Perception-Based Self-Localization
Using Fuzzy Locations

Alessandro Saffiotti¹ and Leonard P. Wesley²

¹ IRIDIA, Université Libre de Bruxelles
50 av. Roosevelt, CP 194/6, Brussels, B 1050 Belgium
asaffio@ulb.ac.be
WWW: http://iridia.ulb.ac.be/saffiotti

² Computer Engineering Dept., San Jose State University
1 Washington Sq., San Jose, CA 95192 USA
E-mail: wesley@ai.sri.com

Abstract. We describe a fuzzy-based approach to self localization to support
indoor robot navigation. Our approach is perception-based: clues extracted by
the perceptual apparatus are matched against an approximate map to obtain an
estimate of the robot’s location in the map. Each perceptual clue is treated
as a source of partial locational information, represented by a fuzzy set; other
sources, like odometry or external measurements, are also treated in this
way. Information coming from different sources is combined using a fuzzy
aggregation operator. We illustrate our approach by showing experiments
performed on a mobile robot, Flasby.

1 Introduction

Purposeful autonomous navigation requires having reliable locative information
in order to assess the desirability of pursuing particular actions to sense,
perceive, or alter one’s position in the environment. Previous work on autonomous
navigation has typically included non-perceptual locative techniques, such as
dead-reckoning, to maintain a large global Cartesian-map representation. Unfortu-
nately, dead-reckoning approaches suffer from cumulative errors introduced
by encoder imprecision and by wheels and gears slippage. Other approaches
have focused on developing robust perceptual-based inference techniques, and
have used perceptual-based schemes that emphasize the integration of distinct
sensor modality data [1, 6, 9, 8]. Perception-based approaches are faced with a
fundamental tradeoff between using the perceptual clues that occur naturally in
the environment, or instrumenting the environment with artificial clues. While
physical alteration of the environment is not always feasible or desirable, the
use of natural clues brings about the problem of how to deal with the inherent
uncertainty and imprecision of perceptual recognition.

Perceptual-based localization techniques build on the idea to match percep-
tual clues against a map of the environment in order to establish the observer’s
position. This approach requires capabilities to 1) recognize features of the
environment by using the perceptual apparatus; 2) for each recognized feature, find
a matching object in the map; and 3) use this match to compute the position
in the map where the robot should be in order to observe the object where it has been actually seen. This approach must take into account several sources of uncertainty. Perceptual recognition is inherently uncertain and often gives only an approximate estimate of the object's position. Exact match between the perceived feature and an object in the map is unlikely; moreover, different objects in the map can match, perhaps at different degrees, the same perceptual feature. Finally, the information contained in the map can be incomplete, approximate, or uncertain — especially if it has been built by the robot itself.

To better visualize the uncertainty involved in recognizing and using perceptual clues, we show in Fig. 1 some real sonar data collected by Flakey, the mobile robot that we used in our experiments,\(^3\) while passing by an open door. The figure represents the immediate neighborhood of the robot as it is perceived by its sonar sensors; Flakey itself is represented by the octagonal shape, in top-view and pointing rightward. The small squares indicate the last 30 sonar readings, cumulated as Flakey was running down the corridor. The two corridor walls are clearly visible. Because of various phenomena of beam reflection, the configuration of the sonar readings around the door (the "door signature") is confused and largely unpredictable. As it turns out, it is extremely difficult to write a reliable routine that recognizes a door from its sonar signature. Moreover, the position of the door cannot be established with certainty, particularly for what concerns its distance and orientation with respect to the robot.

![Fig. 1. The sonar signature of a door in robot's coordinates.](image)

Many of the current perceptual approaches to self-localization are based on some form of Kalman's filter, and model uncertainty in locations by probability

\(^3\) Flakey is a custom research robot built at SRI International. Flakey is provided with a ring of 12 sonar sensors on the bottom, wheel encoders, and other sensors not used in our experiments. See [2, 12, 13] for other aspects of our work on Flakey.
distributions (see, for instance, [15], or Crowley’s tutorial in this volume). To effectively use the filter, these approaches typically assume that errors are normally distributed, and that they are small enough that linear approximations can be correctly used — in particular, that a good initial estimate of the robot’s position is available. Practical experience suggests that these assumptions are often violated in reality. In this paper, we propose to represent locational imprecision fuzzy sets [16]. Fuzzy techniques have proved effective in controlling systems that are significantly non-linear, which may operate under conditions of great variability, or for which we only have a qualitative description. One may conjecture that fuzzy-set based techniques enjoy similar advantages when used to address the self-localization problem. The preliminary results shown here suggest that this is indeed the case.

The outline of our fuzzy localization algorithm is very simple. At each time-step the robot has an approximate hypothesis of its own location in the map, represented by a fuzzy set $h_t$. During navigation, the robot’s perceptual apparatus recognizes relevant features, and associates them with approximate locations, also represented by fuzzy sets. For each such feature $p$, the map is searched for matching objects using a fuzzy measure of similarity. Each matching object is then used to build a fuzzy localizer $\text{loc}_p[\theta]$; a fuzzy set representing the approximate location in the map where the robot should be in order to see the object where $p$ has been perceived. Each localizer provides one imprecise source of information about the actual position of the robot; information coming from other sources, like an odometer of a GPS system, is also wrapped into a fuzzy localizer. Finally, all the localizers are combined by fuzzy intersection to produce an updated hypothesis $h_{t+1}$ about the robot’s location.

The rest of this paper describes, in the order: the representation of approximate locations; the matching between perceptual features and objects in the map; the way to build fuzzy localizers; the overall localization algorithm; and some experiments based on a preliminary implementation of our technique.

2 Approximate Locations

We represent the approximate location of an object by a fuzzy subset of a given space, read under a possibilistic interpretation [17]: if $P_o$ is a fuzzy set representing the approximate location of object $o$, then we read the value of $P_o(x) \in [0, 1]$ as the degree of possibility that $o$ be actually located at $x$. (See [16, 7] on fuzzy sets, and [11] for some foundation issues.) This representation allows us to model different aspects of locational uncertainty. Figure 2 shows six approximate locations in one dimension: (a) is a crisp (certain) location; in (b), we know that the object is located at approximately 5 (this is commonly referred to as “vagueness”); in (c), it can possibly be located anywhere between 5 and 10 (“imprecision”); in (d), it can be either at 5 or at 10 (“ambiguity”); (e) shows a case of “unreliability”: we are told that the object is at 5, but the source may be wrong, and there is a small “bias” of possibility that it be located just anywhere (this is what Crowley calls “confidence” in his tutorial). As an extreme case, we repre-
sent total ignorance by the "vacuous" location \( P(x) = 1 \) for all \( x \): any location is perfectly possible. Finally, (f) combines vagueness, ambiguity and unreliability. Clearly, the information provided by a measurement device can present any of the above aspects, alone or in combination. It is important to emphasize that a degree of possibility is not a probability value: e.g., there is no necessary relation between the observed frequency of a location and its possibility; and degrees of possibility of disjoint locations need not add up to one.

![Fig. 2. Representing different types of uncertainty by fuzzy sets: (a) crisp; (b) vague; (c) imprecise; (d) ambiguous; (e) unreliable; (f) combined.](image)

In order to represent maps, we consider a **global location space** \( G \), whose elements are triples \( (x, y, \theta) \) of coordinates in a global Cartesian frame; where \( \theta \) is the orientation with respect to \( X \). Figure 3 shows some elements of an office environment in the \( G \) space; the circle shows the position where Flaky was when the snapshot in Fig. 1 above was taken. We call **approximate global position**, or \( \text{AGP} \), any fuzzy subset of \( G \). Let \( \tilde{g} : G \rightarrow [0, 1] \) be the \( \text{AGP} \) of some object; then, for any \( g \in G \), the value of \( \tilde{g}(g) \) measures, on a [0, 1] scale, the possibility that \( g \) be the actual location of that object. In particular, we represent the robot’s hypothesis about its own position in \( G \) at time \( t \) by a \( \text{AGP} \) \( \tilde{h}_t \).

An **approximate map** on \( G \) is a tuple

\[
\langle M, \text{Type}, \text{Pos}, P_1, \ldots, P_n \rangle
\]

where \( M \) is a non-empty set of map indexes (the "names" of the objects); \( \text{Type} \) associates each index to an element in some given set of object types (e.g., \{door, wall, corridor\}); \( \text{Pos} \) associates each object to a \( \text{AGP} \) in \( G \) (its location in the

\(^4\) Although we use here a Cartesian representation of space, our approach should readily apply to other representations as well, like coarse metric maps, partitioned maps, or topological maps.
map); and \( P_1, \ldots, P_n \) are included to account for other properties that are not relevant in this context (e.g., topological properties used by a planner). In this paper, we assume that the map is pre-existing and fixed. Figure 4 shows the AGP's of an approximate map for the above environment. For simplicity, all the AGP's there are triangular, that is, their projection on each axis has the shape shown in Fig. 2 (b). Triangular AGP's are graphically represented by ellipsoids centered at the vertex of the triangle: the \( x \) and \( y \) radiiuses are proportional to the width of the \( x \) and \( y \) components, while the width of the \( \theta \) component is represented by an arc oriented as \( \theta \). For instance, door D1 has possibility 1 of being located at \((2, 2, 90^\circ)\); possibility 0 of being at \((2, 2, 60^\circ)\) or at \((1, 2, 90^\circ)\); and intermediate values in between.

3 Global and Local Location Spaces

The \( G \) space provides a convenient frame for map representation, as it does not depend on the current position of the observer. Most of the data acquired by
the robot by sensing, however, are relative to the robot’s body and within a limited range from it. So, we consider a second space \( L \) of local locations, which is centered on the robot. For example, in Flakey \( L \) is a 4 meter wide Cartesian plane with origin on Flakey (see Fig. 1). The elements of \( L \) are triples of \((x, y, \theta)\) coordinates, where \( \theta \) is the orientation with respect to the robot’s heading.

The \( G \) and \( L \) spaces are linked by the notion of current locational hypothesis: at every moment \( t \) the robot has a hypothesis \( h_t \) about its own location in the \( G \) space (we shall often omit the \( t \) index); that is, \( h \) is the position of \( L \) in \( G \). If \( h \) is a single point, then we can translate locations between \( G \) and \( L \) by plain coordinate transformation: for any \( h \in G \), we assume to have a function

\[
T_{[h]} : L \rightarrow G
\]

that maps each location in \( L \) to a corresponding location in \( G \). When \( G \) and \( L \) are Cartesian frames, \( T_{[h]} \) is simply given by

\[
T_{[h]} \begin{bmatrix} x_G \\ y_G \\ \theta_G \end{bmatrix} = \begin{bmatrix} \cos(h) & -\sin(h) & 0 \\ \sin(h) & \cos(h) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_L \\ y_L \\ \theta_L \end{bmatrix} + \begin{bmatrix} h_x \\ h_y \\ h_\theta \end{bmatrix}.
\]

If the robot knew precisely its location in \( G \), we could in principle get rid of the distinction between the \( G \) and \( L \) spaces — e.g., we could translate all sensor data in global coordinates, and work only in the \( G \) frame. When the robot’s location hypothesis \( h \) is approximate, however, this translation is more complex, and it is reasonable to maintain two separate representations for \( L \) and \( G \). We translate the AGR of a map object \( m \) into its expected local position in the \( L \) space by the following function:

\[
\text{Exp}_{L} (m, l) = \sup_{g \in G} \left( \tilde{h}(g) \land \text{Pos}(m, T_{[g]}(l)) \right),
\]

where \( \land \) denotes the minimum.\(^5\) Intuitively, we expect to find object \( m \) at local location \( l \) if, and to the extent by which, there exists a \( g \in G \) such that: (a) \( g \) is a possible location of the robot (as measured by the current approximate location hypothesis \( \tilde{h} \)), and (b) the global location corresponding to \( l \) given \( g \) is a possible location for \( m \) (as measured by \( \text{Pos} \)).

### 4 Observed Features

We assume that the robot has perceptual routines to recognize relevant features in the environment, classified by the same types used to classify map objects. We expect perceptual routines to build a representation of each perceived feature in the local space \( L \), and to attach it a type, local position, and possibly other properties. As perception is error-prone, type and position are both represented

---

\(^5\) The \( \max \) and \( \min \) operators are commonly used for union and intersection of fuzzy sets. They are also used to model conjunction and disjunction in fuzzy logics; correspondingly, \( \sup \) and \( \inf \) are used for existential and universal quantification.
by fuzzy sets. We denote by \( \text{Feat}(p, t) \in [0, 1] \) the degree of possibility of feature \( p \) being of type \( t \); and by \( \text{Perc}(p) \) the ALP of \( p \) — hence, \( \text{Perc}(p, l) \in [0, 1] \) measures the possibility that \( p \) be located at \( l \in L \).

For a given perceived feature \( p \), the shape of \( \text{Perc}(p) \) depends on the quality of the recognition and on the type of \( p \). For example, consider the way Flakey recognizes a wall (Fig. 5 left). Flakey’s perceptual routines interpolate sequences of co-linear readings to extract linear features. A linear feature \( w \) that is long enough is assumed to be a wall, and marked by a “W”. As having observed a segment of the wall only tells us something about its distance and orientation with respect to the robot, \( \text{Perc}(w) \) is completely vague in its longitudinal component — the vagueness in the other components depends on the length of \( w \), and on the density and dispersion of the sonar readings. Also, both directions for the wall are equally possible, as we cannot discriminate the direction from the observed segment. By contrast, consider the ALP built by Flakey’s door detector (Fig. 5 right). The sonar signature of a door on a side can only give a reasonable indication about the position of the door along the longitudinal axis, leaving its distance and orientation extremely vague; correspondingly, the door’s ALP has the shape shown in the picture (again, there is ambiguity in the direction). Note that \( \text{Perc}(p) \) can also include a “bias” to reflect the unreliability of perceptual recognition (cf. Fig. 2 (c-f)).

![Fig. 5. The ALP built by perceiving a wall (left), and a door (right).](image)

In addition to simple features like doors and walls, we recognize complex objects by applying hierarchical rules that describe composite objects in terms of the component ones. For example, the following fuzzy rule is used to infer an instance of a corridor from two facing walls.

\[
\text{IF } \text{Feat}(w_1, \text{wall}) \land \text{Feat}(w_2, \text{wall}) \land \\
\text{angle}(w_1, w_2) \approx 0 \land \text{dist}(w_1, w_2) \approx 2m \\
\text{THEN } \text{Feat}(c, \text{corridor}).
\]
Objects in the map are also represented hierarchically through “part-of” links. Hierarchical representation of objects facilitates rapid type matching, as well as the manual input of map information. Hierarchical matching also helps to reduce perceptual aliasing in situations of extreme uncertainty: more abstract elements are better suited for subsequent matching because they are fewer in number and are better discriminated by their position, orientation, and constituent components. (See [14] for more on hierarchical matching.)

Once we have recognized an object of a given type, we want to find one (or more) matching object(s) in the map. To do this, we use a measure of similarity between a perceived feature $p$ and a map object $m$ based on the amount of overlap between their respective approximate positions in $L$:

$$\text{Sim}_{\mathcal{B}}(p, m) = \sup_{l \in L} \left( \text{Exp}_{\mathcal{B}}(m, l) \land \text{Perc}(p, l) \right) \land \text{Feat}(p, \text{Type}(m)).$$  \hspace{1cm} (2)

Intuitively, $m$ and $p$ are similar to the extent by which they are of the same type, and there is some location in $L$ that is both possible for $p$ and expected for $m$ given the current location hypothesis $h$. Note that the more $h$ is vague, the more objects are considered similar.

5 Localizers

The heart of our localization procedure consists in matching perceptual clues with map objects to build partial hypotheses, or localizers, about the robot’s current position. Each pair of a perceived feature $p$ and a map object $m$ of the same type can be used to build a localizer $\text{loc}_{p, m}$: the meaning of $\text{loc}_{p, m}$ is to describe the fuzzy location in $G$ where the robot should be in order to perceive $m$ in the position where $p$ has been observed. Said differently, the value of $\text{loc}_{p, m}(h)$ measures the possibility that the robot is located at $h$, assuming that $m$ and $p$ identify the same object. Correspondingly, we define $\text{loc}_{p, m}(h)$ as the locational similarity between $m$ and $p$ when the robot is at $h$:

$$\text{loc}_{p, m}(h) = \sup_{l \in L} \left( \text{Pos}(m, T[h](l)) \land \text{Perc}(p, l) \right).$$  \hspace{1cm} (3)

That is, the robot can possibly be at $h$ to the extent that there is a local $l$ that can be both the perceived position of $p$ and the map position of $m$ given $h$.

It is easy to see that the shape of $\text{loc}_{p, m}$ so built depends, among others, on the shape of Perc($p$). This means that the type of information given by $\text{loc}_{p, m}$ depends on the type of $p$, and on the sensing modality used. Figure 6 shows the localizer $\text{loc}_{\text{door}, \text{door}}$ built by matching door $d$ in Fig. 5 with door D2 in the map (the levels of grey indicate degrees of possibility). As we expect, matching a door in a corridor can help the robot to estimate its position along the $X$ axis, but provides little information along the $Y$ and $\Theta$ axes; matching a corridor wall would provide a roughly complementary type information. Indeed, we can read $\text{loc}_{p, m}$ as an elastic constraint on the possible position of the robot, which may constrain some dimensions while remaining vague on others.
The \text{loc}_{p,m} \text{ locator assumes that } p \text{ and } m \text{ match, i.e., they identify the same object. In general, an observed feature } p \text{ can match, at different degrees, several map objects } m_1, \ldots, m_n. In order to build the locator that represents the information given by having perceived } p \text{, then, we merge all the } \text{loc}_{p,m_i} \text{ localizers, } i = 1, \ldots, n \text{, into one locator } \text{loc}_p, \text{ weighting them by the corresponding degrees of similarity:}

\[
\text{loc}_p(h) = \sup_{m \in M} \left( \text{loc}_{m,p}(h) \land \text{Sim}_{p}(m, p) \right) \circ \mu_p,
\]

\[
\mu_p = \sup_{m \in M} \text{Sim}_{p}(m, p),
\]

where \( \circ \) denotes the Lukasiewicz implication, defined by: \( x \circ y = (x - y + 1) \land 1 \). \( \mu_p \) is a normalization factor that makes sure that there is some location that is completely possible. The effect of normalization is to add a “bias” of value \( 1 - \mu_p \) to \( \text{loc}_p \) (cf. Fig. 2 (e-f)), indicating that the locator can be unreliable to the extent that there are no good matches for \( p \) in the map. In particular, if all \( m \)'s have zero similarity to \( p \) (e.g., if \( p \) is an outlier), then \( \text{loc}_p(h) \) is the vacuous location.

If both \( \hat{h} \) and the \( \text{AGP}'s \) in the map are sharp enough, then \( \text{Sim}_{p}(m, p) \) will be non null for just one map element \( m \), and \( \text{loc}_p(h) \) will coincide with \( \text{loc}_{p,m}(h) \). Otherwise, \( \text{loc}_p(h) \) will be affected by some ambiguity. For example, Fig. 7 shows the \( \text{loc}_d \) locator built for door \( d \) of Fig. 5, assuming that \( \hat{h} \) is a sharp \( \text{AGP} \) as in Fig. 3. Two doors in the map have non-zero similarity to \( d \): \( D2 \) and \( D3 \). Then, \( \text{loc}_d \) is the union of \( \text{loc}_{d,D2} \), taken with weight 1, and \( \text{loc}_{d,D3} \), with weight 0.7, as shown in the figure — the \( X \) component has become more vague in both localizers due to the vagueness of \( D2 \) and \( D3 \) in the map.

The ambiguity in finding an element matching \( p \) is explicitly represented in the \( \text{loc}_p \) locator; this ambiguity can then be resolved by using the information contained in other localizers, possibly at a later moment. The explicit representation of ambiguity allows us to circumvent the criticality of the matching, or “correspondence”, problem. In fact, a well know problem in using a measure of similarity to chose one matching element is that similarity depends on the accuracy of the current location hypothesis \( \hat{h} \), and we can incur in serious matching
errors when $\hat{h}$ is highly uncertain. Thus, we only use similarity to cut impossible alternative off, but delay the choice between possible ones until new information is available. As we will see, this gives us the ability to cope with situations of extreme uncertainty. (See [10] for a related solution in a Kalman framework.)

6 The Localization Algorithm

We are now ready to describe the main step of our fuzzy localization algorithm. We assume that, at each time-step $t$, the robot has an approximate location hypothesis about its own location in the global space $G$, represented by a AGP $\hat{h}_t$; and that it has observed and classified a set of perceptual cues $P[t] = \{p_1, \ldots, p_k\}$. Each clue $p_i$ is used as a source of information about the possible current location of the robot, represented by the fuzzy localizer $\text{loc}_{p_i}[t]$. Intuitively, all we have to do is to combine all these $k$ localizers by some form of fuzzy combination to produce the new location hypothesis $\hat{h}_{t+1}$.

We still need one last ingredient, though. Usually, the robot has one more source of locational information available: the measure of the distance traveled since the previous time-step provided by the wheel encoders, commonly referred to as "dead-reckoning". To use this information, we build a dead reckoning localizer $\text{loc}_{dr}[t]$ by taking the current location hypothesis $\hat{h}_t$ and adding the measured displacement through a fuzzy coordinate transformation — in our current implementation, we simply translate and rotate $\hat{h}_t$ according to the encoder data, and "unfocus" it to account for the various errors.

The fusion step of our algorithm is summarized by the following formula:

$$\hat{h}_{t+1}(g) = \bigwedge_{p \in P[t]} \text{loc}_{p}[t](g) \land \text{loc}_{dr}[t](g).$$

(5)

The full localization algorithm consists in the successive evaluation of (1), (4) and (5). Note that the algorithm is recursive, and that it can be seen as an instance of the predict/match/update cycle described in Crowley’s tutorial.
The effect of the \( \text{loc}_d \) localizer in (5) is to propagate, with decreasing precision, the result of previous localization steps into the current step; this allows us to combine the information obtained from clues observed at different times. Note that localizers that are a superset (in the fuzzy-set sense) of the current \( \hat{h} \) do not affect the result of the combination. In particular, a vacuous localizer (e.g., one produced by a feature that does not have a match in the map) acts as an identity; the algorithm ignores false observations or incompleteness in the map.\(^6\) As expected, the locational uncertainty (the width of \( \hat{h} \)) monotonically increases if no perceptual clue is used, i.e., if the only available localizer is odometry.

Two possible variants of (5) are worth mentioning. First, other sources of locational information can be considered in the algorithm; e.g., information provided by and external observer; data from a GPS system; or environmental constraints. All we have to do is to represent the information in the form of a fuzzy localizer, and include it in the conjunction. Second, we can replace the \( \min (\wedge \cdot\cdot\cdot) \) operator in (5) by other operators for fuzzy intersection, corresponding to different ways to aggregate information (see [4] for a wide analysis of fuzzy combination operators). For example, using the point-wise product would result in a combination where the possibility of locations that are supported by several clues is increased. The use of different combination operators is a subject of our current experiments.

7 Experiments

We have written an experimental implementation of the localization algorithm, and tested it on the SRI mobile robot Flakey. Our current implementation makes several simplifying assumptions with respect to the mathematical framework above. The most important ones are:

- The shape of the fuzzy sets used for \( \text{AGP} \)'s and \( \text{ALP} \)'s is constrained to be a union of triangles, i.e., we only model vagueness and ambiguity;
- The shape of \( \text{Perc}(p) \) only depends on the type of \( p \), i.e., we do not take the quality of the specific recognition into account; and
- The map contains only crisp locations, i.e., \( \text{Pos}(m) \) is a point of \( G \).

The last two assumptions have been made for implementation convenience, and will be relaxed in future experiments. The first assumption is much harder to relax. There are two reasons to make it: representation simplicity, and the computational complexity of evaluating (1), (2) and (3) for arbitrary fuzzy sets. Triangular fuzzy locations greatly simplify computations, at the expense of a decrease in representational power, and of the unnecessary inflation of uncertainty in the approximated computations. With these restrictions, the algorithm runs in 1 to 10 msecs, depending on the number of perceived features, and is evaluated at every control cycle in Flakey (100 msecs).

\(^6\) We could of course use unmatched observations to update the map. How to modify the map is an important issue that lays outside the scope of the present paper.
Figure 8 illustrates the operation of our algorithm while Flakey is navigating in the environment of Fig. 3. Each screen-dump shows: On the left, the local state of Flakey, represented in the local frame $L$ (cf. Fig. 1); this includes sonar readings (dots), observed features (doors, marked by a “D”, and walls, marked by a “W”), and the expected position of map features (double lines for walls, and brackets for doors). On the right, Flakey’s internal map in the global frame $G$; the current self-location estimate $\hat{h}$ and the localizers are drawn in this window.

In (a), Flakey just entered the horizontal corridor, coming from the vertical one. The locational uncertainty in $\hat{h}$, visualized by the shape of $\text{loc}_{dr}$, is moderate along $X$ and $\theta$, but large along the $Y$ axis: the reason is that, while in the vertical corridor, Flakey used the walls to correct its estimate along $X$ and $\theta$. On the left, the perceptual routines have recognized the corridor walls (“W”); note that the expected position of the walls is relatively far from the observed one. The perceived walls are grouped into a corridor by the hierarchical rule shown above, and this corridor is then matched to the map to produce the $\text{loc}_C$ localizer (right): Flakey could possibly be everywhere in the long ellipsoid. In (b), $\text{loc}_C$ and $\text{loc}_{dr}$ have been merged into a new $\hat{h}$, thus reducing the uncertainty along $Y$; the expected and the observed position of the walls now coincide (left window). In (c), a door is perceived (“D”, left window), and the corresponding localizer is built (right window). This is then merged with the dead-reckoning localizer, built according to the encoder readings, to produce the new $\hat{h}$ shown in (d). The expected and observed position of the door now coincide (left).

In the previous example, $\hat{h}$ was always sharp enough that only one element in the map had a non-zero similarity to the perceived feature, and so all the localizers built were non-ambiguous. This is not the case in the next experiment, illustrated in Fig. 9. Here, Flakey was started in a situation of total locational ignorance, represented by a vacuous $\hat{h}$ — i.e., $\hat{h}(g)$ is identically 1. After a while (a), Flakey perceives two parallel walls (left window), and infers a corridor by its hierarchical rules. Given the vacuous $\hat{h}$, this corridor matches all the corridors in the map, in both directions, with similarity 1. Correspondingly, we build the (highly ambiguous) localizer shown on the right: Flakey could be in any one of the corridors, directed as shown. Merging this localizer with the vacuous $\hat{h}$ returns the localizer itself as the new $\hat{h}$. In (b), $\hat{h}$ has been updated by dead-reckoning (right window), and Flakey perceives a door on its right (left window). Given our $\hat{h}$, this door can match any one of the doors in the upper corridor, while its similarity to the lower door is 0 due to the incompatible orientation. Correspondingly, the door localizer consists in the union of the four ellipsoids shown in the right window. The new $\hat{h}$ produced after merging is shown by the four smaller ellipsoids: the set of possible locations for Flakey is now much narrower, but it is still ambiguous. In (c), Flakey perceives a second door. This time, given the current $\hat{h}$ (again obtained by dead-reckoning), only one door in the map has non-zero similarity, producing the non-ambiguous localizer shown on the right. The intersection of this localizer with the (ambiguous) dead-reckoning

---

\footnote{As the dead-reckoning localizer \text{loc}_{dr}[g] is built at every cycle (100 msec), it almost exactly overlaps $\hat{h}[g]$; for visual clarity, only one of them is drawn on the screen.}
Fig. 8. A run of the localisation algorithm on Flakkey.
localizer is the fuzzy location shown by the single small ellipsoid. The new \( \hat{h} \) is now non-ambiguous, and so are the expected locations of the map elements based on this \( \hat{h} \); these are drawn in (d), giving now a visually clear idea of the actual position of Flakey in the environment.

8 Conclusions

We have presented a solution to the problem of position estimation for mobile robots based on the use of fuzzy techniques. The distinctive points of our approach are as follows.

- Position estimation is based on the observation of abstract features of the environment;
- Each observation is treated as a source of partial locational information, modeled as an elastic constraint;
- Locational uncertainty is modeled by fuzzy sets; and
- Information from multiple sources is combined by fuzzy aggregation.

Relying on abstract features, like doors or walls, rather than on low-level sensor data allows us to use a coarse map where only (some of) these features are represented, possibly ignoring their precise geometry. It should also make it easier to extend our technique to different sensing modalities and/or environments; we have already incorporated visual markers in some of our experiments. Finally, as noticed when discussing the hierarchical representation, matching abstract features helps to mitigate the perceptual aliasing problem.

The multi-source view of the self-localization problem has several advantages. First, different types of locational information can easily be integrated in this way, provided that they can be represented by fuzzy localizers; this modularity proved very convenient in practice. Second, different operators can be used to combine the information coming from the different sources; we are currently exploring the use of fuzzy aggregation operators different from the minimum. Finally, this view lends itself to be formalized in a multi-agent epistemic logic, where each source of information is modeled by an agent: in [8] we have proposed such a logic, where a “distributed belief” operator performs fuzzy combination of possibilistic information.

The use of fuzzy techniques is probably the most peculiar aspect of our proposal. Although we feel that there is no definite conceptual reason to prefer, in this domain, fuzzy sets to other representations of uncertainty, there are a few practical reasons that justify our choice. First, fuzzy sets can represent different aspects of uncertainty, including vagueness, ambiguity, imprecision, and unreliability (or “confidence”). In particular, the ability to represent the unreliability of a localizer allows us to disregard false observations; and the ability to represent ambiguity allows us postpone the resolution of uncertainty in the matching until further information, coming from other observations, is considered. These features suggest that our fuzzy approach may compare favorably to typical Kalman filtering approaches in situations of extreme uncertainty. The
Fig. 9. An example of self-localization starting from total ignorance.
last example shown above supports this conjecture. Second, the use of fuzzy locations can be smoothly integrated with fuzzy-logic based controllers. This is an important issue, as locational knowledge has eventually to be used to take decisions; to this respect, we are in the process of integrating the outcome of our localization algorithm into Flackey’s fuzzy controller. Finally, fuzzy techniques lend themselves to efficient implementation. Quick computation is critical for robotic applications; in order to guarantee high reactivity, the localization algorithm should be evaluated as often as possible. Our simple implementation can be evaluated at every control cycle, providing continuous re-localization during movement.

Our technique proved effective for keeping our robot well registered during navigation in an unstructured office environment by using naturally occurring perceptual clues. The technique was also able to produce a correct location hypothesis starting from a situation of total ignorance. Despite these encouraging results, our approach is still preliminary, and several points remain for further work. In additions to the extensions already mentioned, one important point raised at the Workshop concerns the re-usability of existing probabilistic knowledge. Although the problem of converting probabilistic information into a corresponding possibilistic one does not have a general solution (but see [5]), we hope that some heuristics can be developed for the specific domain of sensor modeling. On the experimental side, the most urgent task is clearly to relax the simplifying assumptions made in our current implementation. In particular, the restriction to multi-triangular fuzzy sets constitutes a major departure from the mathematical framework presented here. We are currently searching a representation that is both less restrictive and can still guarantee efficient computation. We are also rewriting our implementation to include a better model of dead reckoning, and to take into account the quality of perception and the uncertainty in the map.

Acknowledgments Work by the first author was supported by the Belon project, founded by the Communauté Française de Belgique. The experiments on Flackey were conducted while both authors were at the AI Center of SRI International. Collaboration with Enrique Ruspini and Kurt Konolige during that period has been essential to the development of the ideas reported here. Interactions at the Workshop greatly helped to put this work in perspective.

References


This article was processed using the \LaTeX{} macro package with LLNCS style