to measure the positions of the other robots in the field of view of its vision sensor in its local reference frame, concurrently with the **3SMCL** algorithm, (b) to transform the measurements in a global reference frame common to all robots, and (c) to send these values to the detected robots via a wireless link. Position hypotheses are generated only from

Let k denote a time instant at which the position of the i-th robot is detected by a set of robots $R_i(k) \subseteq \mathscr{R}$, ($|R_i(k)|$ being its cardinality). The robots belonging to $R_i(k)$ send their

measurements to the i-th robot, which collects them in the following matrix:

$$h_i(k) = \begin{bmatrix} \hat{x}_i^1(k)\ \hat{y}_i^1(k) \\ \vdots \\ \hat{x}_i^{|R_i(k)|}(k)\ \hat{y}_i^{|R_i(k)|}(k) \end{bmatrix}. \quad (1)$$

$\hat{x}_i^j(k)$ and $\hat{y}_i^j(k)$, $j = 1,\ldots,|R_i(k)|$ are vectors containing the estimates of the position of the i-th robot expressed in Cartesian global coordinates. The position estimates contained in the j-th vector are generated starting from the position hyphoteses of the j-th robot which has detected robot i. These position hypotheses are the result of a *Density-Tree* clustering [9] procedure applied to the set of particles. Position estimates are generated only from position hypotheses whose weight is above a given threshold. The likelihood of the position hypotheses is not propagated in matrix $h_i(k)$ because in highly symmetrical environments a high weight does not guarantee the correctness of the hypothesis.

The set of all measurements received by the i-th robot up to time k is defined as $H_i^k = \{h_i(1),\ldots,h_i(k)\}$.

2.2 The algorithm

We now describe the core of the **3SMCL** algorithm, which runs onboard each robot.

The algorithm outlined in Figure 2 is basically organized as a Finite State Machine (FSM) with three states. The first state is the default initial state and it is active when the robot is in GL state. Then, when a proper *accordance* function (later defined) is below a certain threshold, the algorithm enters the UN state. This state indicates that there are two or more "dominant" position hypotheses, based on the axes of symmetry of the map, but the algorithm is still not able to definitely choose which hypothesis has the highest likelihood to be the right one.

When one hypothesis markedly prevails, i.e., when the localization performances are sufficiently accurate, the algorithm changes its state to PT.

It is important to stress that the the GL state is the only state in which the belief of a robot (let's say robot R_A) is influenced by the estimates of the other robots detecing R_A at a certain time. This means that the particle set of the R_A robot is directly influenced by external measurements *only* when it is in the GL state. Typically, when a robot is in the GL state, its pose estimate is highly uncertain, since various position hypotheses are still "alive". For this reason, if a robot in the GL state receives uncertain position hypotheses from other robots that are in UN or PT state, its position uncertainty is reduced. Once the robots reach the UN state (and then the PT state), the localization performances are only monitored, and the belief of the robots is no more mutually influenced. Ideally, once reached the PT state, the robots should never switch back to the UN state. However, the algorithm continues to monitor the localization performances. In case of localization performance degradation, the algorithm switches again to the UN state and possibly to the GL state.

The algorithm is based on particle filters [24]. It receives as inputs the set of particles χ^{t-1} at time $t-1$, the sensors measurements $d_i(t)$, the set of robots $R_i(k)$ detecting the i-th robot at time k, the lower and upper bound of the number of particles N_{kld} employed in the resampling algorithm (N_{min} and N_{max} respectively), the grid map m of the environment, and N_{hyp}, which is the maximum number of particles that can be used to approximate the probability distribution of the received position hypotheses, as detailed later. The algorithm gives as output the set of pose hypotheses $\Phi_i(t)$ of the i-th robot and the best one ϕ_i^{best}.

Observing the pseudo-code of the algorithm in Figure 2 relative to the GL and UN states, it can be noticed the typical prediction phase at line 13 and the update phase at lines 15-16.

Require: $\chi^{t-1}, d_i(t), R_i(k), H_i^k, N_{min}, N_{max}, N_{hyp}, m$
Ensure: $\Phi_i(t), \phi_i^{best}(t)$
1: **if** state = 'PT' **then**
2: $[\chi^t, \mu_k, l]$ = **position_tracking**$(d_i(t), \chi^{t-1}, h_i(k), l)$
3: $[\Phi_i(t), \phi_i^{best}(t)]$= **DT_clustering**$(\chi_t)$
4: **if** $l > n_{p2u}$ **then**
5: $[\mu_k]$ = **loc_perf**$(\phi_i^{best}(t), h_i(k))$
6: **if** $\mu_k \geq \mu_{p2u}$ **then**
7: state = 'UN'; $l = 0$
8: **end if**
9: **end if**
10: **else**
11: initialize χ_t
12: **if** $d_i(t) = o_i(t)$ **then**
13: $p_i(t)$ = **sample_motion_model**$(d_i(t), p_i(t-1))$
14: **else if** $d_i(t) = z_i(t)$ **then**
15: $w_i(t)$ = **measurement_model**$(d_i(t), p_i(t), m)$
16: $\bar{\chi}^t = \bar{\chi}^t + \langle p_i(t), w_i(t) \rangle$
17: **if** $R_i(k) = \emptyset$ **then**
18: χ^t = **KLD_1**$(\bar{\chi}^t, N_{min}, N_{max})$
19: $[\Phi_i(t), \phi_i^{best}(t)]$ = **DT_clustering**(χ^t)
20: **else**
21: $l = l + 1$
22: **if** state == 'GL' **then**
23: χ^t = **Mutual_KLD**$(\bar{\chi}^t, N_{min}, N'_{max}, N_{hyp}, h_i(k))$
24: $[\Phi_i(t), \phi_i^{best}(t)]$ = **DT_clustering**(χ^t)
25: **end if**
26: **if** state = 'UN' **then**
27: **if** $l > n_{u2p}$ **then**
28: $[\mu_k]$ = **loc_perf**$(\phi_i^{best}(t), H_i^k)$
29: **if** $\mu_k \leq \mu_{u2p}$ **then**
30: state = 'PT'; $l = 0$
31: **end if**
32: **end if**
33: **if** $l > n_{u2g}$ **then**
34: $[\mu_k]$ = **loc_perf**$(\phi_i^{best}(t), H_i^k)$
35: **if** $\mu_k \leq \mu_{u2g}$ **then**
36: state = 'GL'; $l = 0$
37: **end if**
38: **end if**
39: **end if**
40: **if** state = 'GL' **then**
41: **if** $dist < \mu_{g2u}$ **then**
42: $state = 'UN'$
43: **end if**
44: **end if**
45: **end if**
46: **end if**
47: **end if**

Fig. 2 The 3SMCL algorithm in pseudocode

The prediction phase computes the vector $p_i(t)$ containing the predicted pose (in terms of global coordinates $(x,\ y,\ \theta)$) for each particle, while the purpose of the update phase is twofold. It gives the vector $w_i(t)$ containing the importance factors for each particle, and it verifies whether matrix $h_i(k)$ contains position estimates outside the map. If this is the case, their weight quickly decreases. Then the particles after the update phase are added to the temporary set $\bar{\chi}^t$. Then, at line 17, the algorithm verifies if it has received a vector of measurements $h_i(k)$ from other robots of the set $R_i(k)$ at time k. If $R_i(k)$ is empty, a classic *Kullback-Leibler Divergence* (*KLD*) resampling occurs (see [24]); N_{min} and N_{max} are respectively the lower and upper bound of the number of particles N_{kld} employed in the resampling algorithm. If instead $R_i(k)$ is not empty, a modified version of the *KLD* resampling is implemented (line 23). The idea is to exploit the Cartesian position measurements (contained in vector $h_i(k)$) that the i-th robot receives from the other robots of $R_i(k)$ to propagate the information about the few asymmetries of the environment. In the preliminary solution implemented in [6] the proposed policy simply employed all the available particles

($N_{max} - N_{kld}$) to approximate each belief contained in matrix $h_i(k)$. In this work we adopt the *Kullback-Leibler Divergence* to compute the number of particles needed to approximate each pose hypothesis contained in matrix $h_i(k)$ and we propose an original resampling algorithm called *Mutual KLD* outlined in Figure 4. The algorithm is described later in Subsection 2.3. After this resampling phase, a classic *Density-Tree* clustering [9] (lines 3, 19, 24 of Figure 2) is always performed, providing a set of N_h hypotheses $\Phi_i(t) = \{\phi_i^j(t)\}$, $j = 1, \ldots, N_h$, on the pose of the i-th robot, among which the best hypothesis $\phi^{best}(t)$ is selected. Each hypothesis $\phi_i^j(t)$ consists of the predicted pose $p_i^j(t)$, its covariance matrix $\Sigma_i^j(t)$, and the associated weight $W_i^j(t)$, representing its confidence level:

$$\phi_i^j(t) = \{p_i^j(t), \Sigma_i^j(t), W_i^j(t)\}. \tag{2}$$

The best hypothesis at time t is defined as

$$\phi_i^{best}(t) = \arg\max_{W_i^j}(\Phi_i(t)) = \{p_i^{best}(t), \Sigma_i^{best}(t), W_i^{best}(t)\}. \tag{3}$$

The center of mass of its own hypotheses is defined as

$$\bar{p}_i(t) = \sum_{j=1}^{N_h} \frac{p_i^j(t)}{N_h}. \tag{4}$$

The mean distance among its own hypotheses is defined as

$$dist = \sum_{j=1}^{N_h} \frac{\sqrt{(\bar{x}_i(t) - x_i^j(t))^2 + (\bar{y}_i(t) - y_i^j(t))^2}}{N_h}, \tag{5}$$

Switching rules among the three states are based on the following *accordance* function:

$$\mu_k = \sum_{q=k-n}^{k} \sum_{j=1}^{|R_i(q)|} \frac{\sqrt{(\hat{x}_i^j(q) - \hat{x}_i^{best}(t))^2 + (\hat{y}_i^j(q) - \hat{y}_i^{best}(t))^2}}{n|R_i(q)|}, \tag{6}$$

where n is the length of the sliding window used to compute the average in (6). This quantity is an average on the distance between the hypotheses and the best position estimate of the i-th robot, averaged over the last n times the i-th robot has been detected. It must be noted that both (4) and (6) do not consider the weights of the hypotheses, as the particle filtering framework would suggest. The reason depends on the particular application field we are considering, which is localization in highly symmetrical environments. In this situation, it can frequently happen that a position hypothesis with a high weight is not the correct one. Therefore, using weights in (4) and (6) may lead to a degradation in performance monitoring.

If the robot is in the UN state and it is deciding whether it can switch to PT, n is set equal to n_{u2p}. If the robot is in the UN state and it is deciding whether it has to switch back to GL, n is set equal to n_{u2g}. The variable l is used to count the times the i-th robot is seen by the other robots in the set $R_i(k)$ at time k, and it is useful to decide when the i-th robot has been seen n_{u2p} or n_{p2u} times, hence when it is appropriate to evaluate the accordance function of (6).

The inner summation in (6) averages the distances among the elements of $h_i(k)$ and the best position hypotheses of the i-th robot. The outer summation in (6) performs a moving average of length n on the results of the inner summation. Therefore μ_k measures the accordance between the actual belief on the position of the i-th robot and the average of the beliefs that the other robots have on its position at time k.

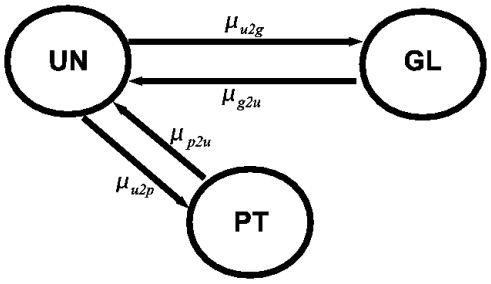

Fig. 3 The states and the thresholds.

When μ_k is lower than a certain threshold μ_{u2p} (empirically determined) the algorithm switches to PT. This phase is aimed at tracking the position of the robot over time, and it is implemented using classical *KLD* resampling (see [24]). μ_k is computed also during the position tracking phase: if μ_k becomes greater than a given threshold μ_{p2u}, the algorithm switches again to UN.

When the robot is in GL, it switches to UN if and only if the distance among the hypothesis defined in (5) is smaller than a certain threshold μ_{g2u} (see Figure 2, lines 40-42 and Figure 3). The values of the various thresholds that are present in the algorithm are empirically determined, taking into account the environment characteristics, mainly its dimensions (see 2.4 for a more detailed discussion).

Remark 1 Before proceeding, we want here to point out the main differences between the proposed approach and other approaches, mainly [9].
The first difference between the method proposed in [9] and our method lies in how two generic robots (R_A and R_B) are able to mutually influence their belief. In [9] when one robot detects another, the sample sets are synchronized implementing a sampling version of the Multi-Robot Markov localization. In particular (13) of [9] shows the synchronization phase from a theoretical point of view. Equation (13) is implemented using *density trees* and a refined density reflecting both the beliefs of R_A and R_B is obtained.
In our work robot R_A, detecting robot R_B, sends to R_B a subset of the position hypotheses it has about robot R_B (implemented by a clustered version of the density trees of [9]). Then robot R_B draws randomly on the hypotheses a number of particles that depends on the *Mutual KLD* resampling criteria detailed in Subsection 2.3.
This idea of mutually influencing the belief of the robot can be found also in another recent paper [10]. The advantage of our work with respect to [10] is that our *Mutual KLD* assures the automatic selection of a number of particles that refine the distribution and guarantees efficiency.

Another difference between our work and [9] is that we have introduced a method to evaluate if the team is correctly localized. The method consists of introducing a Finite State Machine with three states, and a switching mechanism among the states based on the evaluation of a proper *accordance function*. We do not claim here to have a proof about the *convergence* of the particle filter to the true hypothesis, but we have experimental validation about the correct localization of the team when the robots are in the PT state. The reliability of localization in our PT state is a very important condition from the application point of view, since once all the robots have reached the PT state, correct localization can be assumed and generic tasks assigned to the team can be carried out.

2.3 Mutual KLD resampling

In the solution proposed in this paper we adopt the *Kullback-Leibler Divergence* to compute the number of particles needed to approximate each pose hypothesis contained in matrix $h_i(k)$ and we propose an original resampling algorithm called *Mutual KLD*, which is outlined in Figure 4. The intuition behind this algorithm is that it allows to refine the probability distribution on the position of a robot using the estimates of other robots and to determine the number of particles needed to approximate the distribution according to the *KLD*. In line

Require: $\bar{S}^t, N_{min}, N_{max}, N_{hyp}, N_{kld}, R_i(k), \varepsilon$
Ensure: χ^i
1: $n_{hyp} = |h_i(k)|$
2: **if** $N_{kld} > N_{max} - N_{hyp}$ **then**
3: $N_{kld} = N_{max} - N_{hyp}$
4: **end if**
5: **if** $N_{limitHyp} = \frac{N_{max} - N_{kld}}{n_{hyp}}$ **then**
6: **for** $i = 1 : n_{hyp}$ **do**
7: $N^i_{mkld} = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta}$
8: $N^i_{curr} = min\{N^i_{mkld}, N_{limitHyp}\}$
9: add N^i_{curr} to S^t_i
10: $N_{kld} = \sum_i N^i_{curr}$, $i = 1, \ldots, n_{hyp}$
11: $S^t = \sum_i S^t_i$, $i = 1, \ldots, n_{hyp}$
12: **end for**
13: **end if**

Fig. 4 The Mutual *KLD* resampling

1 the number n_{hyp} of hypotheses received is calculated as the cardinality of the elements contained in $h_i(k)$. N_{kld} is the number of particles used to approximate the belief without taking into account the position hypotheses coming from the other robots. This number is computed in lines 2-3. If this number is greater than $N_{max} - N_{hyp}$, we reduce it to $N_{max} - N_{hyp}$, where N_{hyp} is the maximum number of particles that can be used in this case to approximate the probability distribution of the n_{hyp} position hypotheses.
Then, the number of particles N^i_{mkld} is computed for the i-th hypothesis as

$$N^i_{mkld} = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta}$$

(as in (13) in [8]), where $\chi^2_{k-1,1-\delta}$ is a chi-square distribution with $1-k$ degrees of freedom. This value is the required number of particles to guarantee that with probability $1-\delta$ the Kullback-Leibler distance between the Maximum Likelihood Estimate (MLE) of the position hypothesis and the true distribution is less than ε.
It is not necessary to accurately approximate the probability distribution of the incoming hypotheses, because they are only used to add coarse information on the position of the i-th robot. Therefore we can accept an approximation with few particles of the probabiliy distribution of these hypotheses. By increasing or decreasing the value of ε and δ, better or worse approximations of the distribution of each position hypothesis are set. This has indeed an impact on the number of particles used to approximate the final belief on the position of the robots, as it will be discussed in Remark 2 of Subsection 3.4.
Higher values of ε allow to potentially use less than $N_{max} - N_{kld}$ particles to approximate the belief of the robots, hence the value of ε in the case of our *Mutual KLD* is higher than the value adopted in the standard *KLD* resampling.
The value of ε can indeed be seen as a tunable parameter that changes the importance of the

position hypotheses contained in $h_i(k)$ on the final belief of the i-th robot. We do not focus here on how to find general methodologies to calculate ε, but we simply set this value to five times the value of ε in the standard *KLD* sampling.
In line 8, for each hypothesis we constrain the actual number of particles N^i_{curr} to be at most equal to $N_{limitHyp}$, which is the maximum number of particles that can be reserved for each hypothesis. The new N_{kld} number of particles approximating the belief of the robot is computed in line 10 of the algorithm in Figure 4 as the sum over i of all the N^i_{curr}. For each hypothesis, then the x, y coordinates of the particles are randomly generated according to a Gaussian distribution with mean equal to the center of the hypothesis considered, and heading uniformly distributed over the $0 - 2\pi$ space.
The algorithm in Figure 4 can be seen as an application to the multirobot case of the adaptive resampling proposed in [8], since it adapts the sample set to represent the belief of a robot using also informations coming from the other robots of the multirobot team.

2.4 Parameters significance and choice guidelines

The proposed algorithm depends on a certain number of parameters, and we intend here to clarify their meaning and influence, giving also some intuition about possible strategies on how to set these parameters in different situations. The first parameters to be set are N_{min} and N_{max}, which are respectively the minimum and maximum number of particles that can be used to approximate the belief of the robot. In the case considered in this paper, the number of particles that approximates the belief of the robot position is adaptive, therefore N_{min} and N_{max} are mainly chosen according to considerations related to the computational load.
The meaning of k, ε and δ, the *KLD* resampling parameters, can be found in [8] and references therein.
N_{hyp} is the maximum number of particles employed to approximate the distribution of the n_{hyp} position hypotheses of a robot. This exact number of particles is determined by our *Mutual KLD* criterion, but it has to be constrained in some way, since the entire particle set is upper-constrained by N_{max}. $N_{limitHyp}$ is instead the maximum number of particles that can be employed to approximate each position hypothesis.
The parameters that regulate the switching among the GL, UN and PT states are μ_{g2u}, μ_{u2g}, μ_{u2p}, μ_{p2u}. A robot switches from GL to the UN state when the distance between its position hypotheses is under the μ_{g2u} threshold. If this threshold has a low value (e.g., under 2 m), the robot will switch to the UN state only when the distance between its position hypotheses is lower than 2 m. If instead this threshold has a high value (e.g., 10 m), the robot will switch very fast to the UN state.
In symmetric environments (which are the focus of this work) it happens that multiple position hypotheses have similar weight, and in this situation the particle filter may prematurely converge to one of the symmetric hypotheses, thus leading to failure in the localization process. For this reason we have designed the UN state. When the particle filter has converged to a set of hypotheses whose mean distance is lower than μ_{g2u}, the robot enters the UN state. At this point, to reach the PT state (and hence to be considered as correctly localized), the robot has to be detected a number of times equal to n_{u2p}, and the value given by the accordance function must be lower than μ_{u2p}.
When the localization performances worsen, a similar mechanism allows the robot to switch back to UN and then to GL. Now the problem is how to choose values for μ_{u2p}, n_{u2p} and the other similar parameters used when localization performances worsen. Let us consider for instance μ_{u2p}. Choosing a low value imposes that a robot, to switch to PT, needs the position

estimates sent by the other robots to be very close to the actual postion estimate of the robot receiving these estimates. In an application, this parameter should be imposed equal to μ_{g2u}, and then finely tuned, taking into account the accuracy of the sensors used, and observing the performances in simulation. The other parameter that influences the switching time between UN and PT is n_{u2p}. High values of this parameter extend the time needed for robot switching to PT, but at the same time they reduce the possibility to have switching to PT when the robot is not correctly localized yet. Therefore the tuning procedure of n_{u2p}, n_{u2g}, and n_{p2u} really depends on the considered environment.

3 Simulation and experimental tests

In this Section we validate the effectiveness of the proposed **3SMCL** algorithm, carrying out a series of online localization tests in simulation.

Mobilesim [2] simulation software was used to perform simulations of the robots and their environment. It is based on the *Stage* library [11], and it simulates MobileRobots platforms. We performed experiments with a team of simulated Pioneer P3DX robots, endowed with sonar sensors and laser range finders. The data coming from sonar is employed in the localization algorithm, while the laser range finder is used in conjunction with the camera to determine the position of the detected robots.

The simulator embeds a model of the behavior of sonar and laser range finders, provides robot odometry pose estimation with cumulative error, and allows multiple robots simulation.

The simulator has also been improved by adding a simple simulated vision sensor and the support for communication among robots.

Three environments were considered. The first environment, used in the first three tests, simulates a large logistic area (see Figure 1), where occupied black areas may be thought as containers or similar bulky items stored by transport societies before distribution. The dimension of the whole environment is 80×65 m, the black areas are 20×10 m and the corridors are 5 m wide.

In this case, the high symmetry of the environment makes the global localization a difficult task, which the proposed **3SMCL** algorithm successfully accomplishes, as the tests will show in different situations.

The second environment, used in test 4, is the hallway considered in [9] and reported here in Figure 5. We ran the **3SMCL** algorithm in this environment in order to test it in a situation similar to the one proposed in [9] in test 4.

The third environment, used in test 5, is a more realistic scenario, which presents a greater number of asymmetries, given by the interior of the Intel Research Lab in Seattle (Figure 11) available in the Radish dataset [3]. In all the simulation experiments the robots move randomly, because we are not interested here in evaluating active localization strategies. Moreover, since the localization errors shown in the following subsections are averaged over multiple runs, it makes sense comparing the localization errors, without knowing the exact paths and how often the robots observe each other.

3.1 Simulation test 1

In this test we analyze the robustness of the **3SMCL** algorithm with respect to random variations in the initial position of the robots. We define the localization error of the *i*-

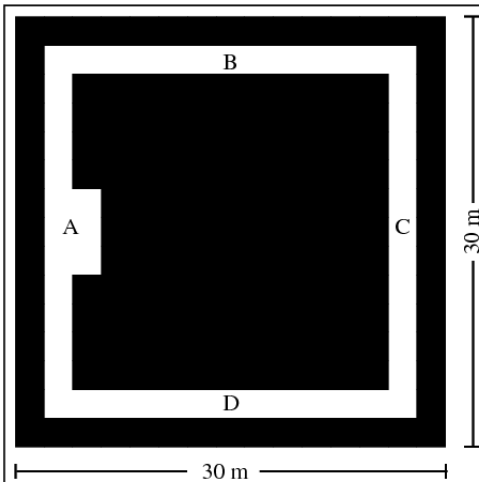

Fig. 5 The hallway.

th robot as the distance between the ground-truth Cartesian position $(x_i^{gt}(t), y_i^{gt}(t))$ and its Cartesian position estimation given by the best hypothesis, i.e., as

$$e_i^{\rho}(t) = \sqrt{(x_i^{gt}(t) - \hat{x}_i^{best}(t))^2 + (y_i^{gt}(t) - \hat{y}_i^{best}(t))^2}. \tag{7}$$

The pose information of the best hypothesis is given by $p_i^{best}(t)$ and can be extracted by $\phi_i^{best}(t)$, defined in (3).
We randomly initialize the pose of $N_R = 6$ robots in free areas of the map, let them move according to a simple obstacle avoidance behavior, and monitor the localization error $e_i^{\rho}(t)$ for $i = 1, \ldots, N_R$ up to $t = 2500$ s. We are interested in evaluating the average localization error among repetitions of the experiments, hence we repeat them several times, each time setting randomly the initial position of the robots, and we define $\bar{e}_i^{r}(t)$ as the average of $e_i^{\rho}(t)$ for the i-th robot over n_e realizations. The results are shown in Figure 6 for $n_e = 100$. The localization error $e_i^{\rho}(t)$, $i = 1, \ldots, N_R$ decreases approximately linearly for all the robots, and the mean error among all the 6 robots (dashed line in Figure 6) reaches a final value below 0.4 m. The **3SMCL** algorithm is thus not susceptible to variations in the initial positions of the robots. This fact has an important impact on the application side, in particular when considering robotic applications in logistic spaces, since the algorithm does not require any particular initial formation of the robots, avoiding any human intervention to initially place the robots in a specific area of interest.

We now give the definition of the first and the last switching time from one state to another during the experiments. The first switching time is the instant at which a robot changes its state for the first time, while the last switching time is the instant at which a robot changes its state, definitely remaining in the final state. In Table 1 and Table 2 we show the comparison between the first switching time and the last switching time between the UN and the PT states and between the GL and UN states. Table 1 refers to the first switching time, while Table 2 refers to the last switching time. The first row of each table is relative to the switching from UN to PT, while the second row is relative to the switching from GL to UN. Observing the average of respectively the first and the last switching time between the GL and the

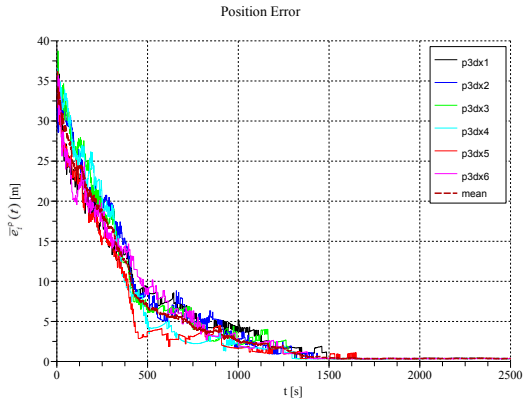

Fig. 6 Simulation test 1: average localization errors.

Table 1 First switching time (s)

	Min	Mean	Max
$UN \rightarrow PT$	560	805	1268
$GL \rightarrow UN$	26	65	137

Table 2 Last switching time (s)

	Min	Mean	Max
$UN \rightarrow PT$	560	923	1325
$GL \rightarrow UN$	203	504	774

UN states, we notice a great difference, since the first switching time is 65 s while the last switching time is 504 s. This behavior is particularly positive, since robots may switch to the UN state many times and in different moments during the localization process, thus avoiding false positive that may compromise the correct localization of the whole team. Observing again the average, it can be noticed that the last switching time between the UN and the PT states occurs only two minutes after the first switching time. This means that the algorithm does not bounce for a long time between the UN and the PT states.

3.2 Simulation test 2

This test has been designed to understand how the localization performance of the **3SMCL** algorithm is affected by the number of robots in the team, in terms of Cartesian position error.
We define the average position error among the N_R robots of the team over the $n_e = 100$ realizations of the experiments as:

$$E^r_{N_R}(t) = \frac{1}{r}\sum_{j=1}^{r}\sum_{i=1}^{N_R}\frac{e^{\rho}_i(t)}{N_R} \tag{8}$$

where $e_i^{\rho}(t)$ is the localization error defined in (7). The results of the simulations for $N_R = 1,3,6,9$ are reported in Figure 7 and in Table 3.

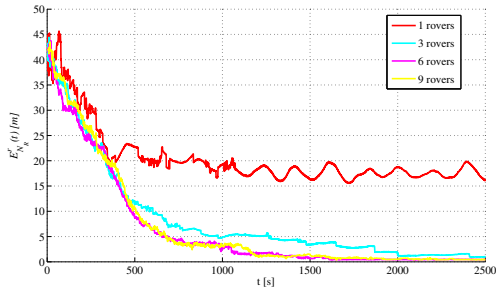

Fig. 7 Simulation test 2: average localization errors.

Table 3 Correct Localization Percentage

# of robots	Correct Localization Percentage
1	78%
3	98%
6,9	100%

It can be clearly seen that one robot is not sufficient to resolve the ambiguity in localization, and that also three robots are not enough to assure reliable localization, since robots are able to localize themselves correctly only the 98% of the trials. As soon as 6 robots are employed, the localization error goes below 2.5 m after nearly 1000 s, and all the trials are successful; the results with nine robots are comparable with those obtained with six robots. Path planning algorithms become more effective for the robots relying only on their position estimations, thus allowing robots to accomplish in a more reliable way the assigned task (e.g., handling hazardous events collaboratively). Therefore the obtained results are particularly relevant in practical applications. Of course, the *exact* number of robots that ensure correct localization of all the members of the team depends on the size of the area where the robots move. Future investigations will be devoted to study the performance of the proposed algorithm with respect to variations of the ratio between the number of robots and the area to be covered by the robot team.
Table 4 finally shows the average first and last switching times from the UN to the PT states considering 3, 6 and 9 robots. The switching times dramatically decrease passing from three robots to six and remain nearly the same when passing from six to nine robots. The performance of the **3SMCL** algorithm in terms of switching time improves increasing the number of robots of the team, since there is an evident increase in the speed with which the robots reach the PT state.

Table 4 Switching times for 3,6 and 9 robots (s)

	First	Last
3 robots	1235	1296
6 robots	770	860
9 robots	700	902

3.3 Simulation test 3

This test is aimed at demonstrating that, once all the robots are in the PT state, the algorithm is robust even with respect to partial variations of the map. To show this robustness, we have set up a case study where $N_R = 6$ robots are deployed in the same logistic area of the tests 1 and 2 (Figure 1). After all the robots have reached the PT state, a fork lift is supposed to enter the logistic area in order to remove and add pallets. The fork lift moves ideally at a constant speed of 1 m/s, and removes or adds randomly a pallet in the map, employing 3 seconds to perform these operations. Tests have been performed with a decreasing occupancy percentage, starting from 90% of occupancy in steady state condition, up to 50% occupancy with step of 10%. It is important to say that the information about the map variations are not communicated to the robots, therefore the challenge here for the **3SMCL** algorithm is to maintain the PT condition and to keep the localization error low for all the robots. Figure 8 shows the localization error $e_i^{\rho}(t)$ for $i = 1,\ldots,6$, considering only one realization of the experiment. The first plot shows the localization error reduction when the robots, in each

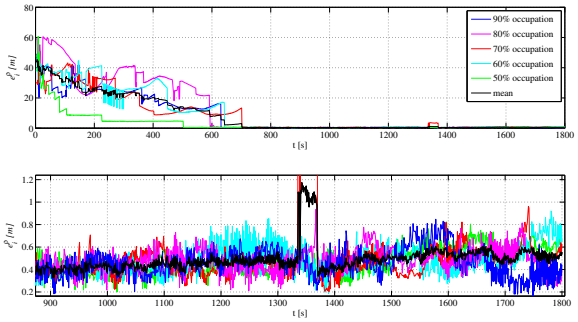

Fig. 8 Simulation test 3: case study with variations in the map.

test, reach the PT condition. On average after approximately 700 s all the robots in each test are in PT, and the algorithm that simulates the intervention of a fork lift begins to modify the map.
Observing the second plot, which is simply a zoom of the first one, we see that the error increases, but only from 0.4 m to 0.6 m. Therefore the PT state of the proposed algorithm can be considered stable with respect to random variations in the map.

3.4 Simulation test 4

We have applied the **3SMCL** algorithm in localization experiments with the map shown in [9] and reported in Figure 5.
We performed three experiments: two of them with 3 and 6 robots respectively, allowing exchange of mutual information (so running the **3SMCL** algorithm) and the third one with 6 robots, with no information exchange. In this case the algorithm reduces to a classical MCL algorithm with *KLD* resampling.

The number of repetitions is 50, as in Test 2. Figure 9 shows the average position error $E^r_{N_R}(t)$ for $N_R = 3$, with mutual information exchange allowed, and for $N_R = 6$, with and without mutual information allowed. The position error obtained with three robots is comparable with that with six robots and local communication allowed, and it is significantly reduced with respect to the experiment with six robots and communication not allowed. These results are comparable with those obtained in [9], where at 600 s, in absence of mutual exchange of information a final error lower than 4 m was achieved, while in our case it is less than 3 m. Considering instead the cases with mutual exchange of information, the final error is decreased to less than 1.2 m.
It must be noted that in both cases ultrasound sensors were used, but the results reported in [9] are relative to eight robots and averaged over 8 experiments only.

The proposed algorithm shows only a slight improvement in the localization time with respect to that in [9]. From our point of view the main improvement given by our work is relative to the possibility of checking if robots are well-localized, i.e., if they all have reached the PT state. In particular in all these experiments *all* robots reach the PT state; the switching times are reported in Table 5. We notice again that the switching times on average decrease significantly passing from 3 robots to 6 robots. Considering in particular the experiment with 6 robots, the last switching time is approximately 500 s on average and at that time the average position error is below 1.5 m. A remarkable result is that even if the number of robots increases, they do not become overconfident about their belief. PT state is always reached when all the robots are correctly localized.

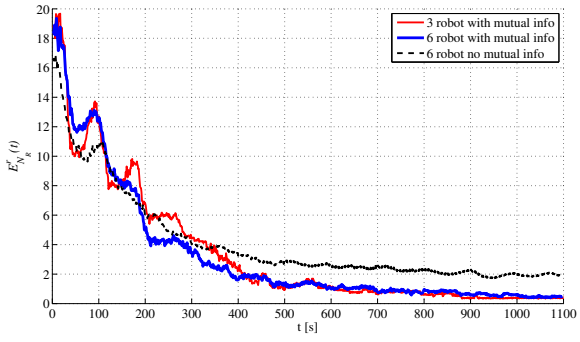

Fig. 9 Simulation test 4: average localization errors.

Remark 2 In Subsection 2.3 we claimed that the number of particles employed to approximate the probability distribution of the position hypotheses is generally less than $N_{max} -$

Table 5 Simulation test 4: switching times (s)

	First	Last
3 robots	683	706
6 robots	454	522

N_{kld}. Figure 10 shows the results of the generic i-th robot in a single experiment, confirming such a statement.
The blue plot is the number of particles resampled at time t, the red plot is the maximum number of particles that could be used to approximate each element of $h_i(k)$ at each time k and the black plot is $\frac{N^i_{mkld}}{n_{hyp}}$ (see the algorithm in Figure 4), i.e., the actual number of particles that the *Mutual KLD* algorithm uses to approximate each position hypothesis.

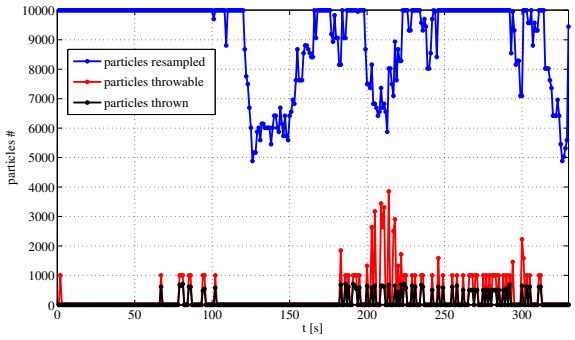

Fig. 10 Comparison among the number of particles employed to approximate the position hypotheses

3.5 Simulation test 5

We also tested the proposed algorithm in a realistic scenario, as shown in Figure 11. We chose the map of the interior of the Intel Research Lab in Seattle, available in the Radish dataset [3]. We define the average position error among the $N_R = 3$ robots of the team over the $r = 10$ realizations of the experiments as in (8). The results of the simulations are reported in Figure 12.

Results show that after approximately 230 seconds all the team members are correctly localized, even if a team of only three robots is used. This is due to the fact that the high number of asymmetries in this scenario, not relative to a logistic environment, is sufficient for the robots to correctly localize themselves even without exploiting multirobot information exchange.

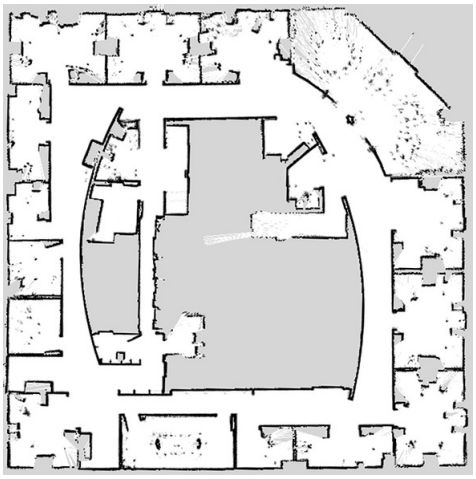

Fig. 11 The Intel Research Lab.

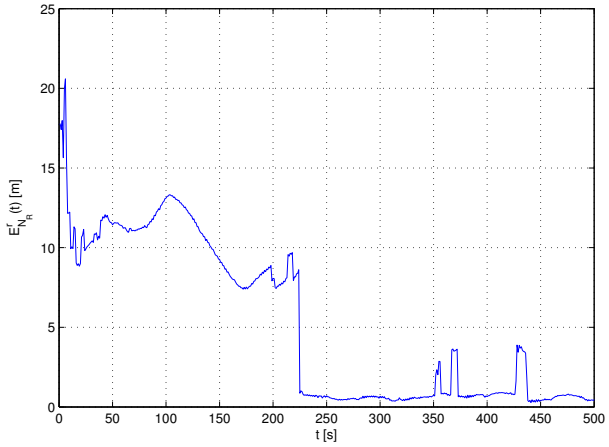

Fig. 12 Simulation test 5: average localization error.

3.6 Experimental tests

We also tested the proposed algorithm in the real world using real robotic platforms. The environment is 20 $\times$ 12 m and is characterized by only little asymmetries. We used a team of three Pioneer P3DX endowed with SICK LMS200 laser rangefinders and low cost monocular vision sensors. Each robot of the team is able to identify the others when they are in a certain field of view, using barcodes as identificators and a proper barcode detection algorithm. The detection is performed in two steps: we use the barcode detection algorithm to estimate the bearing of the detected robots, then we use the laser scanner to get an accurate measure of their distance. Figure 13 shows the robots team, while Figure 14 reports a grid

representation of the site where the experiments have been carried out, showing in particular the minor differences (asymmetries) that distinguish the sides of the environment. Odometry is used in the prediction phase of the filter, while range measurements obtained from laser rangefinders are used in the update phase. The ground truth to evaluate the localization error has been generated offline using a grid-based SLAM algorithm based on PMAP. This algorithm also outputs laser-corrected odometry, which is accurate enough to be used as ground truth for the trajectory of the robot. In both the experiments we use a maximum particle size of 10000 for each robot.

The first test (A) has been performed using only two robots, to show that a well localized robot can speed up the localization of the other that is still uncertain about its position. A robot is positioned near the main asymmetry of the environment (R1 in Figure 14), while the other one (R2) is positioned in a place where there are no discriminant features that can ensure fast localization (see the same Figure). The initial robot headings are indicated by the arrows.
The result of the test is reported in Figure 15, where the red plot is relative to robot R1 and the blue plot to robot R2. The sudden decrease in the localization error for robot R2 at

Fig. 13 The team of Pioneer P3DX robots.

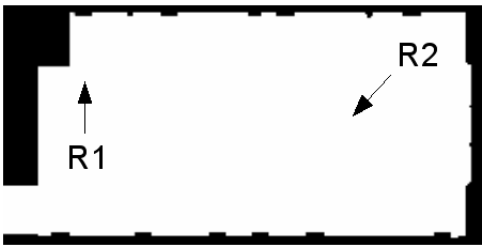

Fig. 14 The map of the site where the localization experiments have been carried out.

116 s corresponds to the moment at which the robots detect each other and their beliefs are

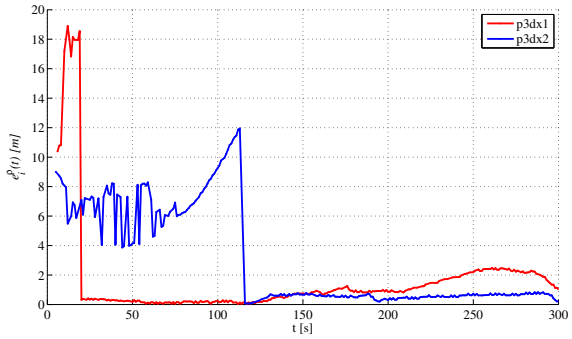

Fig. 15 Experimental test A: localization errors

mutually influenced. The overall result is that the well localized robot (R1) speeds up the localization of the other robot (R2). This simple example demonstrates that the mutual exchange of position information among robots using the proposed algorithm can significantly speed up the localization process in a real experiment.

The second test (B) is a classical kidnapping test. The two robots are randomly placed in the environment and, as shown in Figure 16, one robot (blue plot) is kidnapped approx-

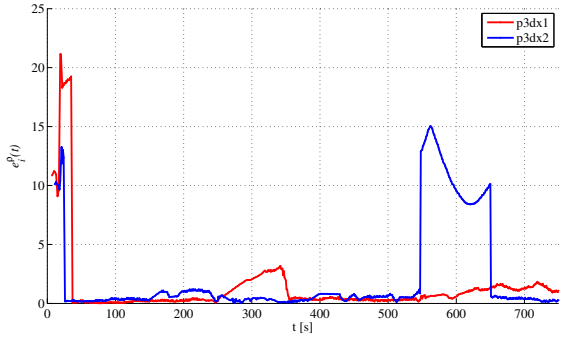

Fig. 16 Experimental test B: localization errors

imately at 500 s, and after 150 s is localized again by the other. Differently from test (A), which only demonstrates that localization can be speed up using the **3SMCL** algorithm, here we notice that the belief of a not well localized robot is influenced by the belief of a correctly localized one.

In the final test (C), we performed $r = 5$ mutual localization experiments using $N_R = 3$ robots randomly placed in the environment. Figure 17 shows the behavior of the average localization error, defined as:

$$E_{N_R}^{j}(t) = \sum_{i=1}^{N_R} \frac{e_i^{\rho}(t)}{N_R} \quad j = 1, \ldots, r. \tag{9}$$

In this experiment we observed that the first and last switching times were the same, so the

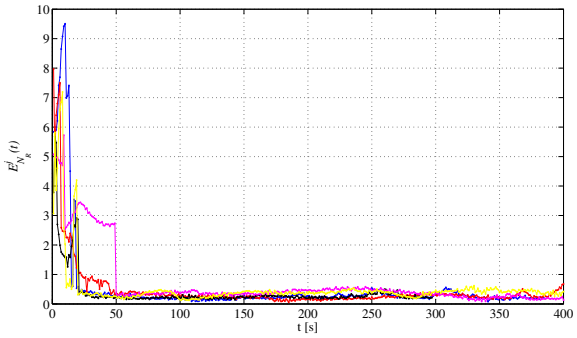

Fig. 17 Experimental test C: average localization errors

robots did not switch back from the PT state to the UN state. On average these switching times are around 350 s.

We also monitored the computational load of the algorithm running on each robot. The algorithm runs on an Intel Atom 1.6 Ghz with 2 GB of RAM. The CPU usage remains below 30% during global localization and falls below 10% during position tracking. The memory usage is about 30 Mb during the whole time.

4 Conclusions

The paper shows how the problem of correct localization of a team of robots can be successfully solved by the proposed **3SMCL** algorithm in highly symmetrical environments. Thanks to the cooperative action of all the robots of the team, the knowledge about the small asymmetries of the environment is quickly spread among the robots, since they can communicate their positions when they are in the field of view, allowing to remove the ambiguity on localization. Moreover the approach allows to avoid practical, ad-hoc solutions (e.g. the use of coded landmarks) to distinguish different regions of the area. The proposed solution can then be usefully adopted in practical applications, where a team of vehicles must autonomously move in an area characterized by a regular grid of corridors or streets. The robustness of the 3SMCL approach with respect to the initial position of the robots is an important advantage in practice, since the initial team formation can be completely arbitrary. Moreover, the automatic switch from the PT state to the UN state prevents the occurrence of macroscopical errors, due to temporary sensor failures or robot kidnapping. Experimental results in simulated and real symmetrical environments are reported in order to validate the proposed multirobot localization solution. It must be noted that the correct localization of the team is achieved if at least some of the robots detect an asymmetry before all the team is localized. Although it is difficult to estimate the number of robots which have to see an asymmetry, our simulation tests show that with a sufficient number of robots the probability of failure is very low. Future work will be devoted to the study of suitable active localization strategies and to an in-deep analysis of the propagation of measurement errors in the algo-

rithm. Finally an extensive study should be conducted on the effects of the symmetry of the map on the optimal number of robots and on localization performances.

Acknowledgements This work was supported by Regione Piemonte under the "MACP4Log" grant (RU/02/26)

References

1. Kiva systems. Website. `http://www.kivasystems.com`.
2. Mobilesim. Website. `http://robots.mobilerobots.com/MobileSim/`.
3. Radish: The robotics data set repository. Website. `http://radish.sourceforge.net`.
4. Seegrid. Website. `http://www.seegrid.com/`.
5. F. Abrate, B. Bona, M. Indri, S. Rosa, and F. Tibaldi. Switching multirobot collaborative localization in symmetrical environments. In *IEEE International Conference on Intelligent RObots Systems (IROS 2008), 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV)*, 2008.
6. F. Abrate, B. Bona, M. Indri, S. Rosa, and F. Tibaldi. Three state multirobot collaborative localization in symmetrical environments. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, pages 1–6, 7th May, 2009.
7. Sunghwan Ahn, Minyong Choi, Jinwoo Choi, and Wan Kyun Chung. Data association using visual object recognition for ekf-slam in home environment. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2588–2594, Oct. 2006.
8. D. Fox. Kld-sampling: Adaptive particle filters. In *In Advances in Neural Information Processing Systems 14*, pages 713–720. MIT Press, 2001.
9. D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000.
10. A. Gasparri, S. Panzieri, and F. Pascucci. A fast conjunctive resampling particle filter for collaborativemulti-robot localization. In *Workshop on Formal Models and Methods for Multi-Robot Systems*, Estoril, Portugal, may 2008.
11. B. Gerkey, R.T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th Int. Conf. on Advanced Robotics (ICAR 2003)*, pages 317–323, 2003.
12. D. Göring and H.-D. Burkhard. Multi robot object tracking and self localization using visual percept relations. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 31–36, 2006.
13. N. Karam, F. Chausse, R. Aufrere, and R. Chapuis. Localization of a group of communicating vehicles by state exchange. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 519–524, 2006.
14. A. Martinelli. Improving the precision on multi robot localization by using a series of filters hierarchically distributed. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1053–1058, 2007.
15. T Matsubara, M Kubo, and Y Murachi. Particple filter for collaborative multi-robot localization tollerant to recognition error. *Advanced Robotics*, 24(15):2043–2058, 2010.
16. M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino. Simultaneous localization and map building for a team of cooperating robots: A set membership approach. *IEEE Trans. on Robotics and Automation*, 19(2):238–249, 2003.
17. A.I. Mourikis and S.I. Roumeliotis. Performance analysis of multirobot cooperative localization. *IEEE Trans. on Robotics*, 22(4):666–681, 2006.
18. S. Panzieri, F. Pascucci, and R. Setola. Multirobot localization using interlaced extended kalman filter. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2816–2821, 2006.
19. M. Peasgood, C. Clark, and J. McPhee. Localization of multiple robots with simple sensors. In *IEEE Int. Conf. on Mechatronics and Automation*, pages 671–676, 2005.
20. G. Pillonetto and S. Carpin. Multirobot localization with unknown variance parameters. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1709–1714, 2007.
21. I. Rekleitis, G. Dudek, and E. Milios. Probabilistic cooperative localization and mapping in practice. In *IEEE Int. Conf. on Robotics and Automation*, pages 1907–1912, 2003.
22. S.I. Roumeliotis and G.A. Bekey. Distributed multirobot localization. *IEEE Trans. on Robotics and Automation*, 18(5):781–795, 2002.
23. C.J. Taylor and J. Spletzer. A bounded uncertainty approach to cooperative localization using relative bearing constraints. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2500–2506, 2007.
24. S. Thrun, W.Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.