Knowledge-Based Multi-Criteria Optimization to Support Indoor Positioning

Alessandra Mileo[1], Torsten Schaub[2], Davide Merico[1], and Roberto Bisiani[1]

[1] NOMADIS, University of Milan-Bicocca, Italy

[2] University of Potsdam, Germany

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 lacking 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 report upon a case-study and provide a first experimental evaluation of knowledge-based position estimation.

1 Introduction and Motivations

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 applying trilateration. The distance can be computed by measuring variations of signal intensity or by timing the latency from transmission to reception. The precision of positioning systems varies from a few meters in 2D to a few decimeters in 3D. Given the low cost of infrastructure and their flexibility, localization system based on Wireless Sensor Networks (WSN) are a good starting point for tackling the localization problem in home environments. However, when data are noisy and incomplete due to

Proceedings of the 17th International RCRA workshop (RCRA 2010):
Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion: Preliminary Report
Bologna, Italy, June 10–11, 2010

delays, breakdowns, errors in transmission, and alike, their lack of precision in position estimation needs to be supported and compensated.

Tracking is more 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.

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 [7, 12, 11]. Although many of these algorithm 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 [8]. Therefore, PFs are the most popular and commonly used methods for tackling the problem of tracking devising motion models to update the particle statuses. For this reason, we consider in our preliminary experimental evaluation results of PFs as representative of a wider calss of similar but less effective techniques.

Particle Filters discretize the probability distribution function of the position of the mobile object instead of controlling the whole position space. 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 commonsense knowledge for tracking, and
- easy customization and extensibility of existing solutions in a more flexible way whenever new technology or new sensor-data becomes 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 moreover it 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.

For validating our approach we collected a considerable amount of sensor data by appeal 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. 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. 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.

The simulation environment and properties of generated data are described in Section 2. Section 3 introduces the ASP formalism and investigates our modeling and reasoning strategies, while preliminary results are presented in Section 4. A short discussion follows in Section 5.

2 Data Simulation

To position an object or a device, the basic step is to use a reference point to determine the distance and angle between the device and the reference point itself. To do so we use the Receiver Signal Strength Indicator (RSSI), based on the degradation of radio signal while traveling in a space.

In addition to RSSI, we use Passive Infrared (PIR) and Range Finders (RF). PIR is used to catch the movements of a person (or 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.

In order to generate 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)[1].

Following a taxonomy of selected Agent-Based Modeling Tools illustrated in [9], we choose 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,

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

we are able to efficiently generate large data sets simulating days of the household activities of a person and corresponding sensor values.

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 the way 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 also 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 of graph 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 the 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 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); the division of the environment into rooms and areas is specified so that an area can be associated to one room and a room can contain different (disjoint[3]) areas;

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;

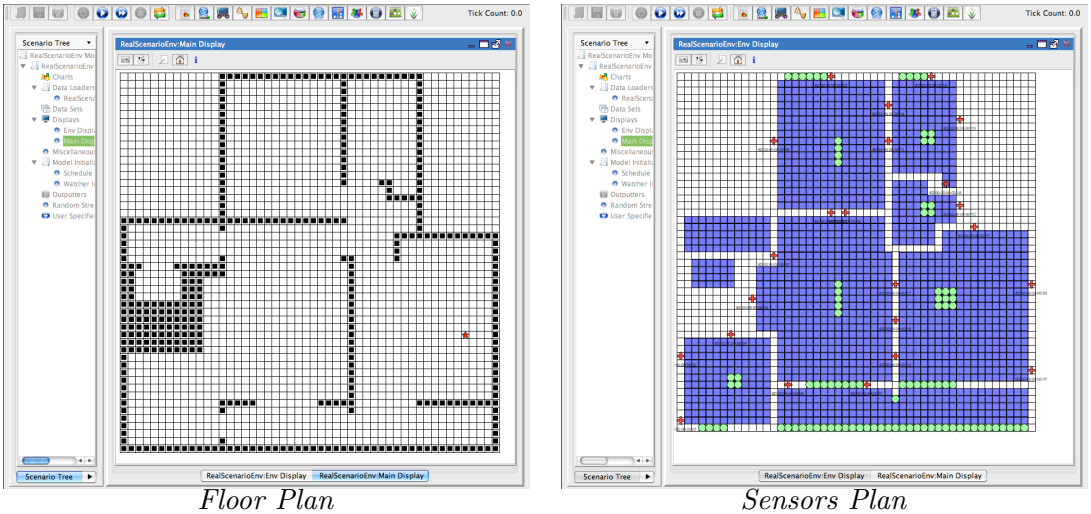
environmental sensors: during the data-generation phase, these sensors can read the temperature, humidity and brightness values;

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 [6, 13]; the position and the initial status of windows and light switches is also defined.

[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).

[3]The spatial inclusion of areas $A_1, \ldots, A_n$ in a room R is such that $\bigcup_{i=1}^{n} A_i \subset R$.

Figure 1: Repast S Displays for Simulation



Floor Plan *Sensors 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 according RSSI, PIR, and RF sensor data, (ii) simulate light, temperature, and humidity evolution during the day.

During every simulation tick, the *LocalizationSensor* agents compute the 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).

3 Supporting Localization and Tracking

All RSSI values produced in the simulation process are further processed using PF algorithms and they represent a starting point for the subsequent reasoning phase. Other (homogeneous) sensor values are aggregated with ad-hoc algorithms for features analysis and used to support position estimation. 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 simulation scenario and the generated data are first parsed in form of logic predicates and qualitative criteria are specified and combined to reason about localization and tracking.

3.1 ASP Basics

We assume the reader to be familiar with the terminology and basic definitions of ASP (see [1] for details). In what follows, we rely on the language supported by grounders *lparse* [10] and *gringo* [5], 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.

3.2 Modeling the Localization Scenario

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) is identified by coordinates X, Y. Each cell has some static properties that are mapped into a set of logic predicates as detailed in Table 1.

Locations are also associated to dynamic sensor information, according to available sensor data at each time step T. Such information can be derived by (i) processed RSSI values with associated percentage of likelihood P and (ii) PIR or RF signals. Logic predicates representing dynamic sensor data are listed in Table 2.

The available signals are treated differently in the definition of optimization criteria: processed RSSI signals are used as a starting point, that is, whenever a set of candidate positions is provided by the PF algorithm based on RSSI, PIR and RF are used to validate the most plausible position(s).

Predicate	Description
cell(X,Y,Room,Area)	association of a cell to an area of a room
passage(X,Y,R1,A1,R2,A2)	association of a cell to a passage
wall(X,Y)	description of a cell as part of a wall
loc(X,Y)	a location L is described by two coordinates
invalid(loc(X,Y))	specification of invalid cells
dataExpected(pir,loc(X,Y))	locations covered by PIR sensors
dataExpected(rf,loc(X,Y))	locations covered by RF sensors

Table 1: Logic Predicates representing static properties of cells

Predicate	Description
sensed(rssi,P,L,T)	processed RSSI signal with likelihood P received at time T for location L
sensed(pir,L,T)	PIR signal received at time T for location L
sensed(rf,L,T)	RF signal received at time T for location L

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

At any time step, a location becomes a candidate position when a processed RSSI, PIR or RF signal has been received for that location and the location is valid (e.g. it is not a wall). This allows us to immediately reduce the space of solutions to the positions for which at least one signal is available. The corresponding ASP code is as follows:

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

Besides the information about received sensor data at a given time step T for a location L, we also describe two additional dynamic properties for a localization and tracking problem: the *distance* measure between two consequent admissible moves and the *support* for a location as a candidate position at a give time step.

Distance between two locations $l_1 = loc(x_1, y_1)$ and $l_2 = loc(x_2, x_2)$ is represented by the propositional variable $dist(l_1, l_2, d)$ that is true for where

$$d = |x_1 - x_2| + |y_1 - y_2|$$

Support for a location L is the number of sensor data collected at location L at a given time T and it is represented by predicate $support(L, N, T)$ where

$$N = |\{sensed(S, L, T)\}| \; where \; S \in dataExpected(S, L)$$

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 filtered candidate positions obtained by PF by processing RSSI signals represent starting points to estimate the actual position. Our reasoning process is flexible enough to deal with noisy and even missing sensor values. Since we reduce the set of possible solutions to the ones for which results of PF are availabe, when this information is missing tracking the person on the grid becomes more difficult and the solution space can be huge, since the model of movement may produce a higher number of possibilities. This happens not only in case of malfunctioning sensors, but also when the PF cannot provide acceptable candidate positions for several sequential time steps. In this case, the qualitative analysis of the available information can still help, as detailed in the next subsection.

3.3 Multi-Criteria Optimization

The interpretation of heterogeneous sensor data used to support localization and tracking is based on the definition of properties of sensed information that can be meaningful according to available sensor data. In our solution, we identified three properties which combination can originate several optimization criteria.

Best PF property considers positions with the highest likelihood P, that are provided as a result of the PF algorithm; unfortunately it is not always true that the higher the likelihood obtained by the PF, the closer the candidate position is to the real one; for this reason we define additional criteria to validate positions; when results of PF are not available, a different reward function is used, which is closely related to the notion of *support* for a location L defined in the previous subsection: the higher the support for L, the better L is as a candidate position.

Best Coherence maximizes the value of the coherence property, which identifies a percentage indicating how many of the sensor signals expected from a given location L are effectively captured for that location at time T; this property is represented by predicate $coherence(L, C, T)$; if no additional sensor signals (excluded results of Particle Filter) are expected for a location L, its coherence C will always be 100% for all time step; in the same way, C is 0 at time T if we expect at least one additional sensor value for a location L and we do not have any of them; otherwise, we have that, for a location L,

$$C = (100 * M)/N$$

where $M = |\{dataExpected(S, L\}|$ and N is the support.

Best Move property identifies the best location at any time step T as the location L for which the distance between L and the location at time step $T-1$ is minimum; the associated property is represented by predicate $bestMove(L, T)$.

When new sensor information is introduced, we can easily re-define or extend the list of properties. 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*.

Once we have the list of properties, they can be combined into optimization criteria. The preference relation between two criteria is defined by their numbering such as C_i is preferred to C_j ($C_i >_{pref} C_j$) when $i > j$.

In our prototype, we identified four criteria C_i, $i = 1..4$, combining properties on sensor information as follows:

C_1 is satisfied for a location L at time T when L satisfies both the best move and the best coherence property;

C_2 is satisfied for a location L at time T when L verifies the best coherence property;

C_3 is satisfied for a location L at time T when L verifies the best move property;

C_4 is satisfied for a location L at time T when at least one of the predicates in Table 2 is true on L at time T, and L has the highest likelyhood or the highest support; this is the weakest criterion and it is applied only when no better solution can be found by applying the other criteria.

This ordering is motivated by the fact that we want to obtain the solution satisfying all criteria as the best option, but if it does not exist, we prefer to give up the movement model and consider coherence as more reliable[4]. 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.

In order to implement optimization on ordered criteria, we introduce the concept of *best value* B for a criterion C_i at time T, $B_{C_i}(T)$. The best value is determined in two different ways according to the fact that (a) we have L as a candidate position at time T as a result of Particle Filter or (b) no candidate position is produced by Particle Filter at time T but we have other sensed data supporting a location L as a candidate position. The definition is as follows:

a) $B_{C_i}(T) = max\{P : sensed(rssi, P, L, T), \nexists\ B_{C_j}(T)\ with\ C_j < C_i\}$

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

Table 3: Preliminary results: accuracy and fault tolerance of the ASP approach.

Size (time)	Missing data	MAE (ASP) PF, PIR, RF	MAE (Particle) RSSI Only
30	0%	5.55	6.17
30	5%	5.53	-
30	15%	5.09	-
50	0%	5.88	4.41
50	5%	5.69	-
50	15%	5.68	-
90	0%	5.00	4.87
90	5%	5.04	-
90	15%	5.02	-
180	0%	3.14	3.34
180	5%	3.11	-
180	15%	3.64	-

b) $B_{C_i}(T) = max\{N : support(L, T, N), \nexists\ B_{C_j}(T)\ with\ C_j < C_i\}$

The best location L represented by predicate $best\_location(L, T)$ can now be identified by selecting the best value returned by the application of the most preferred criterion. The actual implementation of this optimization technique is evaluated using the *Clasp* solver [4] on grounded logic programs obtained by using *Gringo* [4] as a grounder.

Our specification of the problem needs a way to generate the space of solutions and then select the best one via optimization. This is done by applying the generate and test approach: a cardinality constraint specifies that for each time step T, exactly one location is selected (generating phase) and an integrity constraint specifies that a selected location L for a time step T has to be the best location for T (testing phase) as follows:

```
1 { at(L,T) : location(L) } 1 :- time(T).
:- at(L,T), not best_location(L,T).
```

4 Preliminary Experiments

In this section, we provide a first experimental evaluation of our knowledge-based approach to position estimation.

Our experiments are based on the generation of several test instances of different size. Each instance has been generated 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.

We focus our attention on the following aspects of our approach:

- the overall *accuracy* of results (Mean Absolute Error);
- the *fault tolerance* expressed as the impact of missing data on the accuracy;
- the *effectiveness* of optimization criteria in filtering results.

We determine the overall accuracy of our results via the Mean Absolute Error (MAE), proposed in [2] 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:

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

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. Note that in our simulation environment, the dimension of each cell is 20cm, which means that an MAE of 5 indicates an average error of one meter which is fully acceptable.

We compare the MAE indexes obtained through Particle Filters (PF) based on the RSSI signals only, with the MAE index obtained by ASP combining results of PF based on RSSI signals with Range Finder (RF) and Passive Infrared (PIR) measurements.

If we observe the first row of each instance (with no missing data), we see that the introduction of additional (noisy) sensor data gives us an MAE that is slightly higher with the ASP-enhanced approach or very close (as in the last instance). The only real exception is the MAE of the smallest instance of size 30. 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, MAE index slightly decreases as size rises: on bigger instances, particles tends to converge on the more plausible positions, resulting in higher accuracy. This increased accuracy propagates on ASP results, since the ASP approach itself uses PF results as a starting point, and it is not furher affected by the size of the instance in terms of MAE.

Another aspect we took into account is 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 robust and able to provide candidate solutions even in case of malfunctioning sensors or weak PF results.

The second column in Table 3 indicates the amount of sensor data we removed from the input given to ASP. We observe that in general we obtain lower MAE values by removing data, probably due to the fact that our knowledge-based approach makes it possible to estimate a position by using the model of movement or additional domain-related knowledge to reduce the search space and propose plausible estimations even when the PF algorithm is not able to provide some. Another consideration is that removing data could also remove noise, thus giving better global results.

If we consider the last instance in Table 3, we notice that removing 5% of PF data gives us an initial reduction of MAE, which then increases when we get to removing 15% of PF 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.

Although we need more tests to validate our intuitions, the main idea is that if we have a higher number of sensors, 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 show that this approach makes accuracy robust against incomplete information, but we need more accurate tests to support this statement. 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, optimization criteria can help selecting

the best candidate positions in the search space.

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 C_2 and C_3 giving higher preference to the movement model than the coherence property, the MAE increases.

In addition, our approach makes it 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.

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

5 Discussion

We presented a hybrid approach to indoor position estimation by combining and extending quasi-standard quantitative methods in a knowledge-based yet qualitative 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 preliminary experiments. Although we have not discussed the encoding in detail, it 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.

This preliminary work is only a starting point. Future work will have to address more extensive and systematic experiments in various simulated as well as real scenarios. More sensor and semantic information needs to be taken into account in order to evaluate the true scalability of our approach. Finally, our current implementation accumulates data in hourly intervals. We plan to refine this to use the incremental ASP solver *iClingo* [3] in order to provide a much more fine-grained approach to localization and tracking, eventually aiming at real-time conditions.

References

[1] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[2] M. Broxton, J. Lifton, and J. A. Paradiso. Localization on the pushpin computing sensor network using spectral graph drawing and mesh relaxation. *SIGMOBILE Mobile Computer and Communication Review*, 10, 2006.

[3] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele. Engineering an incremental ASP solver. In M. Garcia de la Banda and E. Pontelli, editors, *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, volume 5366, pages 190–205, 2008.

[4] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In *Ninth International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 260–265. Springer-Verlag, 2007.

[5] M. Gebser, T. Schaub, and S. Thiele. Gringo : A new grounder for answer set programming. In *LPNMR*, pages 266–271, 2007.

[6] Hedetniemi, Hedetniemi, and Liestman. A survey of gossiping and broadcasting in communication networks. *NETWORKS: Networks: An International Journal*, 18, 1988.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[8] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3):9, 2007.

[9] M. J. North and C. M. Macal. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. Oxford University Press, Inc., New York, NY, USA, 2007.

[10] T. Syrjänen. Lparse 1.0 user's manual. http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz.

[11] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.

[12] S. Thrun, D. Fox, W. Burgard, and F. Dallaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[13] R. Verdone, D. Dardari, G. Mazzini, and A. Conti. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design.* Academic Press, 2008.