Multirobot Object Localization: A Fuzzy Fusion Approach

Kevin LeBlanc and Alessandro Saffiotti, *Senior Member, IEEE*

***Abstract*—In this paper, we address the problem of fusing information about object positions in multirobot systems. Our approach is novel in two main respects. First, it addresses the multirobot object localization problem using fuzzy logic. It uses fuzzy sets to represent uncertain position information and fuzzy intersection to fuse this information. The result of this fusion is a consensus among sources, as opposed to the compromise achieved by many other approaches. Second, our method fully propagates self-localization uncertainty to object-position estimates. We evaluate our method using systematic experiments, which describe an *input-error landscape* for the performance of our approach. This landscape characterizes how well our method performs when faced with various types and amounts of input errors.**

***Index Terms*—Fuzzy logic, information fusion, multirobot systems, object localization.**

I. Introduction

ONE of the most important challenges in autonomous robotics is to accurately determine the state of the world. This knowledge is crucial to a robot's ability to appropriately and reliably perform actions. In particular, it is important for a robot to know the positions of objects relevant to its current task. In this paper, we address the problem of determining this information in a multirobot system.

The multirobot object localization problem can be seen as an instance of the more general information fusion problem. Combining information from different sources and/or different times, if done properly, yields more accurate estimates about the state of the world. In general, redundant information can improve accuracy and reliability, whereas complementary information can resolve ambiguities and incompleteness.

In single-robot object localization, information about object positions arrives at different times and/or from different sensors. In multirobot systems, more sensors are available, often with different points of view and different characteristics. This can virtually extend the overall system's field of view, as well as reduce the effects of sensor range and accuracy limitations.

The advantages of multirobot systems come at a cost. Unreliable communication channels and bandwidth limitations can make it difficult to share information. Furthermore, shared information must be represented in a common reference frame—normally, a global coordinate system. Object-position estimates in this frame will strongly depend on a robot's knowledge about its own pose in the frame.

This paper, which extends the work in [1], makes three main contributions. First, we propose a fuzzy-logic-based approach to representing and fusing object-position information in multirobot systems. Our method uses fuzzy sets to represent information, and we perform fusion using fuzzy intersection. This yields a consensus about the fused information, as opposed to the compromise produced by many other approaches. Fuzzy logic has successfully been used in many areas of robotics [2], including information fusion (e.g., [3] and [4]) and self-localization (e.g., [5] and [6]); however, the use of fuzzy logic to address multirobot object localization is new. Second, our method fully considers uncertainty in self-localization when converting local position estimates to global coordinates. As we discuss in Section II, most existing approaches assume perfect self-localization, which is often not achievable in robotic systems. Finally, we suggest a methodology for systematically evaluating our approach, where we characterize its performance with respect to various types of errors on its inputs. This analysis departs from the traditional approach to evaluating subsystems in robotics, where tests are performed only under typical conditions, and the subsystem under test is specifically tuned for those conditions. Our methodology allows us to see how a subsystem performs *outside of its comfort zone*; this can be useful when deciding what method to use for a given application. In addition to these contributions, we also present some approximations of our method, which reduce computational and bandwidth requirements if needed.

Often, fusion is combined with data association and prediction to perform tracking. In this paper, we focus only on the fusion process itself. To allow the performance of our fusion method to be measured in isolation, we assume that the identity of detected objects is known (which trivializes data association), and we assume that objects are static (hence, prediction is not needed).

The rest of this paper is organized as follows. In Section II, we discuss related work. Section III gives a brief overview of fuzzy sets and discusses how we use them to represent and fuse uncertain position information. In Section IV, we describe our overall framework for multirobot object localization, and in Section V, we discuss our implementation of the framework. Finally, in Section VI, we present and discuss the results of our experiments; we conclude with Section VII.

Manuscript received March 19, 2008; revised November 9, 2008. This work was supported in part by the National Graduate School in Computer Science (CUGS), Sweden. This paper was recommended by Associate Editor I. Bloch.

The authors are with the Center for Applied Autonomous Sensor Systems, Department of Technology, Örebro University, 70182 Örebro, Sweden.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2009.2015279

II. Related Work

Object localization is a fundamental challenge in robotics, and as such, many works address it. However, most methods rely on a relatively small number of underlying information fusion approaches. In this section, we give an overview of the fusion approaches that are most commonly used for object localization tasks in robotics. We also briefly discuss two related problems—the self-localization problem and the multisensor multitarget tracking problem.

Single-robot and multirobot approaches to object localization are often addressed using the same underlying fusion techniques. The main difference is that, in multirobot systems, sensors usually have unknown relative positions, which means that a common reference frame for position information is needed. This reference frame is normally a global coordinate system; such a reference frame is often used in other parts of the overall system as well. Estimating object positions in a global coordinate system normally requires that a robot should know its own pose in this frame, i.e., self-localization needs to be addressed. There are, however, a few approaches that avoid the need for global self-localization by using relative reference frames, for instance, [7] and [8]. Once object-position estimates have been situated in the common reference frame—a step that is normally nontrivial given self-localization uncertainty—single-robot and multirobot object localization problems can be addressed using similar methods.

Many works address the self-localization problem (e.g., [5], [6], [9], and [10]), and it is often combined with the mapping problem (a survey of this field is presented in [11]). The self-localization problem has some aspects in common with the multirobot object localization problem, and similar fusion methods are often used in both problems. However, a number of different challenges are involved when dealing with multirobot object localization. As such, we will only briefly touch upon the self-localization problem when we discuss the landmark-based fuzzy self-localization approach we use in this paper, which is based on [5]. A similar approach is described in [6]; this approach, which is also based on fuzzy logic, uses sonar scans instead of landmarks to determine possible robot poses.

Fuzzy logic is a popular tool for addressing information fusion problems [1], [3], [4]. For instance, [4] presents a general approach to information fusion using fuzzy logic. This general fusion approach is then applied to the single-robot self-localization problem as an example. The work presented here has similar foundations, both in terms of information representation [2] and information fusion. This paper aims to apply well-studied fuzzy information fusion techniques to the multirobot object localization problem. As we will show, such an approach has a number of built-in advantages.

Despite the popularity of fuzzy logic for addressing information fusion in general, many robotics applications rely on probabilistic fusion techniques. A large number of object localization methods are based on Kalman filters [12], [13] and linearized Kalman filters, like the extended Kalman filter [14], [15] and the unscented Kalman filter [16]. The Kalman filter algorithm can be seen as a continuous-space implementation of the Bayes filter algorithm [17], [18], in which information is represented using Gaussians. In general, such methods are accurate, easy to implement, and computationally efficient. They are used both to fuse information arriving at different times and to combine information arriving from different sources (e.g., [19]–[23]). Despite their widespread use, Kalman filters have a number of limitations. For one thing, they are unimodal. This limitation can be somewhat offset by multiple hypothesis tracking [24], [25], which allows multiple modes to be maintained in parallel. Another limitation of Kalman filter-based methods is that since information is combined using weighted averaging, fused results can significantly be degraded in the presence of false positives or outliers. Various forms of gating (e.g., [26]–[28]) can be used to reduce the effects of this limitation; however, this requires careful tuning.

Markov localization (e.g., [10] and [29]) is another probabilistic method often used in robotic localization tasks. The algorithm can be viewed as a discrete-space implementation of the Bayes filter algorithm [18]. The idea is to maintain a (possibly multimodal) discrete probability distribution over the state space—this is normally either grid-based or sample-based. New information increases the probability that an object is in a given region of the distribution and decreases the probability that it is anywhere else. The resulting distribution will typically have a higher probability in positions that are consistent with the majority of the fused information. Markov localization has been shown to be robust, although it is computationally more demanding than Kalman filtering, in general. There is typically a tradeoff between accuracy and computational load; sample-based methods, in particular, allow a smooth transition along the axis of this tradeoff. A hybrid method that combines Kalman filters with Markov localization, called Markov–Kalman object localization, has been shown to be very effective [30], [31].

The multisensor multitarget tracking problem is closely related to the multirobot object localization problem, except for the fact that, in the former, sensors normally have fixed positions. The vast majority of tracking approaches combine observations using either Kalman filtering or sample-based probabilistic filtering [22], [29], [32], [33]. Recent developments in this field mainly aim at improving performance with respect to data association and prediction. Data association, in particular, is known to be a computationally complex and difficult problem, and many approaches focus on this; [34] provides a survey of this field. Recall that, in this paper, we do not address data association or prediction.

We have discussed the most common approaches to fusion in multirobot object localization, all of which share one significant limitation: self-localization uncertainty is not considered. Most works explicitly assume perfect self-localization, which is not normally achievable in robotic systems. In particular, even small errors in orientation can cause large errors in object-position estimates.

Aside from the work that we are extending [1], we know of only one approach that explicitly addresses self-localization uncertainty when performing multirobot object localization [35]; in this sample-based approach, a small number of possible object positions are computed taking self-localization uncertainty into account, and these are exchanged between robots.

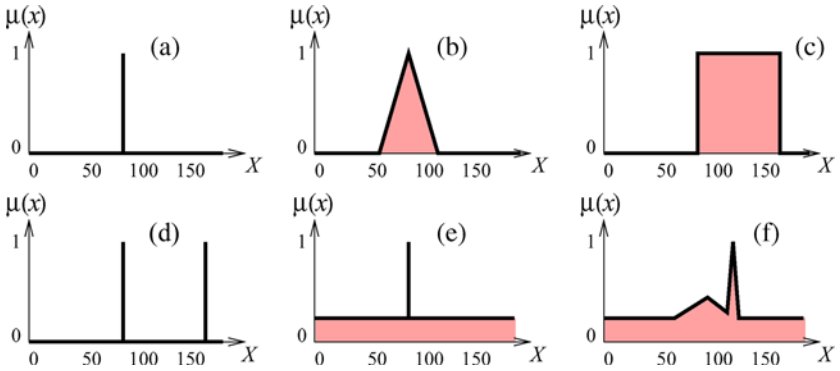

Fig. 1. Various types of uncertainty represented using fuzzy sets. Figure from [2], used with permission.

In a few other approaches, self-localization uncertainty is used to weight estimates of target object positions (e.g., see the arithmetic mean method in [21]; a similar idea is described, but not implemented, in [19]). The work in [36] demonstrates that considering reliability of sources, in general, can improve performance in both self-localization and object-localization tasks.

The method we propose uses fuzzy logic to compute a consensus between sources, and it does not suffer from the effects of averaging. Moreover, our method fully considers self-localization uncertainty when computing object-position estimates.

III. Fuzzy Position Information

In this paper, we use fuzzy sets to represent and combine information about the positions of robots and objects. Fuzzy sets provide a powerful and convenient way to represent and fuse possibly uncertain position information. This section gives a brief overview of fuzzy sets, which should allow the reader to understand how they are used in this paper.

A. Representing Fuzzy Position Information

Fuzzy sets were proposed by Zadeh [37] as a way of representing noncrisp concepts (e.g., tall or old) by allowing set elements to have degrees of membership. These degrees of membership are represented by real numbers in the [0, 1] interval. Given an element x belonging to the universal set X, one can denote the degree of membership of the element x to the set described by the concept A as $\mu_A(x)$. Mathematically, membership functions are defined as

$$\mu_A : X \rightarrow [0, 1]. \tag{1}$$

In this paper, we adhere to a *possibilistic* interpretation of fuzzy sets [38], [39], where $\mu_A(x)$ indicates the degree of possibility that x possesses the property A. Therefore, if A represents the position of an object, we read $\mu_A(x)$ as "the degree of possibility that object A is at position x." Note that under this interpretation, low possibility values actually provide more information than high values, as they rule out potential elements. In particular, complete ignorance is represented by a fuzzy set in which all elements have membership values of 1.0—in other words, all values are equally and fully possible. Conversely, the most informative distributions are those in which all elements have membership values of 0.0 except for one, which has a membership value of 1.0.

The use of fuzzy sets allows us to represent several different types of uncertainty in position information. The ability to represent information at precisely the level of detail at which it is available is often claimed to be one of the most compelling reasons to use fuzzy sets. Fig. 1 illustrates some of these uncertainty types. In Fig. 1(a), the position of the object is known with certainty to be 80. In Fig. 1(b), the position is approximately 80, and it is, therefore, *vague*. In Fig. 1(c), the position is between 80 and 160, and it is, therefore, *imprecise*. In Fig. 1(d), the position is either 80 or 160, and it is, therefore, *ambiguous*. In Fig. 1(e), the position is, with the highest possibility, at 80, but it is also possible that it is elsewhere (e.g., perhaps it has recently been seen at 80, but it might have been moved since then)—this *unreliability* is represented by setting a minimum value for all elements in the fuzzy set to a certain *bias* value. Finally, in Fig. 1(f), there are a number of different types of uncertainty combined.

Sometimes, it is useful to extract a point estimate $\hat{A} \in X$ from the information contained in a fuzzy set μ_A. This process is called *defuzzification* and can be done in a number of ways. One of the most common defuzzification techniques is to use the center of gravity (CoG) of the fuzzy set μ_A, which is computed according to the following:

$$\hat{A} = \frac{\int_{x \in X} x \mu_A(x) dx}{\int_{x \in X} \mu_A(x) dx}. \tag{2}$$

B. Fusing Fuzzy Position Information

There are a number of operations that can be performed on fuzzy sets; some of the most common are intersection, union, and complementation. In this paper, we focus on intersection, which can be used to *fuse* information represented in fuzzy sets. Intersection of fuzzy sets is normally defined as

$$\mu_{A \cap B}(x) = \mu_A(x) \otimes \mu_B(x) \tag{3}$$

where $\otimes$ denotes a triangular norm or t-norm [40]. T-norms are binary operators that are commutative, associative, and nondecreasing (if $x \leq y$, then $x \otimes z \leq y \otimes z$) and have 1 as the neutral element $(x \otimes 1 = x)$. The most commonly used

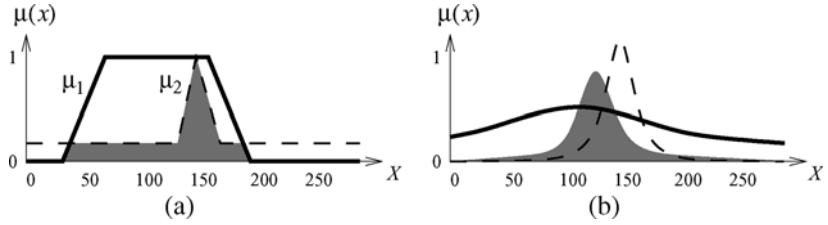

Fig. 2. (a) Fuzzy fusion computes a *consensus* between two sources of information, as opposed to (b) averaging approaches, which compute a tradeoff.

t-norms are the minimum, the product, and the Łukasiewicz operator $\max(x + y - 1, 0)$.

The various t-norms behave differently; the choice is typically domain dependent. For instance, the minimum t-norm is idempotent $[\min(x, x) = x]$. Therefore, this operator is often used when the independence of sources cannot be assumed. This is because fusing the same information multiple times using an idempotent operator yields the same result as fusing it only once. Therefore, it does not matter if the sources are dependent—which means they could be providing different versions of the same information.

In this paper, we typically use the product t-norm, which is nonidempotent. The product should be used when independence is granted since it acts as a reinforcing operator; specifically, it reinforces belief in values that are deemed possible by all sources. See, for example, [4] for more details.

As an example, imagine two sources that both report that $\mu_A(x) = 0.5$ and $\mu_A(y) = 1.0$. Recall that under the interpretation of fuzzy sets used here, $\mu_A(x)$ is the possibility that element x has property A. If the sources are *dependent*, then they are potentially merely repeating the same information. For instance, the information might reflect the beliefs of two people who have read about an event in the same newspaper. In this case, the information should be combined using the idempotent minimum t-norm, which will result in the combined belief being the same as the inputs $\mu_A(x) = 0.5$ and $\mu_A(y) = 1.0$. Since the sources are not independent, there is no reinforcement effect, although the sources agree.

On the other hand, if the sources are *independent*, then they are not merely repeating the same information. For instance, the information might reflect the beliefs of two people who have both witnessed an event. In this case, we can apply a reinforcing t-norm, like the product, which will result in a combined belief of $\mu_A(x) = 0.25$ and $\mu_A(y) = 1.0$. This belief is *stronger* than any of the individual ones because it narrows the set of possibilities more sharply. In this sense, the two opinions have been *reinforced*.

There are two important general facts about fuzzy fusion that should be noticed. First, only values that are regarded as possible by all sources are retained in the result. This can be seen in the simple example shown in Fig. 2(a). In the figure, the result of the fusion of fuzzy sets μ_1 and μ_2 using the minimum t-norm is indicated by the shadowed area. Intuitively, the result of fuzzy fusion represents a *consensus* between sources of information. This contrasts with techniques that use averaging, which yield a tradeoff. Fig. 2(b) shows how two items of information similar to the ones shown in Fig. 2(a) might be represented and fused in an averaging approach. Notice that with fuzzy fusion, the peak of the resulting distribution coincides with the peak of μ_2 since this is compatible with the peak of μ_1; it lies in between those peaks when using averaging.

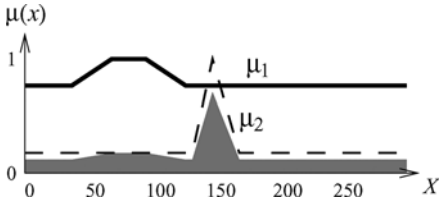

Fig. 3. Discounting unreliable information in fuzzy fusion. Information μ_1 is unreliable, as indicated by the high bias, and, therefore, only has a small influence on the result of the fusion.

The second fact to note is that fuzzy fusion automatically discounts unreliable information. Consider Fig. 3. The information represented by μ_1 includes a high bias (0.8), indicating that this information is unreliable; the information represented by μ_2 has a small bias (0.1). Correspondingly, the result of the fusion (here using the product t-norm) is similar to the information in μ_2, and it is only marginally influenced by μ_1. Fuzzy fusion minimizes the impact of unreliable information, provided this unreliability is correctly represented.

IV. Object Localization Framework

A. Problem Formulation

Our framework assumes that we have a set of M robots, denoted $\{r_1, \ldots, r_M\}$, with $M > 1$, which are able to communicate in some way. We also assume that a global reference frame for position information is available and known to all robots. A robot's knowledge about its own position in this reference frame may be uncertain. This reference frame could be dynamic; however, in this paper, we use a fixed global coordinate system. Positions in the global frame are denoted (x, y), and poses, which include orientation, are denoted (x, y, ϕ).

Each robot r_i maintains a belief distribution about its own pose in the world. We make no assumptions about how this distribution is created or maintained; we simply require that, at all times, each robot can determine, for any pose (x, y, ϕ), how much it believes this to be its true pose. In practice, we represent self-localization information as we represent other types of position information: using a fuzzy set, which indicates the possibility of a given pose being a robot's true pose. In Section V-B, we will briefly describe the landmark-based approach to self-localization used in this paper.

We assume that robots can observe named objects in the environment, and that the identity of an observed object is

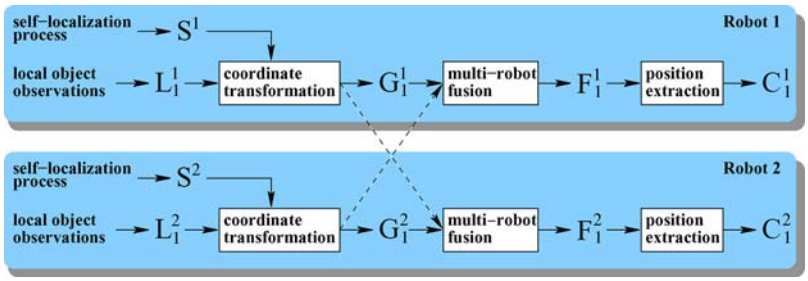

Fig. 4. Overall schema of our multirobot object localization method.

known. An observation includes an estimate of the object's position relative to the robot, as well as some associated uncertainty (e.g., a sensor model). In this paper, we use range and bearing measurements to represent observations, denoted (ρ, θ), and we use separate sensor models for the range and bearing components. We do not consider the orientation of observed objects, although an extension of our framework to include this would be straightforward.

B. Overall Schema

The overall schema of our method for multirobot object localization is graphically represented in Fig. 4 for a case with two robots sharing information about one object. In general, there are five main items of position information that we need to represent in each robot i. Note that each of these is time varying; we omit the time indexes for simplicity.

- S^i A 3-D fuzzy set representing robot i's *self-localization* estimate, in global coordinates. For any pose (x, y, ϕ), $S^i(x, y, \phi)$ measures robot i's belief that it is at that pose.
- L^i_j A 2-D fuzzy set representing robot i's *local object-position* estimate for object j, in polar coordinates. For any (ρ, θ), $L^i_j(\rho, \theta)$ measures the possibility that object j is at distance ρ and bearing θ with respect to robot i. Note that single-point, range-only, and bearing-only estimates can all be represented in this fuzzy set. A single-point estimate $(\hat{\rho}, \hat{\theta})$ is represented by setting $L^i_j(\hat{\rho}, \hat{\theta}) = 1$ and $L^i_j(\rho, \theta) = 0$ elsewhere. A bearing-only measurement of $\hat{\theta}$ is obtained by setting $L^i_j(\rho, \theta) = 1$ if $\theta = \hat{\theta}$, and 0 otherwise. A range-only measurement of $\hat{\rho}$ is obtained by setting $L^i_j(\rho, \theta) = 1$ if $\rho = \hat{\rho}$, and 0 otherwise.
- G^i_j A 2-D fuzzy set representing robot i's *global object-position* estimate for object j, in global coordinates. For any position (x, y), $G^i_j(x, y)$ measures the possibility that object j is at position (x, y).
- F^i_j A 2-D fuzzy set representing the *fused object-position* estimate maintained by robot i for object j, in global coordinates. For any position (x, y), $F^i_j(x, y)$ measures the possibility that object j is at position (x, y), according to the fusion of robot i's own global estimate with those available from other robots.
- C^i_j A single (x, y) position representing the *crisp object-position* estimate maintained by robot i for object j, in global coordinates. This is extracted from F^i_j.

Our method consists of the following three main processing steps, which operate on the described items of information. These steps are individually carried out inside each robot. Note that the descriptions below do not specify when each step should be performed—this will be discussed in Section IV-E.

1) Coordinate Transformation: An observation of a target object j is represented in a local estimate L^i_j, which reflects the measured range and bearing, as well as the sensor models for each of these. This estimate L^i_j is transformed into a global estimate G^i_j via a fuzzy coordinate transformation, described in Section IV-C. This transformation considers the full self-localization distribution S^i when computing G^i_j; that is, all self-localization uncertainty is propagated to G^i_j. This is one of the main distinctive features of our method.

2) Multirobot Fusion: In this step, robot i combines its global estimate G^i_j with those of other robots. The fusion of fuzzy sets was briefly described previously. The details of the fusion step are discussed in Section IV-D. The result of the fusion is stored in F^i_j.

3) Position Extraction: When a point estimate of the position of object j is needed by robot i, e.g., for action planning or execution, it is extracted from the latest fused object estimate F^i_j. This estimate C^i_j is computed by taking the CoG of F^i_j according to (2).

Fig. 5 shows a graphical example of how the fuzzy sets previously described might look after a few observations. The fuzzy set in the top left corner for each robot is the self-localization estimate S^i; the middle fuzzy set is the global estimate G^i_j; the rightmost fuzzy set is the fused estimate F^i_j. The observation is shown in the bottom left corner. Note that robot 1's global estimate is not simply a translation of its self-localization estimate; this is due to the uncertainty in both the observation and the robot orientation. Also, note that robot 2 has much less uncertainty in self-localization; correspondingly, the fused estimate, which is identical for both robots, is very similar to robot 2's global estimate.

Next, we will describe in detail the coordinate transformation and multirobot fusion steps, which constitute the core steps of our approach. After that, we will describe how the steps are put together to form the overall framework.

C. Coordinate Transformation

The coordinate transformation step converts a local object-position estimate L^i_j into a global object-position estimate G^i_j, taking the self-localization estimate S^i into consideration. Note

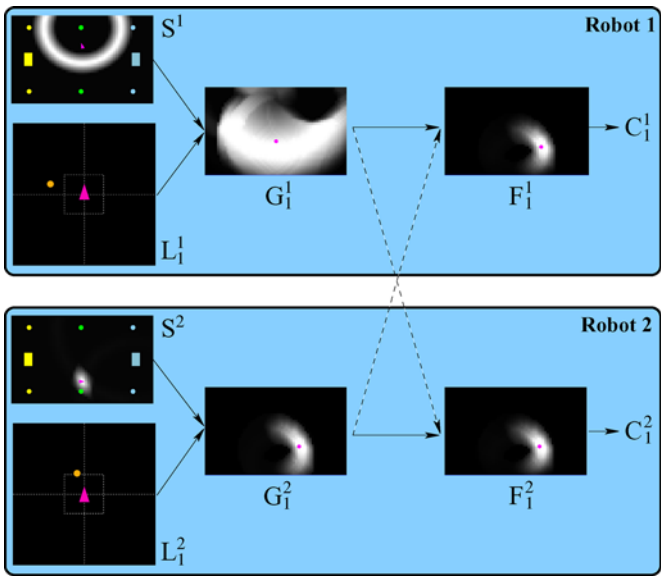

Fig. 5. Snapshot of our framework after a few observations, in a scenario with two robots. Lighter areas indicate possible positions; darker areas are less possible. We do not show the uncertainty in the range and bearing of the local estimates (bottom left for each robot). The resulting fused grids are identical since they result from the fusion of the same two global estimates.

that G_j^i is created based only on the information currently in L_j^i and S^i, and not from any previous information.

The transformation is not straightforward since neither the self-localization estimate nor the object observations are represented as points. Instead, the self-localization estimate is represented by the distribution in S^i, and the observation is represented by L_j^i, which includes uncertainty in range and bearing. It should be noted that one can address situations in which there is no uncertainty in S^i and/or L_j^i by using point estimates for these. This amounts to assuming perfect self-localization and/or a perfect sensor, respectively; our method transparently treats both cases.

We compute G_j^i as follows. Assume that robot i sees object j at range ρ and bearing θ. This is encoded in the local observation L_j^i, which takes uncertainty into account. Let $p = (x, y, \phi)$ denote an arbitrary 3-D pose for robot i—recall that ϕ represents the robot's orientation. Let $q = (x', y')$ denote an arbitrary 2-D position. Then, the possibility of object j being at position q according to robot i is given by

$$G_j^i(q) = \sup_p \left\{ S^i(p) \otimes L_j^i \left(\|\overline{pq}\|, \angle(\overline{pq}) - \phi \right) \right\} \qquad (4)$$

where $\|\overline{pq}\|$ denotes the length of the segment linking p to q, and $\angle(\overline{pq})$ denotes its orientation in the global frame. The overall distribution G_j^i is computed by calculating $G_j^i(q)$ for each possible position q in the global coordinate system; recall that we assume that this global coordinate system exists and is known to all robots. Therefore, the frame of discernment for the distribution G_j^i is the set of all possible positions. Each position gets a value in [0, 1], which reflects the possibility that the object is at that position. In Section V, we will discuss how we implement the set of possible positions as a 2-D grid.

Formula (4) can be explained as follows. The output of the formula is a measure of how possible it is that object j is at the 2-D input position q, given 1) the 3-D pose of robot i, represented by S^i, in global coordinates and 2) the 2-D range and bearing observation of object j made by robot i, represented by L_j^i, in local coordinates. The value $S^i(p)$ is the possibility that robot i is at pose p. The value of $\angle(\overline{pq}) - \phi$ is the observed bearing to the target with respect to ϕ, which is robot i's orientation in the global coordinate system. The value of L_j^i $(\|\overline{pq}\|, \angle(\overline{pq}) - \phi)$ reflects the possibility that robot i could observe object j at position q from pose p given the observation and associated uncertainty encoded in L_j^i.

The values $S^i(p)$ and L_j^i $(\|\overline{pq}\|, \angle(\overline{pq}) - \phi)$ are combined using a t-norm, as indicated by the symbol $\otimes$; in this paper, we use the product. We take the supremum of this combination for all possible poses p, as indicated by the $\sup_p$ operator. This means that the overall possibility of object j being at position q is based on the pose p, which yields the highest possibility of this being true. The graphical example in Fig. 5 shows, for two robots, their self-localization estimates S^i, observations L_j^i, and the resulting global estimates G_j^i.

The coordinate transformation can yield a distribution G_j^i in which no positions are fully possible. This can arise due to inconsistent information in the S^i and L_j^i distributions. For example, an object could be observed in a position that is outside the world. In such cases, we normalize the G_j^i distribution by shifting the values of all positions up until the most possible positions are fully possible. This is intuitive since there should always be some fully possible position. The fact that the normalized estimate incorporates inconsistent information is indicated by the fact that the minimum value of the fuzzy set, called the *bias*, is increased—which means that, to some degree, any position is possible.

It should be emphasized that transformation (4) preserves the full self-localization uncertainty from S^i and propagates it to G_j^i. In this respect, our approach is different from most existing approaches, where object positions are computed by assuming a point estimate for the location of the robot. Such approaches do not correctly propagate uncertainty in the robot's pose since they do not take into account the nonlinearities in the coordinate transformation—recall the example in Fig. 5. Moreover, they do not properly handle ambiguity (i.e., multiple modes) in the robot's self-localization. Our transformation (4) addresses both of these issues. The implementation of the coordinate transformation will be explained in detail in Section V.

D. Multirobot Fusion

Robots exchange their global estimates G_j^i, which are all based on the same global coordinate system. The estimates of M robots are fused together according to

$$F_j^i(x, y) = \bigotimes_{i=0}^{M} G_j^i(x, y) \qquad (5)$$

where $\otimes$ denotes the chosen t-norm operator. As mentioned in Section III, the choice of t-norm is often domain dependent.

For the fusion of position information, we use the product since it is a nonidempotent operator that reinforces belief in positions that are consistent with estimates from all robots. Recall Fig. 5 once again; note that the fused estimates contain only positions that are consistent with estimates from both robots. Also, recall that a distribution with a high bias value will have little effect on the result.

The result of the fusion process is separately stored from G_j^i, in F_j^i, to avoid circular dependencies. Since we are using a nonidempotent t-norm, it is important that the information in each estimate G_j^i only be considered once. Note that this operation is done on the full distributions—many other approaches summarize estimates before sending and fusing them, which can result in significant data loss. Moreover, observe that we are not considering the previous information in F_j^i when updating it; we simply compute, at every update step, the fusion of all individual estimates. We make no assumptions about the reliability of the sources other than to consider the uncertainty represented in the estimates they produce.

The fusion step can result in a distribution F_j^i in which no positions are fully possible. This can arise due to inconsistent information in the G_j^i distributions caused by errors in the individual robots' perception or self-localization. In these cases, we normalize the entire F_j^i distribution in the same way as we normalized global object-position estimates G_j^i, which contained inconsistent information. Again, this normalization increases the bias value of the overall estimate, indicating that, to some degree, any position is possible.

We do not test for agreement between new information and the previous F_j^i estimate since the previous estimate may have been incorrect. Nor do we test for agreement between robots since we do not know which robot, if any, is correct. Some approaches assume that the majority of sources are correct, but this does not always hold. Moreover, a majority cannot always be determined, e.g., when there are only two robots or when two equally large groups disagree.

If a robot is continuously sending incorrect information without representing this (i.e., without knowing it), the fusion process will continuously indicate that the result of the fusion is unreliable. Our method aims to yield a consensus between sources—if no such consensus exists, this information is also returned. We believe that it is crucial to clearly represent, rather than to hide, the fact that incoming information is inconsistent. This information can extremely be useful to the overall system; a high-level module might want to perform actions to verify unreliable estimates. The implementation of the fusion process will be explained in detail in Section V.

E. Putting It Together

So far, we have described the steps that we use to perform multirobot object localization. However, as in many distributed systems, it can be difficult to determine the proper order and timing for these steps. In our approach, we rely on an asynchronous event-based model, and we perform actions in response to various triggers. This makes the overall algorithm intrinsically decentralized and allows any successfully exchanged information to be exploited. If all robots successfully exchange all G_j^i fuzzy sets, all robots will have the same values for each F_j^i estimate since they are all fusing the same information. A robot does not differently treat its own object-position estimate G_j^i than the ones received from other robots. The following are the main trigger–action pairs, as seen from the perspective of a given robot i and target object j.

- *Target observed*. Whenever object j is observed by robot i, the local position estimate L_j^i is built based on the observation and the associated sensor models. The global estimate G_j^i is then built from L_j^i and the current self-location estimate S^i via coordinate transformation (4).
- *Estimates sent*. The global position estimate G_j^i is normally broadcast to all other robots whenever it is modified in response to an observation; however, the approach allows the frequency to be limited and/or the information to be compressed if bandwidth limitations are a concern.
- *Estimates received*. When a global position estimate G_j^h is received from another robot h, it is stored in a local cache for the pair (h, j), overwriting the last estimate sent by that robot for that object.
- *Target requested*. Whenever a global position estimate is requested, the latest values of G_j^h for all robots $h = 1, \ldots, M$, including robot i itself, are combined according to (5), resulting in the fused estimate F_j^i. If a point position is requested, the CoG of F_j^i is computed per (2).

This list outlines the main events considered in our framework and the actions they trigger. There are a number of alternative ways in which the steps of our method could be put together to suit the requirements of a given domain. Depending on which resources are most limited (e.g., memory, computation, or bandwidth), various changes could be made. However, such issues do not affect the fusion process itself, which is the focus of this paper.

V. Implementation

A. Representing Fuzzy Sets

As mentioned previously, in our approach, we represent uncertain position information using fuzzy sets under a possibilistic interpretation. Two of the most common ways to implement fuzzy sets are the *bin model* and the *parametric model*. In the bin model, the universe of discourse is discretized as an array of "bins," usually (but not necessarily) using a fixed step size; each bin stores a number that represents the corresponding membership value. In the parametric model, a fuzzy set is represented by fixing the parameters of a corresponding parametric function. Parametric representations tend to allow more efficient storage and computation; however, they have limited representational power. In the implementation discussed here, we use both bin and parametric models.

The parametric models we use in our implementation are trapezoidal membership functions, described by the following parameters, graphically shown in Fig. 6.

- *Core center and core width*. The core is the set of values that all have the maximum degree of membership. A wider core means a less precise fuzzy set.

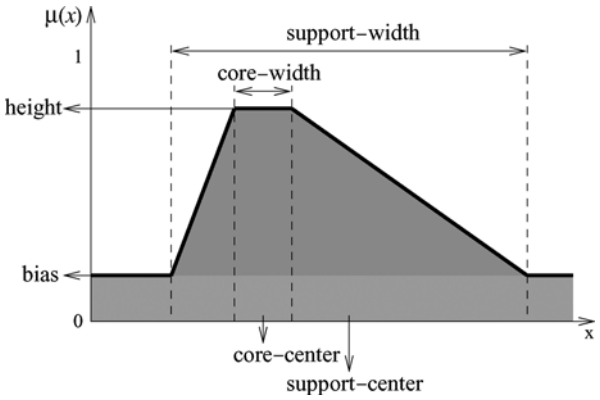

Fig. 6. Trapezoidal membership function.

- *Support center and support width*. The support is the set of values at the bottom of the trapezoid, and it contains the core. The larger the difference between the core width and the support width, the vaguer the fuzzy set. In this paper, symmetric trapezoids are used; therefore, the support center is the same as the core center.
- *Height*. The maximum value of the membership function.
- *Bias*. The minimum value of the membership function.

Trapezoidal fuzzy sets are often used in fuzzy logic, although, normally, the height is 1, and the bias is 0. In our possibilistic interpretation, a fuzzy set with height <1 indicates that no value is fully possible given the available information; a fuzzy set with bias >0 indicates that the information is not fully reliable, i.e., values outside the support of the fuzzy set are still possible. Notice that this representation can account for all of the types of uncertainty illustrated in Fig. 1, except for cases (d) and (f), which require multiple modes.

One thing we use parametric trapezoidal membership functions for is to encode sensor models. An observation sensor model is encoded by two trapezoidal fuzzy sets: μ_ρ and μ_θ, where ρ and θ are the measured range and bearing to the observed object. For μ_ρ, the height is set to 1, and both the core center and the support center are set to ρ. If the range accuracy of the relevant sensor inversely depends on distance, the core width and the support width of μ_ρ are set proportionally to the measured range ρ. Otherwise, they are given fixed values. For μ_θ, the height is set to 1, and both the core center and the support center are set to θ. The core width and the support width are fixed since the accuracy of a bearing measurement will typically not depend on its value. Both μ_ρ and μ_θ have a small bias, which accounts for the possibility of false detection. If necessary, this bias can be increased to indicate that a measurement is unreliable. Experiments have shown that our method is not particularly sensitive to the tuning of these sensor models; in general, intuitively chosen initial values perform well and need not be modified.

We use a bin model to represent position information in the global coordinate system. The 2-D position information contained in G^i_j and F^i_j is represented using discrete possibilistic grids, which represent a square tessellation of the 2-D space. Each cell in the grid has a value in the range [0, 1], which reflects the possibility that the object is located in that cell. To represent the robot's pose S^i, we also need to represent orientation. Using a 3-D grid would be quite expensive in terms of computation and storage. Instead, we use a 2(1/2)-D grid, where each cell c contains, instead of just a membership value, a trapezoidal membership function μ^i_c (see [5]). This function provides a parametric unimodal estimate of the orientation, which the robot could have if it were in cell c. Therefore, for any pose (x, y, ϕ), the possibility that the robot has that pose is obtained by computing the value of $\mu^i_c(\phi)$, where c indicates the cell corresponding to position (x, y). The height of the trapezoid in each cell corresponds to the overall possibility of the robot being in that cell, disregarding its orientation.

B. Self-Localization

In this section, we briefly describe the self-localization method we use in this paper. Recall that our approach to multi-robot object localization does not rely on this specific method; in principle, any self-localization method could be used.

Our self-localization process relies on the fuzzy grid-based approach proposed by Buschka *et al.* [5]. The approach assumes that a map of the environment is provided, which includes the positions of recognizable features (landmarks) in the environment. The process uses a 2(1/2)-D grid as previously described to represent self-localization information.

The self-localization process runs an infinite predict–update loop. Prediction takes into account robot motion, which, in our case, is estimated using odometry. The prediction consists of translation, rotation, and dilation of the S^i grid. The translation and the rotation are applied together, and the dilation is applied afterward to model the uncertainty in the odometric information. The transformations are implemented as fuzzy morphological operations [41]. In the update step, possible positions are computed based on observed landmarks, and these are intersected with the current estimate S^i. Landmark observations are encoded as normal object observations using the trapezoidal fuzzy sets for range and bearing, as described previously. Normalization is performed if S^i contains inconsistent information (i.e., no fully possible poses).

This fuzzy self-localization method has been shown to produce robust results in a highly dynamic domain characterized by significant sensor noise and unpredictable model errors. The method has also proven to be quite insensitive to sensor model tuning [5]. The approach has also been applied to domains with nonunique landmarks [9].

C. Full Object Localization Method

Here, we describe the main implementation of our method using the representations of fuzzy sets described previously. In Sections V-D and V-E, we will describe some approximations of our method, which can be used to reduce computational and bandwidth requirements, respectively, if needed.

1) Coordinate Transformation: When a target object j is observed at range ρ and bearing θ, the first step is to build the corresponding fuzzy set L^i_j. This simply involves setting the parameters of the two trapezoidal fuzzy sets—μ_ρ and μ_θ—as described previously, to represent the sensor model. Once L^i_j has been created, the fuzzy coordinate transformation defined

by (4) is performed to build the global object-position grid G^i_j. Algorithm 1 encodes this computation.

Algorithm 1. Fuzzy coordinate transformation.
Require: S^i = one trapezoid μ^i_c for each cell c.
Require: L^i_j = two trapezoids, μ_ρ and μ_θ.
Ensure: G^i_j
1. $G^i_j \leftarrow \mathbf{0}$
2. **for all** cell c such that height$(\mu^i_c) > \varepsilon$ **do**
3. $\quad \mu_{tmp} \leftarrow \mu^i_c$
$\quad$ core$(\mu_{tmp}) \leftarrow \max\{\text{core}(\mu^i_c), \text{core}(\mu_\theta)\}$
$\quad$ support$(\mu_{tmp}) \leftarrow \max\{\text{support}(\mu^i_c), \text{support}(\mu_\theta)\}$
4. $\quad$ **for all** cell q such that $\mu_\rho(\|\overline{cq}\|) > \text{bias}(\mu_\rho)$ **do**
5. $\quad\quad G^i_j(q) \leftarrow \max\{G^i_j(q), \mu_{tmp}(\angle(\overline{cq}) - \theta) \cdot \mu_\rho(\|\overline{cq}\|)\}$
6. $\quad$ **end for**
7. **end for**
8. *normalize* G^i_j.

Step 1 sets the global estimate G^i_j to zero. In step 2, ε is a fixed threshold below which we ignore cells in S^i; this threshold is typically low (e.g., 0.1). This check allows us to skip iterations of the main loop, which we know will produce low possibility values. In step 3, we account for uncertainty in the bearing reading by setting a temporary trapezoid equal to μ^i_c (from S^i) and widening its core and support values to be at least as wide as the uncertainty in the bearing sensor model. This trapezoid is used to determine if the observation could have been made from cell c given its bearing. In step 4, we limit the cells we update to those that are at a distance, which is consistent with the range reading. These cells are on an annulus with radius $r = \|\overline{cq}\|$ and width $w = \text{support}(\mu_\rho)$. This check can efficiently be implemented using Bresenham's circle drawing algorithm [42]. We iteratively run the algorithm for radii between $r - (w/2)$ and $r + (w/2)$. Step 8 normalizes G^i_j if there are no fully possible values.

The computational complexity of the full coordinate transformation for one object is $O(ND)$, where N is the number of cells in S^i, and D is the number of cells in G^i_j, which are possible according to the range sensor model (step 4). Since we use grids of the same size for S^i and for each G^i_j, the worst case computational complexity is $O(N^2)$.

2) Multirobot Fusion: When robot i requires a global position estimate for object j, the cached grids G^h_j of all robots, including robot i itself, are combined according to (5). Combination is performed cell by cell in a single scan, and the result is stored in F^i_j. In our implementation, we use the product t-norm to reinforce belief in positions that all robots consider possible. Recall that if there are no fully possible positions in F^i_j, the grid is normalized.

The computational complexity of this step for one object is $O(NM)$, where N is the number of cells in a grid, and M is the number of robots. The memory needed for storing the grids from all robots is $O(NM)$ cells. Note that the computation is individually carried out in each robot. Since the computational and memory costs are both linear in the number of robots and in the number of objects, the fusion step scales quite well.

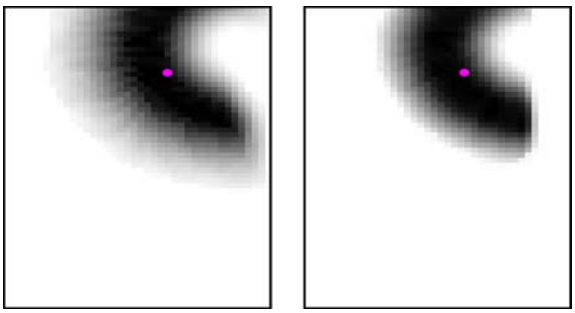

Fig. 7. Distribution on the left is computed using the full coordinate transformation; the one on the right is computed using the approximate coordinate transformation.

3) Position Extraction: A point estimate C^i_j can be computed from F^i_j using the CoG per (2). We consider only parts of the grid above a dynamic threshold, which is computed as a function of the *bias* of the distribution: the higher the bias, the higher the threshold.

D. Approximate Coordinate Transformation

Here, we describe an approximation of the coordinate transformation step, which allows global position estimates G^i_j to be derived using less computation than the full method.

The cost of algorithm 1 critically depends on the width $w = \text{support}(\mu_\rho)$ of the trapezoid representing range uncertainty. One way to reduce the complexity of the algorithm is to ignore range uncertainty in the algorithm itself and introduce an approximation of it *a posteriori* by performing a blurring operation on the resulting G^i_j grid. The approximation still considers the full uncertainty in the self-localization grid, as well as uncertainty in the bearing of the observation; it approximates only the range uncertainty in the observation. As the observation range uncertainty is increased, the accuracy of the approximation decreases. However, we have experimentally verified that, in the domain considered in this paper, using the approximation has a negligible effect on the results achieved by our method.

As an example, see Fig. 7, where two G^i_j grids produced using the same data are shown; the grid on the left was created using the full algorithm, and the one on the right was created using the approximate algorithm. Note that the uncertainty in the approximated grid is slightly less prominent in the horizontal direction; this reflects the fact that the approximation does not consider the full range uncertainty.

The approximated coordinate transformation can be implemented using algorithm 2. The approximate algorithm differs from algorithm 1 in two ways. First, in step 4, we only consider cells where $\|\overline{cq}\| = \rho$; these cells lie on a circle of radius ρ. We can quickly find these cells using a single iteration of Bresenham's circle drawing algorithm [42] as opposed to the multiple iterations used in the full algorithm. Second, in step 8, we apply a fuzzy morphological dilation operation to blur the G^i_j grid by an amount proportional to core(μ_ρ). This operation is meant to approximate the range uncertainty in the observation, which was ignored in step 4.

Algorithm 2. Approximate fuzzy coordinate transformation.
Require: S^i = one trapezoid μ_c^i for each cell c.
Require: L_j^i = two trapezoids, μ_ρ and μ_θ.
Ensure: G_j^i
1. $G_j^i \leftarrow \mathbf{0}$
2. **for all** cell c such that height$(\mu_c^i) > \varepsilon$ **do**
3. $\mu_{tmp} \leftarrow \mu_c^i$
   core$(\mu_{tmp}) \leftarrow \max\{\text{core}(\mu_c^i), \text{core}(\mu_\theta)\}$
   support$(\mu_{tmp}) \leftarrow \max\{\text{support}(\mu_c^i), \text{support}(\mu_\theta)\}$
4. **for all** cell q such that $\|\overline{cq}\| = \rho$ **do**
5. $G_j^i(q) \leftarrow \max\{G_j^i(q), \mu_{tmp}(\angle(\overline{cq}) - \theta) \cdot \mu_\rho(\|\overline{cq}\|)\}$
6. **end for**
7. **end for**
8. dilate G_j^i by an amount proportional to core(μ_ρ)
9. *normalize* G_j^i.

The computational complexity of the approximate coordinate transformation is $O(CN + KN)$, where N is the number of cells in S^i, C is the number of cells on the circle around the robot, which has a radius equal to the observed bearing ρ, and K is the size of the structuring element used in the dilation. Since the number of cells C can grow at most as $\sqrt{N}$, and since K does not depend on N, the asymptotic complexity of algorithm 2 is $O(N\sqrt{N})$. In our experience, the approximate algorithm typically requires significantly less time to execute than the full algorithm, particularly for observations with much range uncertainty.

E. Global Object Grid Approximations

Here, we describe three approximations of the global position grids G_j^i. These approximations reduce the amount of bandwidth needed to exchange these grids.

Normally, all robots exchange their full G_j^i object grids. If these grids have N cells, then the (uncompressed) size of each message is BN bits, where B is the number of bits used to represent the possibility value in each cell of G_j^i. We typically use 1 B per cell. One simple way to reduce bandwidth is to represent possibility values at a coarser resolution; that is, one can use fewer bits per cell. Obviously, the fewer bits are used, the less accurate the approximation is. In the next section, we show results obtained using various resolutions.

Another way to approximate the G_j^i object grids is to use a parametric representation of the distribution. We have experimented with two such representations, both of which are unimodal. The first is a bounding box of the area of the grid, which contains all possibility values greater than a certain threshold (we typically use 0.8). The bias of the grid (the minimum value) can be sent along with the bounding box, thus tagging the bounding box with a measure of reliability. We normally use 8 bits to represent the bias value, and we mark the four corners of the bounding box using four 32-bit integers. Therefore, each grid can be sent using a fixed size of $(8 + 4(32)) = 136$ bits. Note that this does not depend on the size of the grid.

The second parametric approximation we tested consists of two bounding boxes—one at a higher threshold (e.g., 0.8) and one at a lower threshold (e.g., 0.2). These boxes can be seen as representing the core and support of a 2-D trapezoid, which approximates the discrete distribution in G_j^i. The bias of the distribution is sent in this case as well. The resulting message size is $(8 + 8(32)) = 264$ bits. Again, note that this does not depend on the size of the grid. Results using both parametric representations will be shown in the next section.

VI. Experiments

In this section, we report the results of experiments performed using our method on a team of three robots sharing information about the location of a static ball. These experiments are performed with three goals in mind: 1) to empirically demonstrate the validity of our approach to multirobot object localization; 2) to determine the types of situations in which our method performs best; and 3) to quantify the degradation in performance introduced by using the approximations of the global position grids, discussed previously.

A. Methodology

When we first tried to quantitatively evaluate our method, we noticed that performance greatly varied from one experimental setup to another. By experimental setup, we mean the platforms, sensors, and experimental conditions used (e.g., lighting). This prompted a more systematic analysis of our method, the results of which are presented here. The analysis is intended to describe an *input-error landscape*, which shows how the performance of our method varies as different types and amounts of errors are introduced on its inputs. The input variables that we consider are the self-localization distribution and the range and bearing values of target object observations.

The analysis is empirically done by independently introducing increasingly large amounts of artificial errors on each input variable. To introduce errors on the target observations, we corrupt the measured range and bearing values. To introduce errors in self-localization, we corrupt the landmark observations used by the self-localization algorithm. We consider three types of input errors—systematic errors, random noise, and false positives.

The artificially corrupted data are based on real data, recorded in real time using a number of experimental layouts. The data are first idealized offline; in other words, range and bearing measurements to both landmarks and target objects are set to reflect the ground truth. Various types and amounts of artificial input errors are then introduced. In these systematic experiments, the robots and the target object are static; recall that this is to allow us to examine the performance of the fusion process in isolation without the influence of filtering or prediction.

In addition to this systematic analysis, we also present the results of one experiment that uses real data in a scenario where one robot is moving. As we shall see, the results of this experiment are consistent with the ones obtained using the systematic analysis; they reflect the performance of our method at one specific point on the input-error landscape.

Fig. 8. AIBO robot.

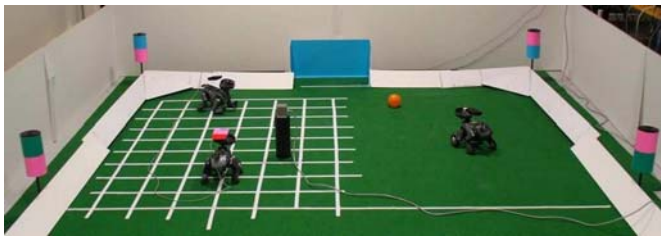

Fig. 9. Experimental environment.

B. Experimental Setup

1) Robots: The robots we used in our experiments are Sony AIBO ERS-210A [43] (see Fig. 8). These robots were used in earlier editions of the RoboCup competition [44]. Each robot has 32 MB of synchronous dynamic random access memory and a 64-bit RISC processor with a clock speed of 384 MHz. The robot's main sensor is a 100 000-pixel complimentary metal–oxide–semiconductor camera, mounted on the head. The robots communicate via wireless Ethernet.

Since the robots use legs instead of wheels, odometry is particularly unreliable mainly due to unpredictable slippage. The most serious errors in perception, for both landmark and target observations, occur because of the following reasons.

1) Range estimation is based on the size of an object in the camera image; hence, accuracy crucially depends on lighting conditions. Furthermore, the precision of range estimates quickly decreases with distance.
2) Bearing precision is limited due to uncertainty in the position of the camera; specifically, the pan joint position estimate often contains errors.
3) False positives and false negatives are relatively frequent due to errors in color segmentation caused by the camera's low resolution and high sensitivity to lighting.

These errors do not affect the systematic analysis since, in this analysis, the sources of errors are artificial—we isolate various types of errors, rather than their sources. However, these sources of error should be kept in mind for the online experiment, which uses real data.

2) Environment: The environment is an area of approximately 3 × 5 m, with eight unique landmarks. The setup is based on one of the playing fields used in earlier editions of the RoboCup competition. All objects of interest are color coded; this allows the data-association problem to be solved relatively easily. A photo of the setup is shown in Fig. 9.

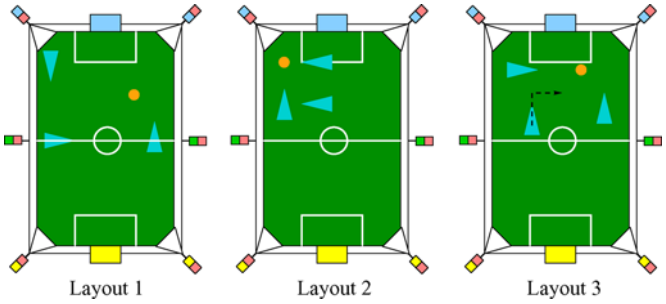

Fig. 10. Experimental layouts used in our experiments.

For all grids, we use a spatial resolution of 100 mm; the precision of our method is limited by this choice. We have verified that using finer resolutions does not significantly affect the results of our experiments. In real situations using the experimental setup described here, errors in estimation due to odometric and perceptual errors are generally much larger than 100 mm. This resolution choice, along with the size of the environment, means that there are approximately 30 × 50 cells in the position grids. Using a maximum resolution of 8 bits per cell, a full position grid has a size of 1.5 kB; this could be reduced using compression if needed.

In all the experiments, there are three robots and one static ball, which is the only target object. During a given run, the robots use the gaze control strategy described in [45] to keep both the target and landmarks under observation. The three experimental layouts we used in the presented experiments are shown in Fig. 10. Other layouts were tested, but the results did not significantly vary from one layout to another. We use the first two layouts, in which the robots are static, for the systematic analysis. The third layout is used in the experiment with real data; in this case, there is one moving robot. The approximate path taken by the robot is shown by the dotted line.

3) Ground Truth: The true positions of static robots and objects are measured for each experimental layout. In layout 3, the moving robot's pose is determined using a Polhemus Fastrak 6D magnetic position tracker [46] mounted on the robot's back. The error of this tracker in our setup is less than 10 mm, which is sufficient for our purposes. The reference point for this tracker can be seen in Fig. 9.

4) Performance Metrics: The performance metric we use for all our experiments is the distance (in millimeters) between the fused object-position estimate, which is based on information from all three robots, and the ground-truth position of the target object, which is known in advance for each layout.

C. Software Setup

The architecture we use on the robots is a modular layered architecture, loosely based on the Thinking Cap software architecture [47]. Our multirobot object localization method was initially implemented within this architecture and run onboard the AIBO robots. For the experiments presented here, the software has been separated into two parts—an on-board part and an off-board part. The on-board and off-board modules communicate via wireless Ethernet.

The on-board software includes modules for perception and motion control. The perception module performs object recognition based on color segmentation [48], and for each observed object, it returns its ID together with its measured range and bearing (ρ, θ). The perceptual module also takes care of gaze control, determining where and when to look for landmarks and target objects [45]. The motion control module sends motion commands to the low-level controller of the robot and returns an estimate of the motion based on odometry. Recall that odometric information is typically very poor on the legged AIBO robots.

The off-board software consists of a customizable tool that implements our multirobot object localization method and allows data to be logged, processed, and analyzed in a number of ways. We log motion updates, landmark observations, and target object observations. These logs are used both in the experiments using artificially corrupted data and those using real data. For the systematic experiments, the tool is also used to create ideal data. These data preserve event ordering and timing; however, they contain modified range and bearing measurements, computed from ground-truth information. The tool also allows the various types of input errors previously discussed to be applied to the data.

When processing the data in our experiments, we ensure that all grids are computed as soon as new relevant information is available. Therefore, whenever robot i observes a target j, an updated global position grid G_j^i is built. Also, whenever any of the robots updates its global position grid, this is shared with the other robots, and an updated fused grid F_j^i is computed immediately. This basically means that we do not limit the frequency at which global position grids are created or transmitted to other robots, and we assume that global position estimates are constantly being requested. In the off-board implementation used for the analysis, this is easily achieved since the logged data are processed from within the tool, which maintains the estimates for all robots within the same process. In the on-board implementation, this would require that enough bandwidth be available to transmit the grids as often as target observations are made. For the case with three robots observing one ball, this would easily be achievable given the bandwidth available from wireless Ethernet.

D. Evaluated Methods

We have computed and compared the results of fusing information using eight different methods:

- 8BPC-exact: our fuzzy fusion method, using the exact coordinate transformation algorithm 1 and 8 bits per cell to represent possibility values in G_j^i;
- 8BPC: same as previous, using the approximate coordinate transformation algorithm described in Section V-D;
- 4BPC: same as previous, using 4 bits per cell;
- 2BPC: same as previous, using 2 bits per cell;
- 2BB: same as previous, using the two bounding box approximation of G_j^i;
- 1BB: same as previous, using the single bounding box approximation of G_j^i;
- IWAVG: an ideally weighted average, used as a reference method;
- AVG: a nonweighted average, used as another reference method.

As mentioned previously, the 8BPC-exact and 8BPC methods produced very similar results; the 4BPC and 2BPC methods also performed similarly. To keep the graphs readable, and to avoid confusion between methods with similar performance, we will only report the results achieved using the following six methods: 8BPC, 2BPC, 2BB, 1BB, IWAVG, and AVG.

IWAVG and AVG are reference methods with which we compare our approach. In evaluating a robotic system, the choice of reference methods is always a delicate issue. Rarely do reference methods reflect the latest and best alternative methods since the implementations of these are often difficult to achieve, nor are the reference methods implemented with as much care and expertise as the methods under test. We attempt to minimize the effect of using an imperfect reference method by using methods that reflect the upper and lower bounds of the results achievable by all *averaging-based* methods, which address uncertainty by averaging the results from multiple sources. Averaging-based methods encompass an important subset of object localization methods. In particular, since we assume static targets, methods based on Kalman filtering would essentially compute a weighted average of observations since the motion model would predict no motion.

For both reference methods, we first compute an estimate of the target object's position according to each robot. This is done by taking the CoG of the self-localization distribution for each robot, and from there, finding the position that corresponds to the observed range and bearing to the target. We use the center of the sensor models for both range and bearing in this computation. The estimates from each robot are then averaged in x and y.

The lower bound reference method is a simple nonweighted average (AVG), which is the least informed way to use averaging. As an upper bound, we use an "ideally weighted average" (IWAVG), where weights are computed using ground-truth information. Note that this method represents an upper bound on the performance achievable by averaging-based approaches; it is not an absolute upper bound on the performance of any alternative method. Also, it is important to keep in mind that the IWAVG method produces results that would not be achievable in a real system since robots obviously do not have access to the ground-truth information used to compute the idealized weights.

The weights in the IWAVG method are computed as follows. Consider M robots observing a target, where robot i observes the target at p_i. The fused estimate p obtained by IWAVG is given by

$$p = \frac{\sum_{i=1}^{M} w_i \cdot p_i}{\sum_{i=1}^{M} w_i}. \tag{6}$$

Each weight w_i is computed by

$$w_i = \frac{\|\overline{p_i q}\|^{-1}}{\sum_{j=1}^{M} \|\overline{p_j q}\|^{-1}} \tag{7}$$

where q is the ground-truth position of the target, and $\|\overline{p_i q}\|$ denotes the distance between p_i and q.

E. Exploring the Input-Error Landscape

To explore the input-error landscape, we used idealized data corrupted by different types of artificially introduced input errors to span the different axes of our landscape. First, we logged 30–60 s worth of data from a number of runs of layouts 1 and 2 from Fig. 10. These data included all observations of landmarks and the target. Then, we ran algorithm 3 on the log files. The types of input errors considered at step 1 were systematic errors and random noise on both range and bearing measurements to the ball, false ball detection, and errors in self-localization. Self-localization errors were created by adding errors of all the previous types to landmark observations. The value of n in step 2 was 20 for the experiments presented here. The idealization at step 4 was performed using ground-truth information and only needed to be done once per log file.

Algorithm 3. Systematic analysis of the input-error landscape.
Require: set of all log files obtained using layouts 1 and 2.
Ensure: statistics about the input-error landscape.
1. **for all** type $\mathcal{T}$ of artificial input errors **do**
2. **for** $i = 0$ to n **do**
3. **for all** logfile F **do**
4. Ideal ← create idealized data from F
5. Corrupted ← corrupt Ideal by errors of type $\mathcal{T}$
6. Result ← process Corrupted logfile
7. **end for**
8. Compute statistics for this run of all logfiles
9. **end for**
10. Compute overall statistics for errors of type $\mathcal{T}$
11. **end for.**

Data corruption (step 5) was performed as follows. In our experimental setup, range estimates are mainly subject to multiplicative errors since we assume that range uncertainty increases as the range to the target increases. This assumption is justified by two observations. First, range errors usually originate from errors in object segmentation in the image; for instance, the width of the object in the image may be overestimated because the image is blurred due to camera motion. Second, an error of one pixel in object segmentation induces a range error that has a magnitude proportional to the distance to the object since we estimate the range to an observed object by comparing its size (in millimeters) in the real world and its size (in pixels) in the image, modulo the optical parameters of the camera. Given this, we computed an artificially corrupted range estimate ρ_c from an ideal range estimate ρ_i as follows:

$$\rho_c = \rho_i \cdot \left(1 + \delta^{\rho}_{\text{sys}} + \delta^{\rho}_{\text{ran}}\right) \tag{8}$$

where $\delta^{\rho}_{\text{sys}}$ and $\delta^{\rho}_{\text{ran}}$ are the percentages of systematic errors and random noise introduced, respectively. Random noise was uniformly distributed.

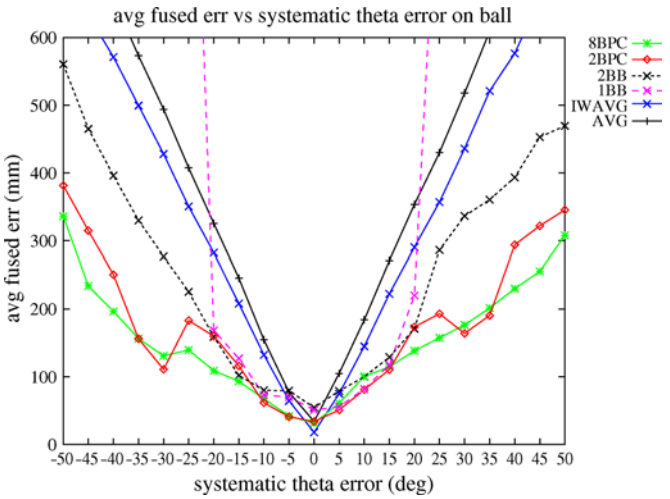

Fig. 11. Systematic bearing errors added to ball observations. Our methods perform considerably better than the best results achievable by weighted averaging, represented by the upper bound IWAVG method.

Bearing estimates are typically affected by additive errors, e.g., due to pan joint position uncertainty. As such, we computed an artificially corrupted bearing estimate θ_c from an ideal bearing estimate θ_i as follows:

$$\theta_c = \theta_i + \theta^{\theta}_{\text{sys}} + \theta^{\theta}_{\text{ran}} \tag{9}$$

where $\theta^{\theta}_{\text{sys}}$ and $\theta^{\theta}_{\text{ran}}$ are the amounts of systematic errors and random noise introduced, respectively.

False positives were introduced by replacing randomly chosen observations with random values within the measurement domain. A value δ_{fp} indicates the percentage of observations that were corrupted.

The rest of this section shows the results of applying algorithm 3 to the data collected in our scenarios.

F. Corrupted Target Observations

The results for the runs where input errors were introduced only on target observations are presented in Figs. 11–14. The graphs show the average errors in the fused object-position estimates for each method, averaged over 20 runs, versus the amount of introduced input error of the given type. Note that in these graphs, self-localization is based on perfect landmark observations. Therefore, the overall self-localization distribution in these cases was both precise and accurate; as such, the results shown here demonstrate the effects of input errors on the object observations only. Accordingly, the fact that our method considers self-localization uncertainty should have little or no effect on these results. The differences between our methods and the reference methods in these graphs show the effects of using fuzzy fusion as opposed to averaging-based methods.

In Fig. 11, the results for systematic bearing errors on ball observations are shown. Here, we see that our methods perform considerably better than even the upper bound IWAVG method. Even the most drastic approximations outperform the reference

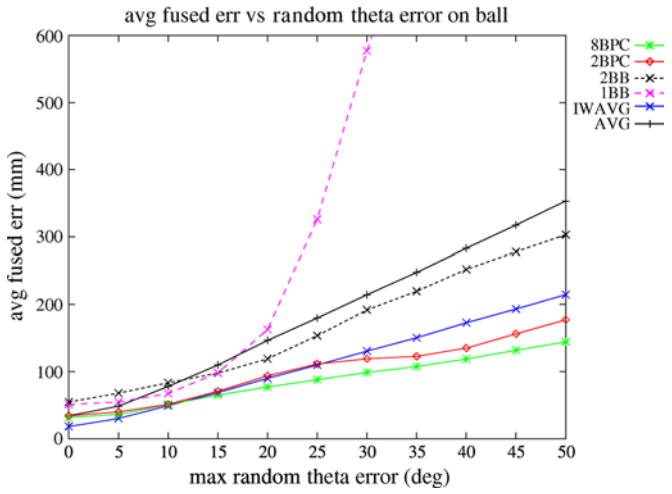

Fig. 12. Random bearing errors added to ball observations. Our methods perform slightly better than the upper bound IWAVG method, although the absolute values are similar.

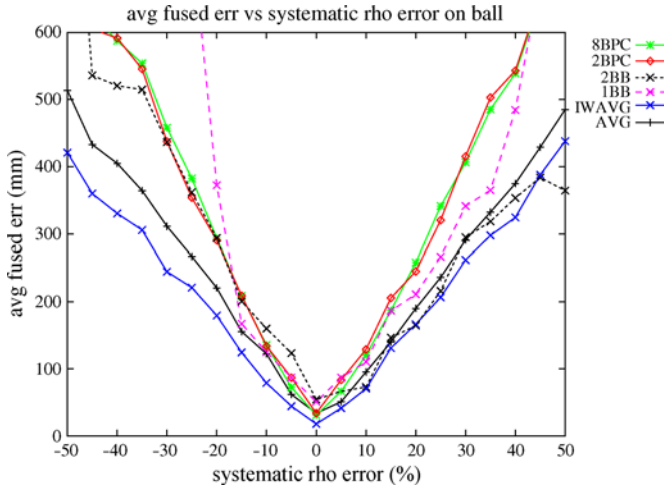

Fig. 13. Systematic range errors added to ball observations. The reference methods perform slightly better than our methods, although the absolute values are similar.

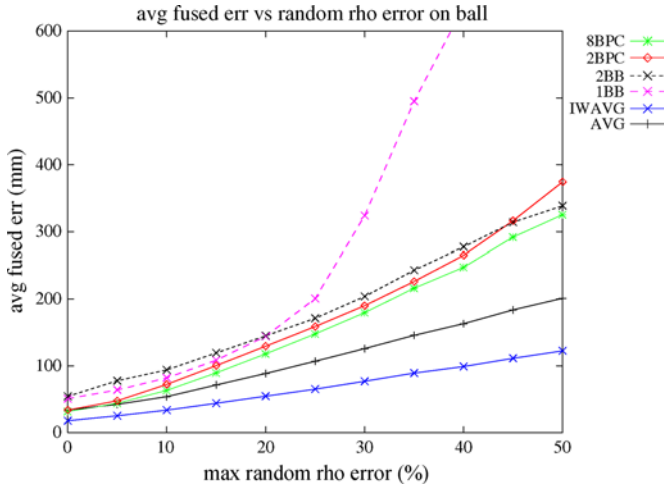

Fig. 14. Random range errors added to ball observations. The reference methods perform slightly better than our methods, although the absolute values are similar.

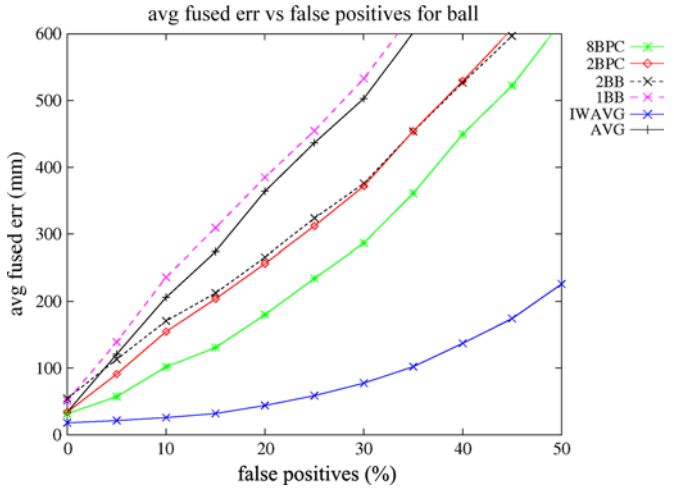

Fig. 15. False-positive ball observations added. The methods perform similarly, except for the upper bound IWAVG. Since this method has knowledge of the false positives (via the ground-truth information used to compute the weights), it obviously drastically outperforms other methods.

methods for relatively small bearing errors. In particular, note that the 2BB method, which requires very little bandwidth, consistently outperforms the reference methods.

Fig. 12 shows the results for random bearing errors on ball observations. Again, our methods outperform the reference methods, although to a lesser extent. In particular, the approximations suffer from the lack of consistency in the data, although the 2BPC method still performs well.

We see the results for systematic and random range errors on ball observations in Figs. 13 and 14. In these cases, even the lower bound AVG reference method outperforms all our methods, although the absolute values of the errors are fairly similar. Interestingly, all presented approximations perform nearly as well as the full method, except the 1BB approximation, which suddenly collapses when errors reach a fairly small threshold.

Fig. 15 shows the results obtained when some observations are replaced with false positives. In this case, the upper bound IWAVG method drastically outperforms the other methods. This is understandable since the ground-truth information used to compute the weights for the averaging gives the method effective knowledge about which observations are false positives. Our methods outperform the lower bound AVG reference method, however. Note that, again, the 2BB method performs quite well, matching the performance of the 2BPC method, which requires substantially more bandwidth to implement. Recall that we did not use any filtering in the experiments presented here, for a real application filtering would likely provide a substantial improvement in performance, particularly in the presence of false positives.

G. *Corrupted Landmark Observations*

Since changes to self-localization estimates result in nonlinear changes in object-position estimates, it is not informative to show the results of separately applying each type of input error to landmark observations. Instead, to examine the impact

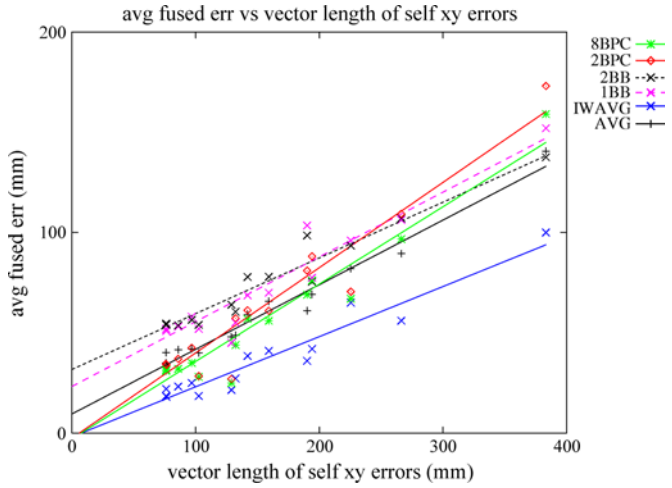

Fig. 16. Fused error versus combination of translational self-localization errors from all robots. Only points where orientation errors are small are considered. The fact that the slope is less than 1 indicates that there is at least some compensation for errors in self-localization. All methods perform similarly.

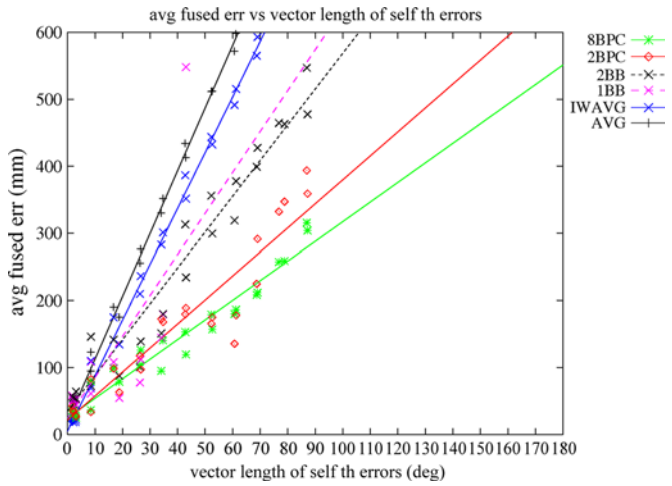

Fig. 17. Fused error versus combination of self-orientation errors from all robots. Only points where translational errors are small are considered. The absolute values are not very informative, but the performance of our fusion methods is visibly better than even the upper bound IWAVG.

of self-localization errors, we gather data for all runs in which *only* landmark observations were corrupted.

In Fig. 16, we plot the error in the fused object-position estimate for each method versus the combined translational (straight-line) errors, also called (x, y) errors, in the self-localization estimates of all robots. The errors are combined using the square root of the sum of squares (SRSS) of the errors from the three robots. In other words, we use the vector length of the errors. Only cases where the combined orientation errors are below 5° are considered since we want to isolate the impact of translational errors in self-localization. In this case, the methods perform similarly, with the IWAVG performing slightly better than other methods.

In Fig. 17, we plot the error in the fused object-position estimate for each method versus the combined orientation errors in the self-localization estimates of all robots. The orientation

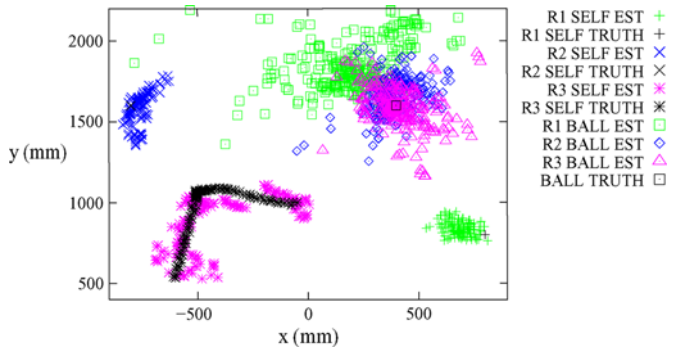

Fig. 18. Plot showing (x, y) estimates of self and ball position, for each robot, during a sample run of layout 3.

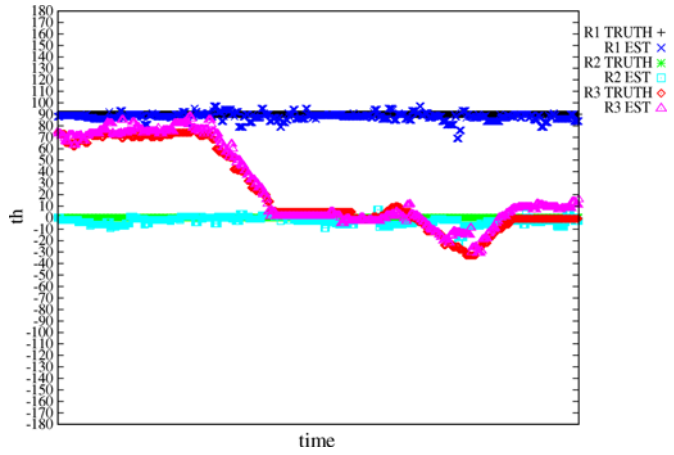

Fig. 19. Plot showing orientation estimates for each robot during the course of a sample run using layout 3.

errors are combined in the same way as the translational errors using the SRSS method. Only cases where the combined translational errors are below 100 mm are considered since, in this case, we want to isolate the impact of orientation errors in self-localization. Our methods all seem to outperform both reference methods, and the degradation of our methods with respect to the amount of data required is quite smooth.

In both figures, each point corresponds to the average error over one run, for one type and amount of introduced error on landmark observations. First-order trend lines computed using regression are added to clarify the data.

H. Live Experiments

We now present the results based on real data, which are obtained using layout 3 from Fig. 10. These results reflect one particular point on the input-error landscape. In each run of the experiment, one of the robots moves forward approximately 300 mm, turns right approximately 90°, and, again, moves forward approximately 300 mm. All robots are observing landmarks and the ball throughout each run. We collected data for 20 runs of layout 3, but one run ended up being unusable; we present the results for the other 19 runs.

Fig. 18 shows the self-localization and target object estimates for each robot during one run of layout 3. Fig. 19 shows the

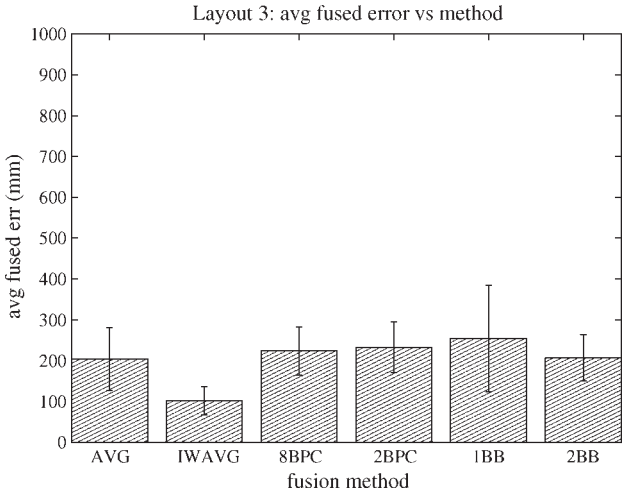

Fig. 20. Results for each method, averaged over 19 runs of layout 3.

orientation estimates for the same run. These plots give a rough indication of the types and amounts of errors contained in the data. Range was typically overestimated probably due to imperfect vision calibration. Bearing and orientation estimates, on the other hand, were fairly accurate. This gives an idea as to where on the input-error landscape the real data case we considered was situated. The average and the standard deviation of the fused estimate error for each method over all runs of layout 3 are shown in Fig. 20. In this case, our methods perform slightly worse than the reference methods. This is consistent with data from the case in the systematic analysis where random range errors were introduced (Fig. 14).

I. Discussion

The experiments presented here have examined the input-error landscape of our method. It is apparent that, in the presence of bearing and orientation errors, our methods tend to outperform even the upper bound reference method. To a lesser extent, our methods are not as effective when faced with range errors. Recall that the upper bound reference method could not be achieved in a real system since it uses ground-truth information to compute the averaging weights.

Another important observation is that the approximations showed reasonably graceful degradation as the bandwidth requirements were lessened. As expected, the approximations yielded less consistent and less accurate results as the amount of shared information was decreased. However, in a number of cases, the parametric 2BB method performed particularly well, particularly considering the small amount of information that needs to be sent when using this approximation. This type of graceful degradation allows one to make an informed decision regarding the tradeoff between bandwidth and accuracy on an application-specific basis.

In the real data experiment, range errors were considerable, whereas bearing errors were less apparent; the results of this experiment are consistent with the results of the systematic analysis of the input-error landscape.

The fact that our method is consistently better at dealing with bearing and orientation errors indicates that our fuzzy

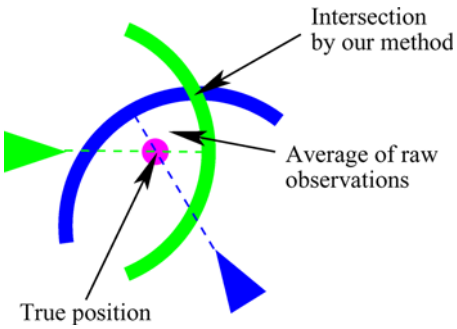

Fig. 21. Example of a situation where range overestimation results in our method performing worse than simple averaging.

fusion approach has certain robustness with respect to these types of input errors, even beyond the tolerance encoded into the sensor models. We believe that there are two main reasons for this. First, our method correctly represents and propagates uncertainty in self-localization, including orientation uncertainty. Therefore, orientation errors are less likely to affect the object-position estimates exchanged between robots. Second, since our method is set-theoretic, it is better able to handle the nonlinearities that result from bearing and orientation errors. Specifically, we compute all possible positions where an object could be and use all of these in the fusion step; this avoids the heavy loss of information that results from approximating estimates as crisp positions before they are fused. This loss of information is of particular significance when dealing with bearing errors since even small bearing errors can result in large position errors.

The reason our method is significantly better at dealing with bearing errors is likely also the reason why it is slightly less effective at dealing with range errors. Tolerance to bearing errors implies that each robot must consider many positions as possible for a given object; these positions are often laid out on an arc. If range errors occur, a large number of these positions are likely to be wrong. In such a situation, the intersection that reflects the agreement between robots may also be wrong. Fig. 21 shows a simple example of a situation where this might occur. In most cases, it is useful to share as much information as possible; however, in some cases, sharing less information could result in more accurate results.

The performance of our method with respect to range errors can be improved by increasing the width of the range sensor model at the expense of reduced precision. Alternatively, we can reduce the width of the bearing sensor model at the expense of a reduction in performance with respect to bearing errors. We have verified that, by changing these parameters, albeit by a fairly large amount, we can adapt our method so that its performance is very close to that of the IWAVG reference method with respect to both range and bearing errors. For the presented experiments, however, we have preferred to use the original tuning, which was intended to reflect (very roughly) the error characteristics of the actual vision sensors used. This tuning is also particularly interesting since it shows how our method offers a significant improvement in performance with respect to bearing errors, while suffering only a slight degradation in performance with respect to range errors.

It is also interesting to note that without the systematic analysis using artificially modified data, these aspects would not have been apparent. Using real data is obviously important for testing any robotic system due to the noise and other errors inherent in such systems. However, as we have shown, it can be useful to use artificial data to more fully characterize the performance of a given method.

VII. Conclusion

In this paper, we have presented a method for addressing the multirobot object localization problem. The problem is seen as an information fusion problem, and our approach relies on techniques from fuzzy logic to represent and combine information arriving from different sources at different times. The resulting method has few parameters to tune, and it is robust with respect to sensor model calibration. Our method also fully considers self-localization uncertainty when computing object-position estimates.

The computational, memory, and bandwidth requirements of our method, particularly considering the approximations that can be used if resources are particularly limited, allow the method to be implemented on fairly limited platforms. We have successfully tested and used implementations of the approximate coordinate transformation and the 2BB global grid approximation on teams of Sony AIBO robots in earlier RoboCup competitions [44]; in this domain, multirobot ball localization was performed at 1 Hz on-board the robots.

The evaluation of our method was based on a rather novel type of experimental analysis. The analysis systematically defined and explored an input-error landscape for our method by examining the effects of applying various types of errors to each of the method's inputs. Also, instead of using poorly implemented versions of alternative approaches as reference methods, we chose to compare our method with upper and lower bounds of averaging-based approaches. We have also presented an experiment with real data, which was consistent with the systematic analysis. This evaluation has proven to be extremely useful, and it can be seen as a methodological contribution in itself. The experiments showed that our method is particularly effective at dealing with bearing errors in object observations and orientation errors in self-localization.

There are a number of ways in which this paper could be improved and extended. For one thing, more scalable representations of fuzzy sets, as well as suitable approximations of these, could be explored since grids are not adequate for representing very large (e.g., outdoor) environments. Also, this paper does not attempt to address the dynamic aspects of the object-localization problem. Prediction and filtering applied in various places could improve the performance of the method by taking domain information into account. The purpose of this paper was to examine the fusion process in isolation; an interesting extension would be to include an examination of the relevant temporal aspects. Extensions to our method for considering temporal aspects do exist; however, these have yet to be fully evaluated. Finally, it would be interesting to see how our methods compare to other alternative approaches, which do not rely on averaging.

References

[1] J.-P. Cánovas, K. LeBlanc, and A. Saffiotti, "Robust multi-robot object localization using fuzzy logic," in *RoboCup 2004: Robot Soccer World Cup VIII*, vol. 3276, D. Nardi, M. Riedmiller, and C. Sammut, Eds. Berlin, Germany: Springer-Verlag, 2004.

[2] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput.*, vol. 1, no. 4, pp. 180–197, Dec. 1997.

[3] I. Bloch, "Information combination operators for data fusion: A comparative review with classification," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 26, no. 1, pp. 52–67, Jan. 1996.

[4] S. Benferha and C. Sossai, "Reasoning with multiple-source information in a possibilistic logic framework," *Inform. Fusion*, vol. 7, no. 1, pp. 80–96, Mar. 2006.

[5] P. Buschka, A. Saffiotti, and Z. Wasik, "Fuzzy landmark-based localization for a legged robot," in *Proc. IEEE Int. Conf. IROS*, Takamatsu, Japan, 2000, pp. 1205–1210.

[6] K. Demirli and M. Molhim, "Fuzzy dynamic localization for mobile robots," *Fuzzy Sets Syst.*, vol. 144, no. 2, pp. 251–283, Jun. 2004.

[7] D. Gohring, "Cooperative object localization using line-based percept communication," in *RoboCup 2007: Robot Soccer World Cup XI*, vol. 5001, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin, Germany: Springer-Verlag, 2008, pp. 53–64.

[8] D. Gohring and H.-D. Burkhard, "Cooperative world modeling in dynamic multi-robot environments," *Fundamenta Informaticae*, vol. 75, no. 1–4, pp. 281–294, 2007.

[9] D. Herrero-Pérez, H. Martínez-Barberá, and A. Saffiotti, "Fuzzy self-localization using natural features in the four-legged league," in *RoboCup 2004: Robot Soccer World Cup VIII*, D. Nardi, M. Riedmiller, and C. Sammut, Eds. Berlin, Germany: Springer-Verlag, 2004.

[10] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artif. Intell. Res.*, vol. 11, pp. 391–427, 1999.

[11] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. San Mateo, CA: Morgan Kaufmann, 2002.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. ASME, J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960, series D.

[13] A. Gelb, *Applied Optimal Estimation*. Cambridge, MA: MIT Press, 1989.

[14] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. New York: Academic, 1973.

[15] P. S. Maybeck, "Stochastic models, estimation, and control," in *Mathematics in Science and Engineering*, vol. 141. New York: Academic, 1979.

[16] S. Julier and J. Uhlmann, "A new extension of the Kalman filter to nonlinear systems," in *Proc. 11th Int. Symp. Aerosp./Defense Sens., Simul. Controls*, 1997. **AQ1**

[17] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *Pervasive Comput.*, vol. 2, no. 3, pp. 24–33, Jul.–Sep. 2003.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[19] A. Stroupe, C. Martin, and T. Balch, "Merging Gaussian distributions for object localization in multi-robot systems," in *Proc. ISER*, 2000, pp. 343–352.

[20] R. Hanek, T. Schmitt, M. Klupsch, and S. Buck, "From multiple images to a consistent view," in *RoboCup 2000: Robot Soccer World Cup IV*, P. Stone, T. Balch, and G. Kraetzschmar, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 169–178.

[21] A. Ferrein, L. Hermanns, and G. Lakemeyer, "Comparing sensor fusion techniques for ball position estimation," in *Proc. RoboCup Symp.*, 2005. **AQ2**

[22] F. Li-Wei, "Distributed data fusion algorithms for tracking a maneuvering target," in *Proc. 10th Int. Conf. Inf. Fusion*, Quebec, QC, Canada, 2007, pp. 1–8.

[23] H. F. Durrant-Whytein "A beginners guide to decentralised data fusion," Australian Centre Field Robot., Univ. Sydney NSW, Sydney, Australia, 2006. Tech. Rep.

[24] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Autom. Control*, vol. AC-24, no. 6, pp. 843–854, Dec. 1979.

[25] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig, "Cooperative probabilistic state estimation for vision-based autonomous mobile robots," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 670–684, Oct. 2002.

[26] P. Pinheiro and P. Lima, "Bayesian sensor fusion for cooperative object localization and world modeling," in *Proc. Conf. Intell. Auton. Syst.*, Amsterdam, The Netherlands, 2004.

[27] M. Fernandez and H. F. Durrant-Whyte, "A failure detection and isolation algorithm for a decentralised multisensor system," in *Proc. IEEE Int. Conf. Multisensor Fusion Integr. Intell. Syst.*, Las Vegas, NV, 1994, pp. 27–33.
[28] H. F. Durrant-Whyte, *Integration, Coordination, and Control of Multi-Sensor Robot Systems*. Norwell, MA: Kluwer, 1988.
[29] Z. Weihong, "A probabilistic approach to tracking moving targets with distributed sensors," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 5, pp. 721–731, Sep. 2007.
[30] M. Dietl, J.-S. Gutmann, and B. Nebel, "Cooperative sensing in dynamic environments," in *Proc. IEEE Int. Conf. IROS*, 2001, pp. 1706–1713.
[31] J.-S. Gutmann, "Markov–Kalman localization for mobile robots," in *Proc. ICPR*, 2002, pp. 601–604.
[32] D. Danu, T. Kirubarajan, T. Lang, and M. McDonald, "Multisensor particle filter cloud fusion for multitarget tracking," in *Proc. Int. Conf. Inf. Fusion*, Cologne, Germany, 2008, pp. 1–8.
[33] Y. Ting, W. Ying, N. Krahnstoever, and P. Tu, "Distributed data association and filtering for multiple target tracking," in *Proc. IEEE Int. Conf. CVPR*, Anchorage, AK, 2008, pp. 1–8.
[34] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 19, no. 1, pp. 5–18, Jan. 2004.
[35] W. Nistico, M. Hebbel, T. Kerkhof, and C. Zarges, "Cooperative visual tracking in a team of autonomous mobile robots," in *RoboCup 2006: Robot Soccer World Cup X*, vol. 4434, G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 146–157.
[36] L. Marchetti, D. Nobili, and L. Iocchi, "Improving tracking by integrating reliability of multiple sources," in *Proc. 11th Int. Conf. Inf. Fusion*, Cologne, Germany, Jul. 2008, pp. 1–8.
[37] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.
[38] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 1, no. 1, pp. 3–28, 1978.
[39] D. Dubois, J. Lang, and H. Prade, "Possibilistic logic," in *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 3, D. Gabbay, C. J. Hogger, and J. A. Robinson, Eds. Oxford, U.K.: Clarendon, 1994, pp. 439–513.
[40] S. Weber, "A general concept of fuzzy connectives, negations and implications based on t-norms and t-conorms," *Fuzzy Sets Syst.*, vol. 11, no. 2, pp. 115–134, 1983.
[41] I. Bloch and H. Maître, "Fuzzy mathematical morphologies: A comparative study," *Pattern Recognit.*, vol. 28, no. 9, pp. 1341–1387, 1995.
[42] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.
[43] Sony Corporation, *AIBO Robot Web Site*. [Online]. Available: http://www.aibo.com
[44] RoboCup Federation, *RoboCup Web Site*. [Online]. Available: http://www.robocup.org
[45] A. Saffiotti and K. LeBlanc, "Active perceptual anchoring of robot behavior in a dynamic environment," in *Proc. IEEE ICRA*, 2000, pp. 3796–3802.
[46] Polhemus Tracking Systems, *Polhemus Web Site*. [Online]. Available: http://www.polhemus.com
[47] A. Saffiotti, A. Björklund, S. Johansson, and Z. Wasik, "Team Sweden," in *RoboCup 2001: Robot Soccer World Cup V*, vol. 2377. Berlin, Germany: Springer-Verlag, 2002.
[48] Z. Wasik and A. Saffiotti, "Robust color segmentation for the RoboCup domain," in *Proc. ICPR*, Quebec, QC, Canada, 2002, pp. 651–654.

Kevin LeBlanc received the B.Sc. degree in electrical engineering from the University of New Brunswick, Fredericton, NB, Canada. He is currently working toward the Ph.D. degree in computer science at Örebro University, Örebro, Sweden.

He is currently working as a Consultant in Lund, Sweden. His research interests include autonomous robotics, cooperative robotics, and anchoring.

Alessandro Saffiotti (M'00–SM'04) received the M.S. degree in computer science from the University of Pisa, Pisa, Italy, and the Ph.D. degree in applied science from the Université Libre de Bruxelles, Brussels, Belgium.

He is currently a Full Professor of computer science with the Center for Applied Autonomous Sensor Systems, Department of Technology, Örebro University, Örebro, Sweden, where he has been heading the AASS Mobile Robotics Laboratory since 1998. He has published more than 120 papers in international journals and conferences, and coedited the book *Fuzzy Logic Techniques for Autonomous Vehicle Navigation* (Springer, 2001). His research interests encompass artificial intelligence, autonomous robotics, and soft computing.