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

Knowledge-Based Multi-Criteria Optimization to Support Indoor Positioning

Alessandra Mileo · Torsten Schaub · Davide Merico · Roberto Bisiani

Received: date / Accepted: date

Abstract Indoor position estimation constitutes a central task in home-based assisted living environments. Such environments often rely on a heterogeneous collection of low-cost sensors whose diversity and lack of precision has to be compensated by advanced techniques for localization and tracking. Although there are well established quantitative methods in robotics and neighboring fields for addressing these problems, they lack advanced knowledge representation and reasoning capacities. Such capabilities are not only useful in dealing with heterogeneous and incomplete information but moreover they allow for a better inclusion of semantic information and more general homecare and patient-related knowledge. We address this problem and investigate how state-of-the-art localization and tracking methods can be combined with answer set programming, as a popular knowledge representation and reasoning formalism. We re-

A. Mileo
Digital Enterprise Research Institute
National University of Ireland, Galway
Galway
Ireland
E-mail: alessandra.mileo@deri.org

T. Schaub
Institut für Informatik, Universität Potsdam
August-Bebel-Str. 89
D–14439 Potsdam
E-mail: torsten@cs.uni-potsdam.de

D. Merico
NOMADIS Research Lab, Dept. of Informatics, Systems and Communication
University of Milan-Bicocca
viale Sarca 336/14
I–20126 Milan
E-mail: davide.merico@disco.unimib.it

R. Bisiani
NOMADIS Research Lab, Dept. of Informatics, Systems and Communication
University of Milan-Bicocca
viale Sarca 336/14
I–20126 Milan
E-mail: roberto.bisiani@disco.unimib.it

port upon a case-study and provide a first experimental evaluation of knowledge-based position estimation both in a simulated as well as in a real setting.

Keywords Knowledge Representation · Answer Set Programming · Wireless Sensor Networks · Localization · Tracking

1 Introduction and Motivation

The continuous progress of computing and communication technology is leading towards the realization of living environments pervaded by a high number of invisible devices affecting and improving all aspects of our lives. A crucial issue in these scenarios is to know the physical location of users, since it represents the basis for context-aware and user-centered systems. The problem of indoor position estimation includes two aspects: localization and tracking.

Localization (also referred to as position estimation) is commonly addressed by acquiring the distance from three or more points of known positions and then combining them applying specific methods such as (multi)teration in order to deduce the location of unknown object. The distance can be computed by measuring variations of signal intensity or by timing the latency from transmission to reception.

The accuracy of positioning systems can significantly vary depending on the used technologies and user requirements. It is common for systems based on Wi-Fi (or similar) technologies to give results within an accuracy of a few meters in 2D. On the other hand, more accurate systems exist providing up to a few centimeters in 3D (e.g. exploiting ultra wide frequency bands). Given the low cost of infrastructure and their flexibility, localization systems based on Wireless Sensor Networks (WSN) are a good starting point for tackling the localization problem in home environments. However, when data is noisy and incomplete due to delays, breakdowns, errors in transmission, and alike, their lack of precision in position estimation needs to be supported and compensated.

Tracking is concerned with how the position of moving objects changes in time; it takes into account knowledge about previous positions and a model of movement to estimate the expected (more plausible) position.

Localization and Tracking represent fundamental issues for several WSN applications and therefore they are much worked-on problems. References [20, 12, 22] give a good introduction to these issues.

Different estimation methods can be used to tackle the problem of node localization, such as Maximum Likelihood, Maximum A Posteriori, Least Squares, Moving Average filter, Kalman filter, and Particle filter [8, 19, 18]. Although many of these algorithms give good results in terms of accuracy and precision, only Particle Filters (PFs) can be easily extended to deal with the problem of target tracking [10]. Therefore, PFs are the most popular and commonly used methods for tackling the problem of tracking. For this reason, we consider in our preliminary experimental evaluation results of PFs as representative of a wider class of similar but less effective techniques.

Particle Filters discretize the probability distribution function of the position of the mobile object. Unlike Kalman Filters, they do not approximate the probabilities using only Gaussian distributions, thus we have to impose less conditions to be able to apply the algorithm. The possible positions of the mobile object are represented as a set of particles, each of which has an associated weight. The distribution of these particles in

space gives the most likely positions. Probabilistic approaches to tracking such as PF strongly depend on the specific sensors used and they are mainly based on quantitative cost functions for computing weights. One of the main limitations of these methods is the difficulty to identify within the same model the right global cost function combining heterogeneous sensor data with other criteria like movement, speed, and alike. Another aspect is that the introduction of new sensor information is not straightforward and requires a lot of customization to properly extend the model.

In order to address these limitations, we propose an alternative and innovative approach to support indoor localization and tracking. Our solution is based on Answer Set Programming (ASP), a declarative logic programming framework, combining a compact, flexible, and expressive modeling language with high computational performance. The knowledge representation and reasoning capacities of ASP provide us with the following advantages:

- a knowledge-based level of multi-sensor fusion for position estimation,
- definition of qualitative criteria to combine sensor data and domain-related knowledge for tracking (such as a motion model of the target), and
- easy customization and extensibility of existing solutions in a more flexible way whenever new technology or new sensor-data become available.

The main advantage of using ASP is to deal with incomplete information, stemming from faulty or diverging sensors, and thus to improve the scalability of the whole system. ASP does not only deal with missing sensor data but it also allows us to exploit semantic information for distinguishing the most plausible among varied sensor data.

Our approach is based on a low-cost and energy-aware localization infrastructure. The data collected by each specific type of sensor is processed by traditional algorithms for feature extraction and aggregation. On top of the aggregation phase, a further level of knowledge-based sensor fusion is in charge of combining heterogeneous sensor data to associate it to specific properties or semantic information referred to as *metrics* in Section 3: *best proximity*, *best support*, *best move* and *best coherence*.

In an very precise setting, the best (highest) proximity would be enough to identify the location with reasonable precision. This is the case for Ultra Wide Band-based systems [15, 16]. In a low-cost and low-precision system as ours, we cannot rely only on RSSI, but we need to use more data sources (i.e. more sensors) and additional properties of these data in order to compensate for the noisiness of proximity sensors and validate their coherence with the context when estimating the position.

The *best support* metric has to do with the fact that, ideally, the more sensor data we have from a location, the highest the probability of the person being in that location. This is not always true, but our multicriteria approach makes it possible to reason about when and how we can rely on a property (or a combination of properties) to validate a location.

The *best move* metric has been introduced to take into account the model the movement of the object to be tracked, thus validating sensor data that best complies with the motion model of the object. The inclusion of a motion model is traditionally used in Particle Filter algorithms to better estimate the position of moving objects [14], but it is encoded within the probabilistic model, while we suggest a more flexible approach to take it into account. In our experiment, we are tracking humans, which move across close locations. For this reason, we use the shortest distance between two nearby locations to identify the *best move*. This simple motion model can be enriched by

considering speed and direction, enhancing even more the flexibility of the multicriteria approach.

As for the *best coherence* metric, it measures how coherent a set of sensor data is for a location, according to how many sensor data we expect to be associated with that location. This percentage turns out to better characterize plausible locations if compared to the simpler notion of "best support", because it takes into account the impact of false positives.

The ASP based reasoning process follows the common generate and test methodology in (i) generating the space of possible positions according to semantic data aggregation criteria and (ii) exploring the search space by applying efficient solving techniques to compensate for the lack of information and to enforce constraint satisfaction and optimization.

For validating our approach we collected a considerable amount of sensor data by resorting to a simulation tool generating large sets of sensor data reflecting entire days of a person's household activities, matching real sensors installed in our lab. In the evaluation of our approach, we also generated and collected data in a real setting and compared results obtained on simulated data with the accuracy of our approach on real sensor data.

The properties of the environment and the characteristics of sensor data are described in Section 2, both considering the simulated setting and the real setting. Section 3 formalizes the localization and tracking problem in terms of properties of locations, metrics, reward functions and optimization criteria used to estimate a position. Section A introduces the ASP formalism and investigates our modeling and reasoning strategies, while results of our simulated and real experiments are presented in Section 5. A short discussion follows in Section 6.

2 Sensor Data for Indoor Positioning

In order to position an object or a device in an environment, the basic step is to use a reference point to determine the distance (or the angle) between the device and the reference point itself. Supposing that both device and reference objects can transmit radio signals, distances can be approximated by using the Receiver Signal Strength Indicator (RSSI). RSSI is based on the degradation of radio signals while traveling in space. The fact that no additional hardware is necessary to obtain RSSI leads to a very broad usage of this indicator.

In addition to RSSI, we use Passive Infrared (PIR) and Range Finders (RF). PIR is used to catch the movements of a person (or an object) that has a different temperature with respect to the surroundings while RF uses sonar to provide the distance of an object from the sensor.

We considered two different settings to collect sensor data and to test performance and effectiveness of knowledge-based indoor positioning. In a preliminary setting, we simulated the generation of sensor data, while in a further step we collected data of a person moving through our lab in order to validate the accuracy of our approach on real sensor data.

The remainder of this section describes both scenarios in terms of how the environment is represented and how sensor data are produced in each setting.

2.1 Data Simulation

In order to generate simulated RSSI, RF, PIR, and environmental sensor data, we implemented an agent-based data-generation tool using the Recursive Porous Agent Simulation Toolkit (Repast Simphony or Repast for short)[1].

Following a taxonomy of selected Agent-Based Modeling Tools [11], we chose to design and implement our data generation tool using Repast because it turned out to be the best combination of ease of model development and modeling power. Exploiting the Repast features, we are able to efficiently generate large data sets simulating days of the household activities of a person and corresponding sensor values.

We exploited the Repast simulator in order to generate data as close as possible to that of a real scenario. We obtained a sensor model for every data source (RSSI, PIR and Range finder) using real data and extended the simulator in order to generate data according to the sensor model modified by a small gaussian white noise.

The simulation scenario is represented by a Repast model that describes three grid-based projections and one network-based projection. The three $N \times M$ grid projections are used to define (i) the floor plan (walls, rooms, and areas), (ii) the position of localization sensors, and (iii) the position of environmental sensors together with how they propagate to adjacent cells. The network-based projection defines two undirected weighted graphs having *FloorCell* agents as nodes. The first graph is mainly used for the movement of the *Person* agent computed using Dijkstra's shortest path algorithm, where weights are used to avoid movements across wall cells.

The second grid projection is mainly used during the RSSI-data generation to verify if a direct line-of-sight between the person and the localization sensors exists. This information is useful in order to better simulate the RSSI propagation model. For this, the weights in the second grid projection are computed without considering walls. The line-of-sight information is computed comparing the results of the shortest path algorithm obtained using the two graphs. These results differ only if there is a wall between the localization sensor and a person.

The simulation scenario is specified in an XML-based scenario-definition file[2]. This scenario file is used to dynamically populate projections during the initialization phase of the Repast simulator, and it defines the following aspects of the simulation context:

environment: according to what is specified in this file, a *FloorCell* agent can be a *wall*, part of a *room-area*, or a *passage* (a cell connecting two areas of different rooms);

localization sensors: in addition to the sensor position on the grid and its orientation, the definition specifies the behavior of the passive-infrared and movement sensors in terms of cells covered;

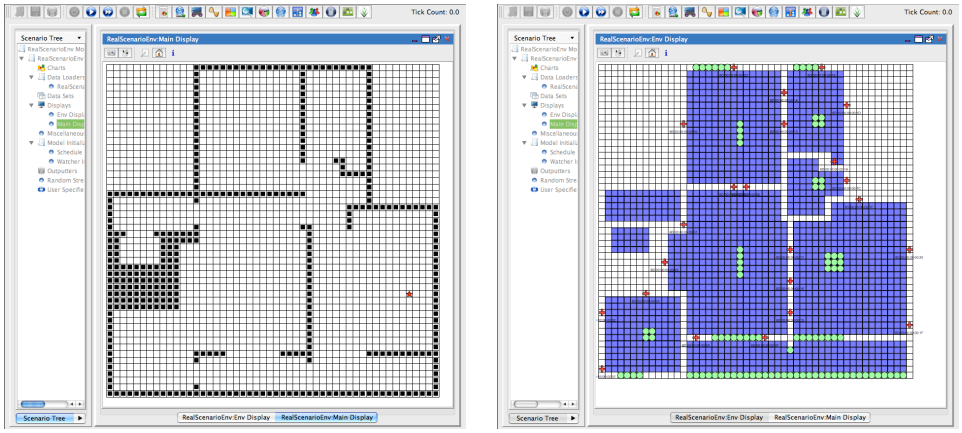
environmental sensors: during the data-generation phase, these sensors can read the temperature, humidity and brightness values;

light and heat sources: these elements define the properties of the objects that are used to manage and propagate the state of light and heat throughout the simulation scenario using a controlled flooding algorithm [7, 21]; the position and the initial status of windows and light switches is also defined.

[1] http://repast.sourceforge.net

[2] Note that Repast can load the required initialization data in several ways (e.g. using agent definitions directly stored in a relational database or in text files).

Fig. 1 Repast S Displays for Simulation



Floor Plan *Sensor Plan*

In addition, we provide an XML file containing a detailed list of actions we want the agent *Person* to perform. The actual version of the simulation tool include three types of elementary actions: going to a specific position in the grid, changing the state of lights and opening/closing doors and windows.

The data-generation is driven by two different event schedulers that are used to: (i) simulate the behaviour of the person and generate corresponding RSSI, PIR, and RF sensor data, (ii) simulate light, temperature, and humidity evolution during the day.

At every simulation tick, the *LocalizationSensor* agents compute RSSI, PIR, and RF considering the position of the agent *Person* in the floor plan. The measurement computation takes into account the model of every sensor: the radio propagation model for RSSI values and the presence of the person in a cell covered by the sensor for PIR and RF. *EnvironmentalSensor* agents generate temperature, humidity and light measurements periodically examining the state of the associated *EnvironmentCell* agent. All measurements are stored in a relational database.

The *DayManager* and *HeatManager* agents are used to manage the evolution of daylight and external temperature and humidity during the data-generation. Using the schedule we mentioned before, they modify the state of *LightSources* and *HeatSources* objects, which in turn modify the state of the associated *LightCells* and *HeatCells* objects.

Figure 1 shows the Repast toolkit executing different projection displays: the left side shows the main projection, walls (squares) and the person (dot); the right side gives the projection of sensor data, showing the light sources (circles), environmental sensors (crosses) and floor cells (squares).

In the first experimental evaluation, we did not consider environmental data, but they can be easily introduced as additional sensor information to disambiguate unclear locations through optimization criteria, as mentioned in Section 4.3.

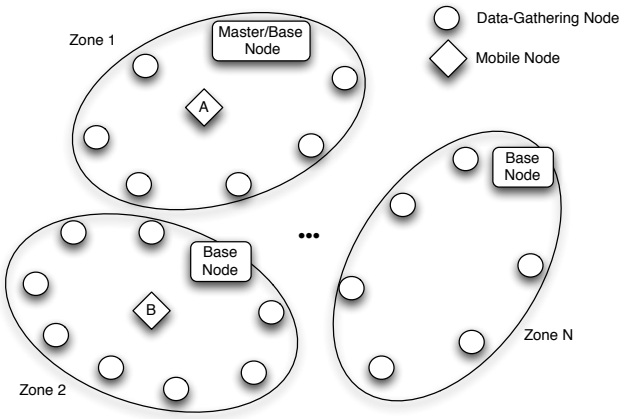


Fig. 2 Hierarchical Network Organization: the Infrastructure nodes and the Mobile nodes

2.2 Real Data Gathering

In order to gather localization information for tracking mobile nodes (or tags) and acquiring environmental data in the surrounding of the tags, we rely on a Data Gathering System, called DiGS [9]. Each node in DiGS is part of a Wireless Sensor Network (WSN) and constitutes a small hardware system consisting of a wireless microcontroller with several kinds of sensors. The WSN used in DiGS is organized hierarchically. The environment, where the data gathering is performed, is divided into several zones (or rooms) each containing a cluster of nodes.

As shown in Figure 2, every zone contains at least:

- a base node, in a fixed and known location, mostly used for network coordination but also with its own sensing capabilities;
- several data-gathering nodes (also fixed) used for localization and for improving the accuracy of the data gathering;
- zero to many mobile nodes (the "tags");

One of the base nodes is connected to a host computer and is called Master Node and behaves as a gateway towards the outside world. In this setting, base nodes manage data routing towards the master node using common WSN routing algorithms [13]. For large installation we considered the use of multi-gateway approaches. All the algorithms used in the DiGS system have been implemented using the IEEE 802.15.4 protocol and have been tested in a real setting with up to ten mobile nodes active in the same zone at the same time.

Further details about the hardware and the software used for our experiments can be found in [9].

The environment in which the target moves is a room in our lab; it is represented as a $N \times M$ grid in which each cell is a square of 0.25×0.25 centimeters. There are five sensors generating RSSI, PIR and RF in the room, placed as illustrated in Figure 3. There are five sensors placed along the walls, gathering RSSI, PIR and RF. The dimension of the room is 5×5 meters. In the real setting, RSSI is filtered using

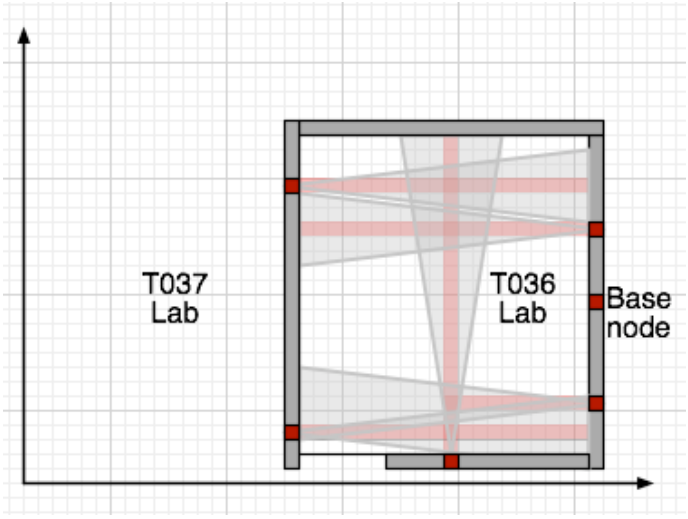


Fig. 3 Map of the Nomadis Lab T036 used for Real Data Gathering

triangulation but no further probabilistic techniques are applied to aggregate data. This should give us a better idea of the accuracy of the knowledge-based sensor fusion.

The PIR sensor has a typical rated detection distance of 5 m when the difference in temperature between the person and the background is more than $4^{\circ}C$. The detectable movement speed ranges from 0.5 to 1.5 m/s. The horizontal detection range (in degrees) is 100, whereas the vertical one is 82. This means that for each sensor device in the room, we have that the PIR signal covers a horizontal area that is 100 degrees wide, up to the opposite wall, and the RF signal covers all the cells along a straight line in front of it, up to the opposite wall.

During data collection in the real setting, we took as ground-truth the position computed by a Ubisense system[3]. The latter requires a long and detailed calibration process but its average localization performance is very interesting: an accuracy of 25 cm in 3D is attained for the 95% of the measurements whereas an accuracy of 14 cm is attained 50% of the times.

Details about how real instances have been created and analyzed are provided in Section 5.2.

3 The Localization and Tracking Problem

In this section we provide a formal description of the problem at hand, namely localization and tracking of a person moving in an indoor environment. In order to do that, we first describe the localization scenario and then give a definition of the problem in terms of how to select a location for a target at a given time. The description provided in this section does not depend on the specific implementation but it is needed to better understand the complexity of the problem at hand, as well as to present clear argu-

[3] The Ubisense RTLS, http://www.ubisense.net/

ments in favour of the knowledge-based approach in terms of flexibility, easy modeling and efficiency.

In our general formalization, we consider the environment as a discrete space represented by a grid of positions. Each possible position in our space can thus be represented as a cell that is identified by coordinates $\langle x, y\rangle$ on the grid. A cell can be part of (i) an area of a room (ii) a passage between two rooms or (iii) a wall.

To be more precise, we consider a set C of cells of form $\langle x, y\rangle$, where x, y are integers over a finite domain. The set of cells contains distinguished subsets, identifying rooms $(R_i)_{i\in I}$, areas $(A_j)_{j\in J}$, passages $(P_k)_{k\in K}$, and walls W. Cells are grouped into rooms and areas according to the following specification.

- rooms are disjoint set of cells, that is,

  $R_i \subset C$ for $i \in I$ and $R_i \cap R_j = \emptyset$ for $i \neq j \in I$

- areas are disjoint set of cells, that is,

  $A_j \subset C$ for $j \in J$ and $A_j \cap A_k = \emptyset$ for $j \neq k \in J$

- an area can be contained in one room only, that is,

  $A_j \subset R_i$ and $A_j \subset R_k$ implies $i = k$ for all $j \in J, i, k \in I$

- a room may contain multiple areas, that is,

  $\bigcup_{j\in J'} A_j \subset R_i$ for some $J' \subseteq J$ and each $i \in I$

Note that there can be cells of a room that are in none of the areas of the room.

Cells that do not belong to any room and are adjacent to cells belonging to two different rooms are referred to as *passages*.

- A set of cells $P \subset C$ is a passage between two rooms R_1 and R_2, if we have for all $\langle x, y\rangle \in P$ and $i = 1, 2$ that there is some $\langle x + j, y + k\rangle \in R_i$ such that $\langle x_1 + j', y_1 + k'\rangle \in P$ for all $j' < j$ or $k' < k$.

Finally, a cell $c \in C$ can also be unreachable or invalid (e.g. cells identified as part of a wall). We identify these cells as being part of a set $W \subset C$.

Sensors are placed in the environment so that they can detect proximity ($RSSI$), movement (PIR), or distance (RF) in a subset of cells. Such cells are said to be *covered* by the respective sensor (formally denoted by the static property $covered_S(c)$ below).

As mentioned in the previous sections, dynamic sensor data that are used for positioning at a given time step t are attached to locations $\langle x, y\rangle$. There are three kinds of dynamic sensor information we consider in our experiments:

- $RSSI$ or proximity signals are processed so as to return a set of locations where the mobile node is reached by the signal, associated to a likelihood, ranging from 1 to 100.
  We use $RSSI_t$ to denote cells reached by the proximity signal with likelihood l at time t, that is,

  $RSSI_t \subseteq \{(c, l, t) : c \in C, 1 \leq l \leq 100\}$.

– *PIR* or movement detectors are associated to a set of locations where the movement has been detected.
We use PIR_t to denote cells where a movement signal has been detected at time t, that is,

$$PIR_t \subseteq \{(c,t) : c \in C\} \ .$$

– *RF* or range finders are associated to a set of locations that are at a given distance from the *RF* sensor that produces the data.
We use RF_t to denote cells reached by the range finder signal at time t, that is,

$$RF_t \subseteq \{(c,t) : c \in C\} \ .$$

The problem of localization and tracking is concerned with the combination of all sensor information in a time interval, and with their interpretation for selecting the best candidate position at each time step. In order to characterize a subset of cells that are plausible positions at a given time t, we define c as a candidate position at time t if $c \in RSSI_t \cup PIR_t \cup RF_t$. Such cells are often referred to as *locations*. For identifying the positions representing best locations according to the available sensor information, we identify four properties (referred to as metrics) of locations, and we use these properties to characterize sets of (valid) locations as follows:

Best Proximity metric identifies location(s) with the highest closeness:

$$\mathcal{P}_t = \{c : (c,l,t) \in RSSI_t \text{ such that } l = max\{l' : (c',l',t) \in RSSI_t,\ c' \notin W\ \}\ \}$$

Best Support metric identifies location(s) with the highest support:

$$\mathcal{S}_t = \{c : (c,t) \in RF_t \cup PIR_t \text{ such that } support_t(c) = max\{support_t(c') : (c',t) \in RF_t \cup PIR_t,\ c' \notin W\ \}\ \}$$

where $support_t(c) = |(PIR_t \cup RF_t) \cap \{(c,t)\}|$ for $c \in C$.

Best Move metric identifies plausible location(s) at time t that are closest to a candidate location at time $t-1$:

$$\mathcal{M}_0 = c_0$$
$$\mathcal{M}_t = \{c : (c,t) \in RSSI_t \cup RF_t \cup PIR_t \text{ such that } dist(c, \mathcal{M}_{t-1}) = min\{dist(c', \mathcal{M}_{t-1}) : (c',t) \in RSSI_t \cup RF_t \cup PIR_t,\ c' \notin W\ \}\}$$

where c_0 is the initial position and for $C' \subseteq C$

$$dist(\langle x_1, y_1\rangle, C') = min\{|x_1 - x_2| + |y_1 - y_2| : \langle x_2, y_2\rangle \in C'\} \ .$$

Best Coherence metric identifies location(s) with the highest coherence:

$$\mathcal{C}_t = \{c : (c,t) \in RSSI_t \cup RF_t \cup PIR_t \text{ such that } coherence_t(c) = max\{coherence_t(c') : (c',t) \in RSSI_t \cup RF_t \cup PIR_t,\ c' \notin W\ \}\ \}$$

where for $c \in C$

$$coherence_t(c) = \begin{cases} \frac{|(PIR_t \cup RF_t) \cap \{(c,t)\}| * 100}{|covered_s(c), s \in \{PIR, RF\}|} & \text{if } \{covered_s(c), s \in \{PIR, RF\}\} \neq \emptyset \\ 0 & \text{if } \{covered_s(c), s \in \{PIR, RF\}\} = \emptyset, \\ & PIR_t \cup RF_t \neq \emptyset \\ n \in \mathcal{N} & \text{if } \{covered_s(c), s \in \{PIR, RF\}\} = \emptyset, \\ & PIR_t \cup RF_t = \emptyset \end{cases}$$

and n is a natural number expressing a more sensor-dependent notion of coherence referred to the reliability of our RSSI signals only. In our convention, it expresses the percentage of an RSSI value to indicate a correct location only on the basis of the signal strength.

We recall that $covered_s(c)$ is the property of c being covered by sensor s.

Example 1 We illustrate how sets of locations satisfying the above properties are identified in a concrete example.

Let $C = \{\langle x, y\rangle,\ x = 1..10, y = 1..10\}$ be the set of cells in our domain, and $W = \emptyset$ be the set of invalid cells. At a given time t, the following sensor data are collected about RSSI signals:

$$RSSI_t = \{(\langle 2,3\rangle, 10, t), (\langle 1,4\rangle, 20, t), ((\langle 2,4\rangle, 50, t), (\langle 2,5\rangle, 5, t))\}$$

Based on these information, best proximity is represented by $\mathcal{P}_t = \{\langle 2,4\rangle\}$
Let's now assume we collected additional sensor data at time t, namely:

$$RF_t = \{(\langle 2,4\rangle, t), (\langle 1,4\rangle, t)\} \qquad PIR_t = \{(\langle 2,3\rangle, t), (\langle 1,4\rangle, t)\}$$

We obtain, for these locations, that:

$$support_t(\langle 1,4\rangle) = 2 \qquad support_t(\langle 2,4\rangle) = 1 \qquad support_t(\langle 2,3\rangle) = 1$$

which gives us $\mathcal{S}_t = \{\langle 1,4\rangle\}$.

Now consider that $\mathcal{M}_{t-1} = \{\langle 1,3\rangle\}$, and we want to find the best move. We start by computing distances, and we obtain:

$$dist(\langle 2,3\rangle, \langle 1,3\rangle) = 1 \qquad dist(\langle 1,4\rangle, \langle 1,3\rangle) = 1$$
$$dist(\langle 2,4\rangle, \langle 1,3\rangle) = 2 \qquad dist(\langle 2,5\rangle, \langle 1,3\rangle) = 3$$

thus $\mathcal{M}_t = \{\langle 2,3\rangle, \langle 1,4\rangle\}$.

In order to identify locations with the best coherence property, we need to take into account static information about which location is covered by which sensor. Let us consider the following scenario:

$$covered_{RF}(\langle 1,4\rangle) \quad covered_{PIR}(\langle 1,4\rangle) \quad covered_{RF}(\langle 2,4\rangle) \quad covered_{PIR}(\langle 2,4\rangle)$$

obtaining

$coherence_t(\langle 1,4\rangle) = \frac{2*100}{2} = 100$
$coherence_t(\langle 2,4\rangle) = \frac{1*100}{2} = 50$
$coherence_t(\langle 2,3\rangle) = 0$

This gives us, as a result, that $\mathcal{C}_t = \{\langle 1,4\rangle\}$.

In order to solve the localization and tracking problem for every time step t of the interval under consideration, we need to select the best position among the plausible ones for every time step. In principle, whenever proximity values ($RSSI$) are available, we consider them as candidate solutions among which we select the best one. When this is not the case, all other available sensor data (PIR and RF in our experimental setting) should be considered to select the best candidate location.

Due to the incompleteness and noisiness of sensor data, we combine properties and metrics described earlier in this section in order to describe how we want to characterize the solution.

We identify four combination of properties (or *optimization criteria*) defining the following sets of possible locations c at time t:

$\mathcal{O}_t^1 = \mathcal{M}_t \cap \mathcal{C}_t$
$\mathcal{O}_t^2 = \mathcal{C}_t$
$\mathcal{O}_t^3 = \mathcal{M}_t$
$\mathcal{O}_t^4 = (RSSI_t \cup RF_t \cup PIR_t) \setminus (W \times \{t\})$

We introduce a preference relation among locations by imposing a ranking on the above criteria: $\mathcal{O}_t^i$ is preferred to $\mathcal{O}_t^j$, whenever $i < j$. This relation is specified according to the combination of properties we prefer to be verified by the location that is selected as the final solution to the localization and tracking problem. Accordingly, we can finally define the solution to the localization and tracking problem:

$$\mathcal{L}_t = \begin{cases} \mathcal{P}_t \cap \mathcal{O}_t^i & \text{if } RSSI_t \neq \emptyset \text{ and } \mathcal{O}_t^j = \emptyset \text{ for each } j < i \\ \mathcal{S}_t \cap \mathcal{O}_t^i & \text{if } RSSI_t = \emptyset \text{ and } \mathcal{O}_t^j = \emptyset \text{ for each } j < i \\ \emptyset & \text{otherwise} \end{cases}$$

Example 2 Let us consider the scenario illustrated in Example 1. Locations are identified by our optimization criteria are as follows.

$\mathcal{O}_t^1 = \mathcal{M}_t \cap \mathcal{C}_t = \{\langle 1,4\rangle\}$
$\mathcal{O}_t^2 = \mathcal{C}_t = \{\langle 1,4\rangle\}$
$\mathcal{O}_t^3 = \mathcal{M}_t = \{\langle 1,4\rangle, \langle 2,3\rangle\}$
$\mathcal{O}_t^4 = RSSI_t \cup RF_t \cup PIR_t = \{\langle 2,3\rangle, \langle 2,4\rangle, \langle 2,5\rangle, \langle 1,4\rangle\}$

As a result, we obtain

$\mathcal{L}_t = \mathcal{P}_t \cap \mathcal{O}_t^1 = \{\langle 1,4\rangle\}$

Note that solution can be influenced by our ordering of criteria as well as by the presence of invalid cells. In fact, if we had that $W = \{\langle 1,4\rangle\}$, this would change our properties on sets of locations as follows:

$\mathcal{P}_t = \{\langle 2,4\rangle\}$ $\mathcal{S}_t = \{\langle 2,4\rangle, \langle 2,3\rangle\}$ $\mathcal{M}_t = \{\langle 2,3\rangle\}$ $\mathcal{C}_t = \{\langle 2,4\rangle\}$

As a consequences, we would obtain

$\mathcal{O}_t^1 = \emptyset \qquad \mathcal{O}_t^2 = \{\langle 2,4\rangle\} \qquad \mathcal{O}_t^3 = \{\langle 2,3\rangle\} \qquad \mathcal{O}_t^4 = \{\langle 2,3\rangle, \langle 2,4\rangle, \langle 2,5\rangle\}$

and, as result $\mathcal{L}_t = \mathcal{P}_t \cap \mathcal{O}_t^2 = \{\langle 2,4\rangle\}$.

All in all, in our formal specification, whenever the best move metric is applied, we select the best location at each time step t depending on the location selected at time step $t-1$. In this way we prefer steady solutions. An alternative approach would be to specify the tracking problem using global optimization over the entire (or partial) temporal interval. As we will better illustrate in Section A, our knowledge-based approach is flexible and expressive enough to make this adaptation possible with little effort.

4 Knowledge-Based Indoor Positioning

In this section, we illustrate how we implemented the localization and tracking problem formalized in Section 3 in the knowledge-based framework of Answer Set Programming (ASP). See Appendix A for a brief introduction to ASP.

As a general principle, we recall that RSSI values as well as other sensor information are aggregated with ad-hoc algorithms for feature analysis.

The knowledge-based support to localization and tracking tackles the problem that data gathered by sensors may be noisy even after aggregation, but their combination may yield a more reliable interpretation. The expressive power of ASP is used to disambiguate unclear situations (e.g., where the person is) by combining heterogeneous data sources and using defaults and qualitative optimization to select the best candidates. To do this, the localization scenario and the sensor data are represented as logic predicates, while qualitative criteria are specified and combined using logic rules to reason about localization and tracking.

4.1 ASP Basics

We assume the reader to be familiar with the terminology and basic definitions of ASP (see [2] for details). In what follows, we rely on the language supported by grounders *lparse* [17] and *Gringo* [6], providing normal and choice rules, cardinality and integrity constraints, as well as aggregates and optimization statements. As usual, rules with variables are regarded as representatives for all respective ground instances.

4.2 Modeling the Localization and Tracking Problem in ASP

The logical description of the localization scenario is provided in terms of interesting properties of the sensed information. The space is a grid where each cell (or location) $c = \langle x, y\rangle$ is identified by the term $loc(X, Y)$. Each cell can be associated to (i) a room/area, (ii) a passage, or (i) a wall, and it is attached to some static properties. The association and the properties formally defined in Section 3 are mapped into logic predicates as detailed in Table 1, where $c = \langle X, Y\rangle$.

Logic Predicate (Static)	**Formal Description**
cell(X,Y,Room,Area)	$c \in Area,\ Area \subset Room$
passage(X,Y,R1,R2)	$c \in\ P,\ P$ is a passage between R1 and R2
wall(X,Y)	$c \in W$
sensed_type(S)	$S \in \{PIR, RF\}$
covered(movement,loc(X,Y))	$covered_{PIR}(c)$
covered(distance,loc(X,Y))	$covered_{RF}(c)$

Table 1 Logic Predicates representing static properties of cells

Logic Predicate (Dynamic)	**Formal Description**	**Sensor Data**
sensed(rssi,loc(X,Y),T,P)	$c \in RSSI_t$	Proximity signal
sensed(pir,loc(X,Y),T)	$c \in PIR_t$	Movement detector
sensed(rf,loc(X,Y),T)	$c \in RF_t$	Distance detector

Table 2 Logic Predicates representing sensor data collected at each time step

A location is *invalid* as a candidate solution when it is a wall [4]:

```
invalid(loc(X,Y)) :- wall(X,Y).
```

Dynamic sensor information associated to locations can come from (i) processed RSSI values with an associated percentage of likelihood, (ii) movement detectors (PIR), and (iii) distance measures (RF). Logic predicates representing dynamic sensor data are listed in Table 2.

At each time step, a cell c becomes a candidate location L at time T when any of the sensor information (RSSI, PIR or RF) has been received for that location, unless it is found to be invalid via $invalid(L)$. This allows us to immediately reduce the set of possible solutions to the positions for which at least one sensor signal is available. The corresponding ASP code is as follows:

```
location(L,T) :- sensed(rssi,L,T,P), not invalid(L).
location(L,T) :- sensed(S,L,T), not invalid(L).
```

Besides the information about received sensor data at a given time step T for a location L, dynamic properties of *support*, *distance* measure between two consequent admissible moves, and *coherence* for a location as a candidate position at a give time step are mapped into logic predicates as illustrated in the first part of Table 3.

Discrete time is in seconds and sensor data are opportunely aggregated and provided when values change beyond a given threshold.

Earlier in this section, we mentioned that positions obtained by processing RSSI signals represent the initial set of candidate solutions to the problem of estimating the actual position. When this information is missing, tracking the person on the grid becomes more difficult and the solution space can be huge because the model of movement may produce a higher number of possibilities. We have to face this problem not

[4] The characterization of invalid locations can be easily extended by introducing appropriate predicates and rules in the declarative ASP framework

Logic Predicate (Properties)	**Formal Description**
$support(loc(X,Y),N,T)$ $dist(loc(X_1,Y_1),loc(X_2,Y_2),D)$ $coherence(loc(X,Y),T,C)$	$support_t(\langle X,Y\rangle) = N$ $dist(\langle X_1,Y_1\rangle,\langle X_2,Y_2\rangle) = D$ $coherence_t(\langle X,Y\rangle) = C$
Logic Predicate (Metrics)	**Formal Description**
$best\_move(loc(X,Y),T)$ $best\_coherence(loc(X,Y),T)$ $max\_support(loc(X,Y),N,T)$ $max\_proximity(loc(X,Y),P,T)$	$\langle X,Y\rangle \in \mathcal{M}_t$ $\langle X,Y\rangle \in \mathcal{C}_t$ $\langle X,Y\rangle \in \mathcal{S}_t,\ support_t\langle X,Y\rangle = N$ $\langle X,Y\rangle \in \mathcal{P}_t,\ (\langle X,Y\rangle,P,T) \in RSSI_t$

Table 3 Logic Predicates representing dynamic properties and optimization metrics

only in case of malfunctioning sensors, but also when the RSSI processing cannot provide acceptable candidate positions for several sequential time steps. In such settings, the ASP-based reasoning process is flexible enough to compensate noisy and missing sensor values, reasoning about the qualitative analysis of the available information, as detailed in the next subsection.

4.3 Multi-Criteria Optimization in ASP

The solution to a localization and tracking problem is based on the definition of properties we want to be verified for candidate solutions according to available sensor data. In order to combine all these properties and find the best solution(s), in Section 3 we introduced *best proximity*, *best support*, *best move* and *best coherence* optimization metrics, combined into four optimization criteria $\mathcal{O}_i, i = 1..4$.

The *best proximity* metric considers positions provided by RSSI that have the highest likelihood P. Unfortunately it is not always true that the higher the likelihood associated to RSSI, the closer the candidate position is to the real one; for this reason we define optimization criteria that do not take the best proximity into account to validate candidate positions.

Rather, best proximity is used as a reward function to select a location L once $\mathcal{O}_i$ is identified as the most preferred optimization criteria and locations $L_i \in \mathcal{O}_i$ are the preferred ones. When RSSI signals are not available, a different reward function is used, which is closely related to the notion of *support* for a location $L_i \in \mathcal{O}_i$: the higher the support for L_i, the better L_i is as a candidate position for criterion $\mathcal{O}_i$.

Optimization metrics are expressed in form of logic predicates as indicated in the second part of Table 3.

Best Coherence maximizes the value of the *coherence* property, which is a percentage indicating how many of the sensor signals covering a location L are effectively captured for that location at time T; if L is not covered by any sensor signal other than the RSSI, its coherence is fixed at 80% for all time steps, according to a probabilistic model of our proximity radio signals; in the same way, coherence is

equal to 0 at time T if L is covered by at least one sensor (different than RSSI), and no sensor value other than RSSI is captured at time T for location L, as follows from the definition of *coherence* given in Section (3).
The correspondent ASP encoding uses aggregates as follows:

```
ex_support(N,L)  :- N = #count [covered(S,L) : sensed_type(S)], location(L,T), N>0.
support_dom(N)   :- ex_support(N,L).
p_support(L,T,C) :- C = #count [sensed(S,L,T) : covered(S,L)], location(L,T),
                    not invalid(L), C>0.
p_support_dom(C) :- p_support(L,T,C).
coherence(L,T,P) :- p_support(L,T,C), P=(100*C)/N, ex_support(N,L).
coherence(L,T,C) :- not p_support(L,T,C), ex_support(N,L), location(L,T),
                    not invalid(L), p_support_dom(C).
coherence(L,T,80):- not p_support(L,T,C), not ex_support(N,L), location(L,T),
                    not invalid(L), p_support_dom(C), support_dom(N).

most_coherent_time(T,M) :- M = #max [coherence(L,T,C) : coherence(L,T,C) = C],
                           time(T), M>=0.
best_coherence(L,T)     :- most_coherent_time(T,C), coherence(L,T,C).
```

Best Move property identifies the location L at any time step T for which the distance between L and the location L_1 at time step $T-1$ is minimum.
The ASP encoding is as follows:

```
distance(D,loc(X,Y),T) :- location(loc(X,Y),T), at(loc(U,V),T-1), D = #abs(X-U) + #abs(Y-V).

distance(D,T)   :- distance(D,L,T).
dist(X,T)       :- distance(X,T).
dist(X-1,T)     :- dist(X,T), X>0.

neg_smallest(X+1,T) :- dist(X+1,T), distance(X,T).
neg_smallest(X,T)   :- dist(X,T), neg_smallest(X-1,T).

best_distance(D,T) :- distance(D,T), not neg_smallest(D,T).
best_movement(L,T) :- distance(D,L,T), best_distance(D,T).
```

The specification of properties and metrics in ASP makes it straightforward to implement the criteria defined in Section 3 as follows:

```
criterion(1..4).

criterion(1,L,T) :- location(L,T), best_movement(L,T), best_coherence(L,T).
criterion(2,L,T) :- location(L,T), best_coherence(L,T).
criterion(3,L,T) :- location(L,T), best_movement(L,T).
criterion(4,L,T) :- location(L,T).
```

The preference relation between two criteria is defined by their numbering such as $\mathcal{O}_i$ is preferred to $\mathcal{O}_j$ ($\mathcal{O}_i \succ \mathcal{O}_j$) when $i > j$.

This ordering is motivated by the fact that we want to obtain the solution satisfying all metrics as the best option, but whenever such a solution does not exist, we prefer to give up the movement model and consider coherence as more reliable[5]. Alternative combinations are also possible and this can be useful to compare them in order to understand which property seems to be stronger or which sensor data appears to be more reliable.

[5] Note that this is one of the possible choices. We can easily change the ordering between criteria to see how reliable our sensors are.

When new sensor information is introduced, we can easily re-define or extend the list of properties, metrics and criteria. An example can be the introduction of environmental data in the definition of coherence: to take such data into account, we just have to introduce it in the notion of *support*. We can also specify them as a different property in case we want to define a separate optimization criteria that will be less/more preferred than the existing ones. This flexibility is a clear advantage of the declarative knowledge-based approach.

In order to implement optimization on ordered criteria, we compute two reward functions represented by the highest proximity and the highest support for a criterion $\mathcal{O}_i$ at time T, respectively.

The best location L for a time step T is represented by predicate $best\_location(L, T)$ and it can now be identified by selecting the best value returned by the application of the most preferred criterion.

```
% reward functions
max_value(C,B,T) :- max_proximity(C,B,T).
max_value(C,B,T) :- max_support(C,B,T).

% overall preferences
best_value(C,B,T) :- max_value(C,B,T), not worse_value(C,T).
worse_value(C,T)  :- max_value(C,B,T), max_value(C1,B1,T), C1<C.

best_location(L,T) :- best_value(C,B,T), criterion(C,L,T), sensed(rssi,L,T,B).
best_location(L,T) :- best_value(C,N,T), criterion(C,L,T), sensed(S,L,T),
                      not has_rssi(T).
has_rssi(T)        :- sensed(rssi,L,T,P).
```

Once all properties, metrics and criteria have been expressed, our problem specification generates the space of solutions and then selects the best one via optimization. This is done by applying the generate and test approach of ASP: a cardinality constraint specifies that for each time step T, exactly one location is selected (generating part) and an integrity constraint specifies that a selected location L for a time step T has to be the best location for T (testing part) as follows:

```
1 [ at(L,T) : location(L) ] 1 :- time(T).
:- at(L,T), not best_location(L,T).
```

The actual implementation is evaluated using the solver *Clasp* [5] on grounded logic programs obtained by using *Gringo* [6] as a grounder. *Clasp* relies on modern Boolean Constraint Technology, matching the performance of state-of-the-art SAT solvers.

5 The Experimental Evaluation

In the evaluation of our approach, we focus our attention on the following three aspects that will be discussed for both the simulated and the real setting.

Accuracy. According to [21], accuracy is the degree to which the random variation is centered on the true value. The overall *accuracy* of results is determined via the Mean Absolute Error (MAE) proposed in [3] for measuring the accuracy of localization with respect to the true ground position. MAE is very similar to the common Root Mean Square and consists of computing the residual error between

the estimated and actual node positions for every node in the network, sum them and average the result, as shown in the following equation:

$$MAE = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2}}{n} \quad (1)$$

where $(\bar{x}_i, \bar{y}_i)$ are the coordinates of the target's estimated positions and (x_i, y_i) the true ground ones.

Fault tolerance. An interesting property we want to verify in our approach is how robust it is against missing data; this aspect is discussed by analyzing the impact of missing data on the accuracy.

Effectiveness. In order to validate the effectiveness of optimization criteria, we illustrate how they can help in reducing noise and filtering localization results provided by RSSI, PIR and RF in the real test setting.

Accuracy and fault tolerance are discussed for the simulated setting and confirmed by experiments on real data, while the effectiveness is clearer if we look at the real setting where initial data are more noisy and error prone.

5.1 Results on Simulated Data

The first experimental evaluation of our knowledge-based approach to position estimation is carried out on data generated by the simulation tool as described in Section 2.1.

Our experiments are based on the generation of several test instances of different size using the agent-based simulation toolkit Repast Simphony. According to the size of the instance we want to generate, we simulate movements of a person across one or more rooms of the house through one or more midpoints. Repast applies a simple shortest-path algorithm for simulation, based on the map of the environment and realistic motion model. The instance size is determined by the number of time steps which corresponds to the number of moves, given that each move is associated to a new time step. In our analysis, we make the assumption that the number of sensors (thus the amount of sensor data received at each time step) is fixed according to a given sampling rate, and it is linear in the number of rooms and doors in the environment (we consider putting four to six sensors per room plus one sensor for each passage). For this reason, we did not consider the amount of data produced by sensors at each time step as a factor that may impact performance.

The MAE indexes obtained through Particle Filters (PF) based on the RSSI signals only, are compared with the MAE index obtained by ASP using RSSI signals, Range Finder (RF), and Passive Infrared (PIR) measurements.

Results are illustrated in Figure 4, where we plot both MAE obtained by Particle Filter and MAE obtained by the ASP multi-criteria approach on instances of several size, from 20 to 180 time steps. MAE is expressed in meters and the instance size is expressed in terms of subsequent time stamps.

We can observe that the introduction of additional (noisy) data sources considered in the ASP-enhanced approach, still give us a lower MAE for small instances, where size is smaller than 50 time steps. This can be explained by the fact that the PF algorithm re-allocates the particles used for position estimation at each time step; as a consequence, it is common that on small instances the particles are still a bit sparse and the precision of the algorithm is lower. For a similar reason, the MAE index obtained

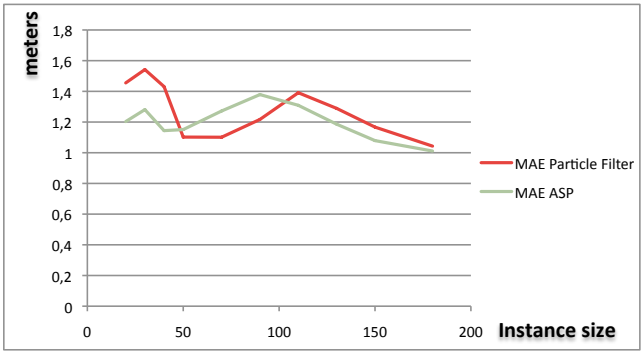


Fig. 4 Accuracy of ASP vs PF on simulated data

by the PF algorithm slightly decreases as size rises: on bigger instances, particles tends to converge on the more plausible positions, resulting in higher accuracy.

A similar behavior is observed for the MAE of the ASP curve, validating the ability of the approach to merge heterogeneous sensor data for indoor positioning, including them in the model in a flexible way.

In general both MAEs tend to decrease (with ASP being slightly more accurate) and converge. More extensive experiments are needed to better characterize MAE's curves, but our preliminary analysis aims at showing how the localization and tracking problem can be successfully solved in a more flexible, declarative and extensible way. We have been able to obtain acceptable results compared to traditional approaches, but in a framework where the problem is easier to model and can be adapted and used to test properties of sensors such as their reliability, as well as properties of the target, such as the movement model.

Another aspect we took into account is robustness, evaluated in terms of how the accuracy can be affected by missing data.

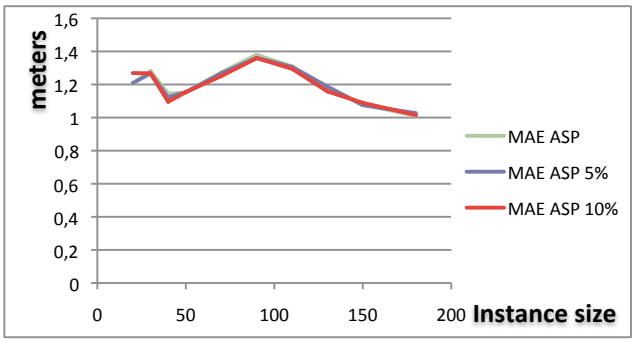
A lack of data can either be caused by sensors not properly transmitting or by a too sparse set of particles, so that the PF algorithm cannot provide candidate positions. Considering missing data scenarios is interesting in view of the fact that we want our approach to be able to provide candidate solutions even in case of malfunctioning sensors or weak PF results.

In order to evaluate robustness, we used the same instances generated in the simulated setting, randomly removing 5% and 10% of the sensor data. Results of all three versions of the instances are illustrated in Figure 5.

The main aspect proving robustness of our approach can be seen if we consider the almost total overlapping of the three curves in Figure 5: on instances of the smaller size, the MAE index is slightly higher with 10% of missing sensor data (as one may intuitively expect), but this phenomenon gets stable on bigger instances where missing data up to 10% do not affect accuracy.

We observe that in general the MAE index is stable when we remove data, probably due to the fact that our knowledge-based approach makes it possible to estimate a position by using the model of movement and the other metrics and optimization criteria to reduce the search space, compensate for missing information, and propose plausible estimations. Furthermore, in the ASP approach the position of the target can

Fig. 5 Robustness of ASP on simulated data



be guessed according to the optimization criteria described in Section A, which turned out to be more stable and flexible[6] than particle re-distribution when there are missing data.

We cannot estimate yet the persistence of robustness properties over higher percentage of missing data. In further analysis it would be interesting to estimate a threshold telling us how far we can go in removing data before MAE starts to increase. Intuitively, it depends on how many different kinds of sensor signals we have, and how noisy are the data they produce: the higher the number of sensors or the noise, the higher the threshold, while if we have lots of reliable sensor data, we would expect this threshold to be lower.

5.2 Results on Real Data

In the real setting we generated instances for 4 different paths of a person moving through our Lab as illustrated in Figure 6. In order to compensate for the eventual small lack of precision of the Ubisense system taken as a ground-truth, we collected 10 repeated instances for each of the paths and computed the average position for each time step.

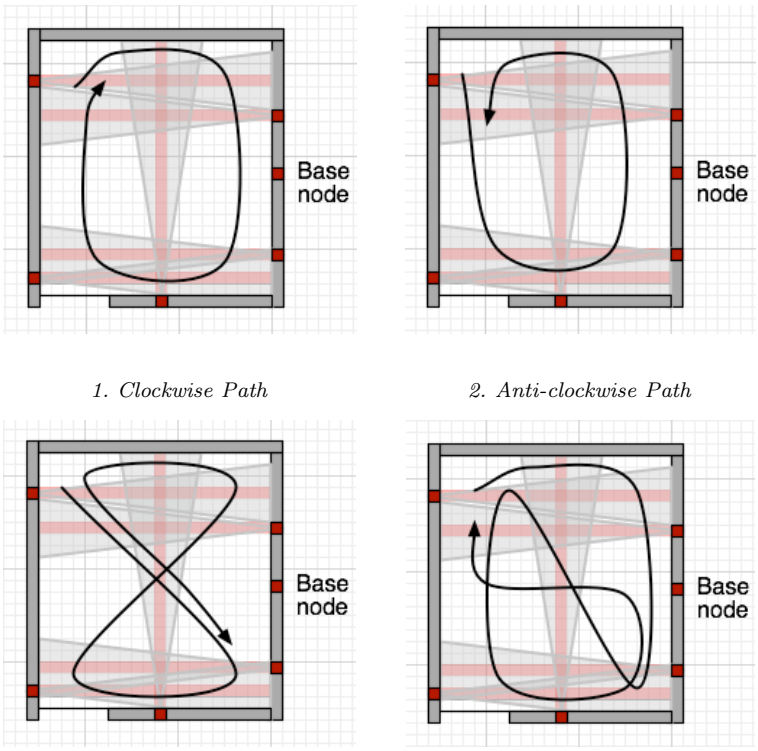
The first aspect to be evaluated in the real setting is the accuracy. For each of the paths, we compared the MAE index obtained by PF with the MAE index obtained by ASP. For all paths, we obtained positioning with 50 centimeters average error, versus more than 1.4 meters obtained by Particle Filter. Results are summarized in Table 4, where the MAE is expressed in meters.

Data generation and data corruption in this setting is less controllable than in the simulated scenario. For this reason we did not evaluate robustness but rather focused on showing the effectiveness of the approach by comparing the ground-truth position and the position estimated by PF and by ASP along time steps.

Results of this data analysis are reported in Figures 7 and 8 for Particle Filter and ASP, respectively.

[6] We can change the order of optimization criteria and their definition according to the performances of our sensor devices.

Fig. 6 Movements of a target in the real setting



1. Clockwise Path *2. Anti-clockwise Path*

3. Zig-zag Path *4. Random Path*

Table 4 Accuracy results on real data

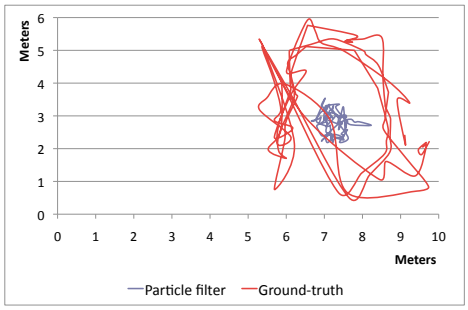
Path	MAE (Particle) RSSI Only	MAE (ASP) RSSI, PIR, RF
Clockwise	1.553	0.478
Anti-clockwise	1.856	0.495
Zig-zag	1.611	0.453
Random	1.482	0.415

Consider that in the ASP results we mapped coordinates expressed in meters into cells, where the dimension of each cell is 0.25 meters. Despite this mapping, in Figure 8 we can see how results of ASP are closer than results of PF to the ground-truth positions. Smaller cells would make it possible to obtain higher accuracy and to verify it more clearly on the plot.

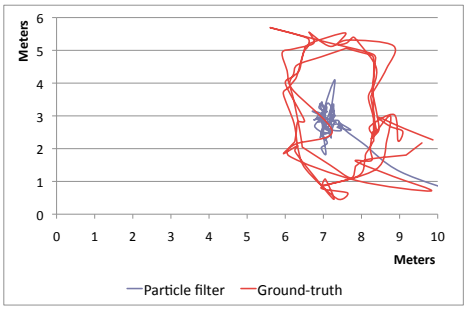
6 Discussion

We presented a hybrid approach to indoor position estimation by combining and extending quasi-standard quantitative methods in a knowledge-based yet qualitative

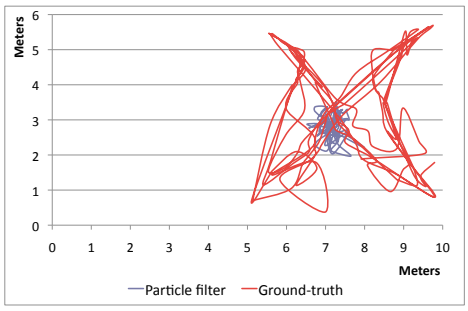
Fig. 7 Movements of a target in the real setting: PF results



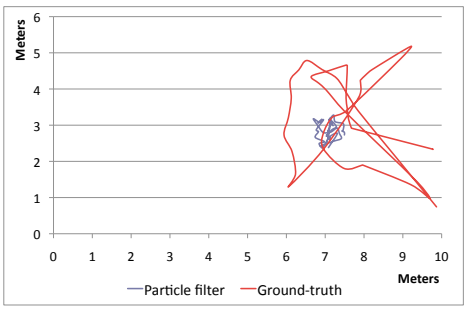
Tracing Clockwise Path



Tracing Anti-clockwise Path



Tracing Zig-zag Path



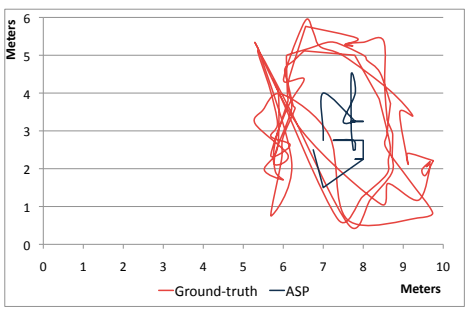
Tracing Random Path

framework. To this end, we took advantage of the knowledge representation and reasoning capacities of ASP for providing a rich model combining various disconnected sensor models with further semantic information, like a movement model and sensor dependencies. The resulting ASP encoding combines the available heterogeneous sensor data in a transparent and easily modifiable way and offers an impressive robustness in dealing with noisy and (partially) absent sensor data, as indicated by our experiments.

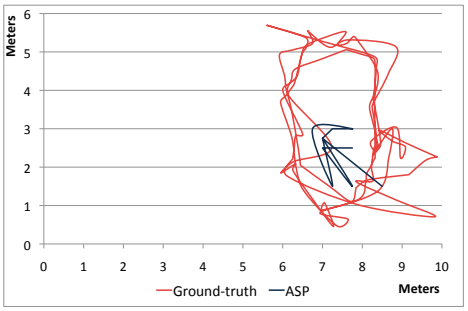
The main idea is that if we can easily introduce additional sensors as data sources in our model, missing data can be better compensated by the knowledge-based approach, as long as the noisiness of data does not grow significantly with the number of sensors. Time steps with no data do not determine a redistribution of the particles as in the PF algorithm: in our approach, the position in a time step for which no data have been collected, are guessed according to the knowledge-based optimization criteria applied to tracking in previous and following time steps. Preliminary tests on simulated data show that our approach can be as accurate as standard techniques based on particle filters, while more recent tests on real data show that the knowledge-based approach is 66% more accurate. This difference is due to the fact that simulated data were "cleaner" than the real ones, thus it was not possible to see the real improvement given by the ASP approach in dealing with noisy and partial data.

The knowledge-based approach presented in this paper makes accuracy robust against incomplete information as illustrated by our tests on simulated data. One may argue that if data are not available for several time steps, the combinatorial explosion of possible solutions can make the search process inefficient. Even in this case, opti-

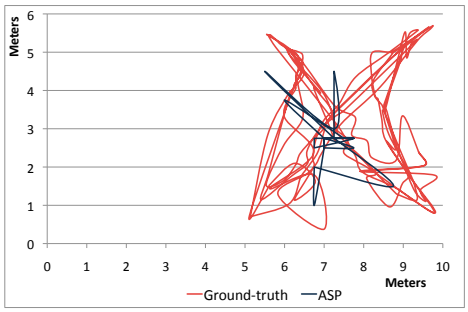
Fig. 8 Movements of a target in the real setting: ASP results



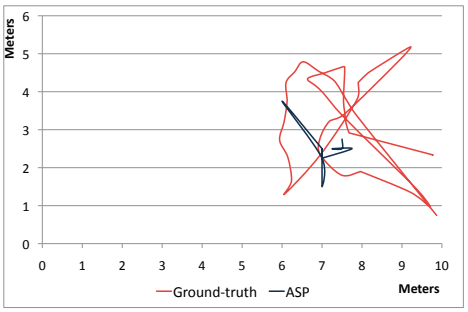
Tracing Clockwise Path



Tracing Anti-clockwise Path



Tracing Zig-zag Path



Tracing Random Path

mization criteria can help selecting the best candidate positions in the search space as the generate and test approach of ASP, guided by optimization criteria, makes it possible to efficiently prune the search space and isolate only a subset of acceptable solutions.

The effectiveness of our approach can be validated by observing that changing the order of criteria could help evaluating the reliability of sensor data or the soundness of the properties we defined according to commonsense knowledge. As an example, we found out that values provided by PIR and RF sensors are very reliable, thus if we switch criteria $\mathcal{O}_2$ and $\mathcal{O}_3$ giving higher preference to the movement model than the coherence property, the MAE slightly increases. We are now working on the specification of additional criteria in order to further investigate this aspect.

Our approach makes it also easier to add new domain-knowledge in form of constraints or optimization measures, which can be used to strengthen the choice of some candidate positions over others going beyond what sensor data tells us. The encoding discussed in Section A allows for easy customization and extensibility. All in all, the elaboration tolerance of the high-level ASP specification makes the major difference of our approach to potential alternatives, and it seems hard to envisage in a purely quantitative settings.

A more general, though very important issue, is the trade-off between flexibility and accuracy of the model. In order to investigate this aspect, we should make fur-

ther experiments on how additional sensor data as well as new criteria or different optimizations may affect accuracy.

Finally, our current implementation accumulates data in hourly intervals. This choice was guided by the fact that we wanted to consider a sufficiently long time window where we can show both the power of ASP inference over a rich and error-prone data instance, and the errors compensation obtained by multicriteria optimization approach even when errors cumulate over a significantly long time window.

We plan to refine this to use the incremental ASP solver *iClingo* [4] in order to provide a much more fine-grained approach to localization and tracking, eventually aiming at real-time conditions.

A Answer Set Programming

This section provides a brief introduction to ASP (see [2] for details), a declarative paradigm for knowledge representation and reasoning, offering a rich modeling language along with highly efficient inference engines based on Boolean constraint solving technology. The basic idea of ASP is to encode a problem as a logic program such that its answer sets represent solutions to the original problem.

The methodology of writing programs in ASP follows a *generate-and-test* approach, inspired by intuitions on *NP* problems. That is, a "generating" part is meant to non-deterministically provide solution candidates, while a "testing" part eliminates candidates violating some requirements.[7] In addition, one may specify *optimization* criteria via lexicographically ordered objective functions.

Syntactically, a logic program is a set of rules of the form

$$h \leftarrow a_1, \ldots, a_m, not\ a_{m+1}, \ldots, not\ a_n \tag{2}$$

where a_i, for $1 \leq m \leq n$, is an *atom* of the form $p(t_1, \ldots, t_k)$, and $t_1, \ldots, t_k$ are terms, viz., constants, variables, or functions. In practice, '$\leftarrow$' is written as '`:-`'.

First-order representations, commonly used to encode problems in ASP, are only informally introduced. The following example is taken from Section 4.3:

```
best_value(C,B,T) :- max_value(C,B,T), not worse_value(C,T).
```

This rule stands for all ground instantiated rules obtained by systematically replacing all variables with appropriate constants and ground terms.[8]

For a rule r as in (2), the *head* h of r is either an atom, a *cardinality constraint* of the form $l\,\{h_1, \ldots, h_k\}\,u$ in which l, u are integers and $h_1, \ldots, h_k$ are atoms, or the special symbol $\bot$ (omitted in practice). If h is a cardinality constraint, we call r a *choice rule*, and an *integrity constraint* if $h = \bot$.

Examples of the two latter types of rules are given at the end of Section 4.3, viz.

```
1 [ at(L,T) : location(L) ] 1 :- time(T).
 :- at(L,T), not best_location(L,T).
```

The cardinality constraint `1 [ at(L,T) : location(L) ] 1` contains a "conditional literal", in which the instantiation of `L` in `at(L,T)` is limited by the instances of `location(L)`. Using brackets rather than parentheses indicates a multi-set interpretation of the constituent literals. For clarity, one may add the keyword `#count`. See [17, ?] for a detailed description of the language.

Semantically, the answer sets of a program are particular classical models of the program satisfying a certain stability criterion (cf. [2]).

[7] Note that this is only a methodological approach; it is neither syntactically required nor computationally pursued.

[8] Note that only finitely many ground rules are obtained although there are infinitely many ground terms.

Acknowledgements The second author was partly funded by the German Science Foundation (DFG) under grant SCHA 550/8-2.

The authors are grateful to Martin Gebser and Roland Kaminski for valuable suggestions on improving ASP encodings.

References

1. The Ubisense RTLS. http://www.ubisense.net/
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
3. Broxton, M., Lifton, J., Paradiso, J.A.: Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation. SIGMOBILE Mobile Computer and Communication Review **10** (2006)
4. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental ASP solver. In: M. Garcia de la Banda, E. Pontelli (eds.) Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08), *lncs*, vol. 5366, pp. 190–205. springer (2008)
5. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: clasp: A conflict-driven answer set solver. In: Ninth International Conference on Logic Programming and Nonmonotonic Reasoning, pp. 260–265. Springer-Verlag (2007)
6. Gebser, M., Schaub, T., Thiele, S.: Gringo : A new grounder for answer set programming. In: LPNMR, pp. 266–271 (2007)
7. Hedetniemi, Hedetniemi, Liestman: A survey of gossiping and broadcasting in communication networks. NETWORKS: Networks: An International Journal **18** (1988)
8. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME Journal of Basic Engineering (82 (Series D)), 35–45 (1960)
9. Merico, D.: Tracking with high-density, large-scale wireless sensor networks. Ph.D. thesis, University of Milano-Bicocca, Dottorato di ricerca in INFORMATICA, 22 (2010-02-03). URL http://hdl.handle.net/10281/7785
10. Nakamura, E.F., Loureiro, A.A.F., Frery, A.C.: Information fusion for wireless sensor networks: Methods, models, and classifications. ACM Comput. Surv. **39**(3), 9 (2007)
11. North, M.J., Macal, C.M.: Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation. Oxford University Press, Inc., New York, NY, USA (2007)
12. Patwari, N., Ash, J., Kyperountas, S., Hero A.O., I., Moses, R., Correal, N.: Locating the nodes: cooperative localization in wireless sensor networks. Signal Processing Magazine, IEEE **22**(4), 54 – 69 (2005). DOI 10.1109/MSP.2005.1458287
13. Perkins, C.E., Royer, E.M.: Ad-hoc on-demand distance vector routing. In: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications, WMCSA '99, pp. 90–. IEEE Computer Society, Washington, DC, USA (1999). URL http://portal.acm.org/citation.cfm?id=520551.837511
14. Ristic, B., Arulampalam, S., Gordon, N.: Beyond the Kalman Filter: Particle Filters for Tracking Applications. Artech House (2004)
15. Shen, Y., Win, M.: Fundamental limits of wideband localization #x2014; part i: A general framework. Information Theory, IEEE Transactions on **56**(10), 4956 –4980 (2010). DOI 10.1109/TIT.2010.2060110
16. Shen, Y., Wymeersch, H., Win, M.: Fundamental limits of wideband localization #x2014; part ii: Cooperative networks. Information Theory, IEEE Transactions on **56**(10), 4981 –5000 (2010). DOI 10.1109/TIT.2010.2059720
17. Syrjänen, T.: Lparse 1.0 user's manual. http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz
18. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. MIT Press (2005)
19. Thrun, S., Fox, D., Burgard, W., Dallaert, F.: Robust monte carlo localization for mobile robots. Artificial Intelligence **128**(1-2), 99–141 (2001)
20. Tseng, Y.C., Kuo, S.P., Lee, H.W., Huang, C.F.: Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. Information Processing in Sensor Networks **2634**, 554–554 (2003)
21. Verdone, R., Dardari, D., Mazzini, G., Conti, A.: Wireless Sensor and Actuator Networks: Technologies, Analysis and Design. Academic Press (2008)
22. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. **38** (2006). DOI http://doi.acm.org/10.1145/1177352.1177355. URL http://doi.acm.org/10.1145/1177352.1177355