Smart Blueprints: Automatically Generated Maps of Homes and the Devices Within Them

Jiakang Lu and Kamin Whitehouse

Department of Computer Science, University of Virginia
Charlottesville, VA, USA
{jklu,whitehouse}@cs.virginia.edu

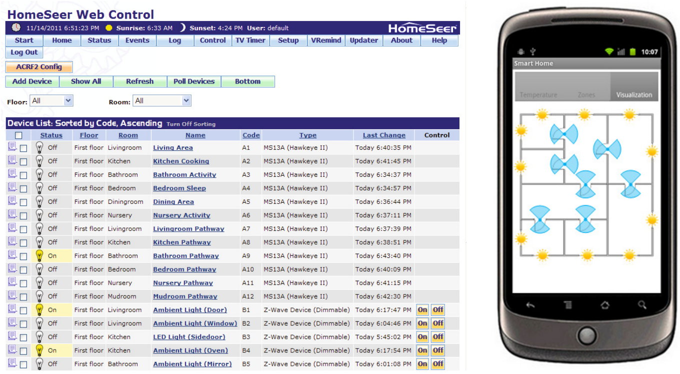
Abstract. Off-the-shelf home automation technology is making it easier than ever for people to convert their own homes into "smart homes". However, manual configuration is tedious and error-prone. In this paper, we present a system that automatically generates a map of the home and the devices within it. It requires no specialized deployment tools, 3D scanners, or localization hardware, and infers the floor plan directly from the smart home sensors themselves, e.g. light and motion sensors. The system can be used to automatically configure home automation systems or to automatically produce an intuitive map-like interface for visualizing sensor data and interacting with controllers. We call this system *Smart Blueprints* because it is automatically customized to the unique configuration of each home. We evaluate this system by deploying in four different houses. Our results indicate that, for three out of the four home deployments, our system can automatically narrow the layout down to 2-4 candidates per home using only one week of collected data.

1 Introduction

Commercial off-the-shelf home automation devices are becoming smaller, more reliable, and less expensive. Furthermore, low-power wireless technology allows these devices to be surface-mounted, which obviates the need for home re-wiring and dramatically reduces installation costs. However, creating a do-it-yourself smart home involves much more than just installing hardware; the application software must also be configured with the *physical context* of each device, including its location within the home and its spatial relationship to other devices. For example, a lighting automation system must know which lights and motion sensors are in the same room in order to turn the lights on when the room is occupied. Home automation devices today, however, are unaware of their context within the home. For example, Figure 1(a) shows a typical home automation tool: it automatically detects all devices in the home network and displays them to the user as a list. The user must note the serial number on each sensor and mentally map each physical device to its virtual representation in software. Then, the physical relationship between devices must be encoded in software as abstract logical rules, e.g. *"Lights #19 and #28 should be controlled by light sensor #5 and motion sensor #15.* This manual configuration process is tedious and error-prone.

In this paper, we present an approach to automatically infer the spatial layout of smart home deployments, including (i) the floor plan of the house and (ii) the location

J. Kay et al. (Eds.): Pervasive 2012, LNCS 7319, pp. 125–142, 2012.
© Springer-Verlag Berlin Heidelberg 2012



(a) HomeSeer (b) Smart Blueprints

Fig. 1. A conventional user interface for home automation software shows all available devices in a list, while Smart Blueprints automatically infer the home's floor plan and shows the sensors in their physical context

of the devices within that floor plan. Our approach requires no specialized deployment tools, 3D scanners, or localization hardware; the floor plan is inferred from the smart home sensors themselves, e.g. light and motion sensors. A homeowner can simply snap sensors into place, and the sensors can provide useful functionality with little or no additional configuration effort. For example, home automation systems can automatically configure space heaters to respond to the temperature and motion sensors found to be in the same room. The system can also produce an intuitive map-like interface for visualizing sensor data or interacting with controllers. Figure 1(b) shows an example of this interface, which is more intuitive than the conventional list-based interface in Figure 1(a). We call this system a *Smart Blueprint* because it is automatically customized to the unique layout of each home and that home's devices.

As a case study, this paper considers the sensors that are required for a home energy management system: light sensors on windows, temperature sensors throughout the house, and motion sensors in each room. Armed with only these sensors, the Smart Blueprint system uses a three-step process to infer the floor plan and sensor locations. (1) First, it analyzes sensor data to infer how many rooms are in the house and which devices are in each room, using the insight that motion sensors in the same room tend to be triggered at similar times due to occupant movement. (2) Once the rooms are identified, the system analyzes motion sensor data to infer which rooms are adjacent, and it analyzes light patterns to decide which side of house the windows in the room are more likely to be on, e.g. East or West. This analysis constrains the location of each room with respect to the other rooms. (3) Finally, the system pieces the rooms together like a puzzle to find the floor plan that best explains the sensor data. If multiple floor plans explain the data equally well, the system asks the user to choose between a small number of alternatives.

The sensor data alone is not sufficient to fully constrain the floor plan and sensor layout, so we *pair* the devices in strategic ways to create additional constraints. First, we deploy motion sensors in pairs by packaging them into a single physical enclosure that snaps in place behind the door jamb of a doorway with the sensors facing in opposite directions, thereby creating a new constraint that the two sensors must be in different rooms. This constraint helps to separate clusters of sensors that show overlapping activity patterns, such as motion sensors with visibility into neighboring rooms, or sensors in different rooms that are often used at the same time by different people. Next, we pair light sensors with motion sensors so that the light sensors can be associated with a particular room, based on activity patterns. Finally, motion sensors in doorways are paired with magnetometers in order to infer the doorway's cardinal direction.

Our basic approach is to infer constraints on spatial layout based on patterns in the sensor data, strategically using sensor pairing and supplemental sensors as necessary to fully constrain the floor plan and sensor layout. Smart Blueprints do not require any specific set of sensors to operate and the sensors described in this paper are only an example; the general approach that can be applied to any smart home or home automation system. For example, many devices such as wireless light switches and electronic appliances will also have usage patterns that correlate with occupant movement and activity. Furthermore, any device can be paired with a motion sensor to constrain its room location, a magnetometer to constrain its orientation, or a light sensor to constrain its side of the house. Additional constraints are also possible by pairing with other sensors, or applying other signal processing algorithms. Pairing sensors to create new constraints adds only marginal hardware cost and, as more constraints are added, the floor plan and sensor layout become easier to infer.

To evaluate this approach, we deployed Smart Blueprints in four houses for 1-3 months each. For three of the houses, the system narrows the system layout down to 2-4 candidates after processing only one week of data. Most users should be able to select their own floor plan out of only 2-4 choices. On the fourth house, however, the system did not identify the correct topology because the house had multiple floors and a very modern, open floor plan. We discuss these and other limitations of our system, as well as future techniques to improve it. We also present the performance of our system using simulated data based on 15 additional floor plans that were downloaded from architecture sites on the Internet. Aside from the Smart Blueprints system itself, the contributions of this work include insights into the computational structure of homes and how sensors can be used to generate and infer topological constraints.

2 Background and Related Work

Many existing technologies can automatically map out the floor plan of a building or localize devices in a building, but to our knowledge the Smart Blueprints system is the first to do both simultaneously without the need for user intervention or specialized deployment tools. For example, optical [1], laser, acoustic, and RF [2] sensors have all been used to find the boundaries of a room, and can be combined with mobile robots [3] or a mobile human [4] to create a complete map of a building. Approximate floor plans can also be generated by tracking personnel movement with in-building tracking systems [5]. However, Smart Blueprints are easier to use because no specialized tools, robots, or in-building tracking systems are required. Furthermore, the notion of using an installation tool or process does not always apply to do-it-yourself deployments that may occur piecemeal over the course of many months or years, as the homeowner adds new functionality. In contrast, the Smart Blueprints system can automatically assimilate new devices into the existing map at any time, with no additional overhead.

An installer could manually configure a smart home deployment using a graphical interface to quickly draw the house floor plan and locate the sensors within that plan. Several such tools already exist that offer visual drag-and-drop interfaces, and some are even designed for cell phone usage to facilitate on-site sketches[1]. However, this approach is no less tedious or error-prone than manually configuring a conventional list of devices: users must still manually create a mapping between the physical world and its virtual representation. Similarly, users could provide basic information about the home such as the number of rooms or the directions of the walls, as initialization information for a Smart Blueprint. We do not preclude this possibility and believe that it will in fact improve results, but it will also introduce new challenges about the confidence of user-supplied information. For example, does a foyer, stairway landing, or hallway count as a room? How many rooms is a great room (a living room, dining room, and kitchen that are not separated by walls)? Similar types of ambiguity exist for the number of floors and the direction of windows or walls. We informally asked several people to indicate how many rooms and floors are in our evaluation homes and received different answers from each. In this paper, we aim to demonstrate that the Smart Blueprints approach is possible even in the absence of any user input.

Smart homes have been an active area of research for several decades, typically involving long-term and highly-engineered sensor installations [6,7]. In contrast, the goal of our work is not to enable new smart home applications, but to facilitate the deployment process for do-it-yourself installations. Several prior projects have addressed configuration challenges for home and building automation systems, but focus on interoperability [8,9], automatic service discovery [10], or the use of modularity [11] and ontologies for seamless integration [12]. All of these solutions are necessary for a zero-configuration smart home system. In this paper, we focus on a different aspect of the configuration issue: identifying the physical context of devices.

A large body of literature has developed formalisms to specify a building floor plans as a constraint satisfaction problem (CSP) so that an architect can review all possible floor plans that satisfy certain physical constraints (e.g. types of rooms, minimum/maximum size, etc.) [13]. The Smart Blueprints system also uses constraints and search algorithms, but generally has much more specific constraints (e.g. doorway #3 has a north-to-south orientation) because it has a different goal: to find a small number of constraints that uniquely specify a single floor plan. Our contribution is not the ability to specify or search through floor plans, but the ability to automatically infer constraints on those floor plans from sensor data.

[1] http://www.floorplanner.com/; http://www.smartdraw.com/

(a) Doorway Sensor (b) Light Sensor

Fig. 2. We packaged multiple sensors together to detect *structural relationships* between data streams, thereby constraining the floor plan

3 A Case Study: Home Energy Management

The Smart Blueprints system presented in this paper is part of a broader project that aims to reduce the two largest energy consumers in the home: 1) lighting, and 2) heating and cooling. The system performs two types of controls. First, it uses a zoned residential heating and cooling system to control the temperature in each room individually, based on activity detected by motion sensors: when the room is occupied, it is heated or cooled to a comfortable setpoint temperature and, when it is not occupied, the temperature is allowed to float to a more energy-efficient setback temperature. Second, the system uses controllable windows shades to adjust the amount of light and heat admitted into the room. This control component builds on the previous work of the authors in daylight harvesting technology [14], and uses both light and motion sensors: when a room is occupied, the system prioritizes indoor lighting comfort and, when it is not occupied, the system prioritizes solar gain through the window to optimize heating and cooling efficiencies. In order to execute these two control strategies, the system needs at least three types of sensors: a motion sensor in each room, a light sensor on every controllable window, and a temperature sensor in each room. We demonstrate how the Smart Blueprints approach can be applied to this home energy management system by modifying the design and deployment of these sensors. However, the general approach can be applied to other home sensing systems as well.

4 Designing Sensors to Infer Context

The first implementation of our zoning and daylight harvesting systems used conventional motion, temperature, and light sensors that were manually configured with room location and other contextual information. In this paper, however, we take a novel two-pronged approach to sensor design that enables contextual information about a sensor to be inferred. First, we always *pair* multiple sensors together in the same physical enclosure. In this way, the values measured by one sensor serve as contextual information for the others, and vice versa. The pairings define a fixed spatial relationship between what would otherwise be independent data streams. By considering all of these relationships at once, we can then piece together both the local context of each sensor and the greater

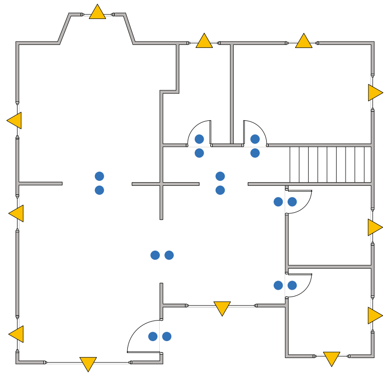
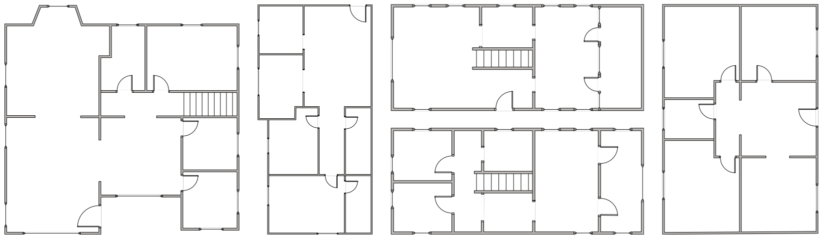


Fig. 3. Two paired motion sensors were installed at each doorway, and a light/motion pair were installed on each window

context of the entire system, i.e. the floor plan. Second, we designed our sensors to be placed at only certain types of moldings and fixtures, specifically in doorways and on windows. The sensors cannot be mounted on any flat surface the way a conventional motion, light, or temperature sensor could be, which add additional constraints on the locations of these sensors.

The doorway sensors are designed to snap in place behind the door jamb for easy installation and architectural camouflage (Figure 2(a)). The device contains two motion sensors, one pointing into each of the neighboring rooms. This physical pairing of motion sensors creates a relationship between the activities in every pair of adjacent rooms. The same enclosure contains a door latch sensor to detect whether the door is open or closed, as shown in the left of Figure 2(a) and can also contain temperature and humidity sensors, and a magnetometer to measure the orientation of the doorway with respect to magnetic north. In our pilot deployment, we built the doorway sensors using Parallax passive infrared (PIR) motion sensors, a standard magnetic reed switch, and the Synapse SNAPpy wireless microcontrollers. Further details about the deployment, power, and wireless properties of the doorway sensors can be found in another paper [15]. Instead of building magnetometers into the enclosure, we used the compass on a smartphone to measure the angle of each doorway.

The window sensors measure both light levels and solar gain caused by each window. The light sensors are placed on the window facing outside and measure the incoming daylight on the vertical glass surface. In our system, we paired these light sensors with motion sensors pointing back into the room. This pairing creates a defined relationship between the light values and the activity in the room that contains that window. The hardware costs only pennies for both light sensors and motion sensors, so this approach should not add substantial financial burden. The pairing can be enforced by putting the two sensors into a single plastic enclosure, although in our pilot deployment we simply deployed two sensors in tandem (Figure 2(b)). We used the X10 passive infrared motion sensors and the U012-12 data loggers designed by Onset, which monitor the environment with the built-in light, temperature, and humidity sensors.



(a) House A (b) House B (c) House C (d) House D

Fig. 4. To evaluate our system, we deployed in four homes with different floor plans

Table 1. We deployed our system in four multi-person homes with a widely varying set of windows, doors, rooms, and floors

House ID	#Residents	#Floors	#Rooms	#Doors	#Windows	Window Orientations	#Months
A	3	1	8	8	12	4	3
B	2	1	7	7	5	2	2
C	2	2	9	15	23	4	1
D	3	1	7	7	4	2	1

We deployed these two types of sensors in four homes of various sizes, including both single-story and multi-story houses, and the durations of the sensor deployments vary from one to three months. We deployed one doorway sensor at the top of doorways that cover all the inner doors, all the inner hallways and all the entryways to the home. We deployed a window sensor on at least one window on each wall of every room. Two homes were detached homes that had windows in all four directions, and two homes were apartments or condominiums that had windows on only two walls. An example of home deployment is shown in Figure 3. The people living in the homes included students, professionals, and homemakers. Scaled diagrams of the actual house floor plans are shown in Figure 4, and the deployment information about these homes is summarized in Table 1.

5 Inferring Rooms and Adjacency

The first step to identify the home's floor plan is to infer the number of rooms in the house, and which rooms are adjacent. For this task, we use the insight that objects in the same room are often used at similar times, and so the number of rooms can be inferred based on the number of clusters of sensors that are temporally correlated. In our pilot study, we use motion sensor data to infer the number of rooms: when people are in a room, multiple sensors in the room should detect motion. This approach faces two challenges. First, motion sensors have visibility through doorways and into multiple rooms. In our deployments, some of the motion sensors could detect activity in up to five rooms. Second, houses with multiple people often have simultaneous activity in more than one room at a time. For these two reasons, the temporal correlation between sensors is a weak indicator of room co-location. In fact, motion sensors in different rooms often had higher correlation than sensors in the same rooms, particularly when they both had high visibility into another highly-frequented room. In prior work, we found that a simple clustering approach was sufficient to identify which devices were in the same room, but this approach assumed only a single motion sensor per room, and could not accurately determine the exact number of rooms [16]. In this paper, we need the exact number of rooms, and we quickly found that a naive clustering approach did not work for any of the four pilot deployments, particularly with more than one motion sensor per room.

To address this problem, we leverage the motion sensor *pairings* on our doorway sensors: any two motion sensors within the same enclosure are guaranteed to be monitoring different rooms. This constraint allows us to ensure that sensor clusterings do not cross room boundaries: any sensor clustering that does is eliminated from consideration. The basic algorithm has five steps. (1) Identify all *simultaneously firing sensors*: those sensors that fire within one second of each other. We call these *firing sets* $S = s_1, s_2, ..., s_K$, where K is the size of the firing set, and $s_i : i < K$ are the sensors in the set. (2) Eliminate any firing set S that contains two paired sensors: if S contains two paired sensors, it might have been caused by multiple people in different parts of the house, or a single person making a doorway crossing. By eliminating firing sets with paired sensors, we increase the likelihood that it was caused by a single person who is safely on the interior of a single room. (3) Calculate the frequency f_S of all remaining firing sets S. (4) Calculate a weight w_S for each firing set S, defined as follows:

$$w_S = \frac{f_S}{\sum_{k=1}^{K} f_{s_k}} \times \frac{f_S}{min_{k=1}^{K} f_{s_k}} \tag{1}$$

The frequency f_S indicates the number of times that a set of sensors fire together. However, this does not necessarily indicate the importance of the set, because some rooms are used more often than others. The weight w_S is designed to normalize the frequency based on the upper bound on the number of times that this set of sensors could possibly have fired together. This upper bound can have two cases. First, the total number of times that any of the sensors fired: $\sum_{k=1}^{K} f_{s_k}$. Second, the number of times that the least frequent sensor fired: $min_{k=1}^{K} f_{s_k}$. We normalize f_S by both of these values to a create a weight w_S that approximates the likelihood that the sensors in the firing set are actually in the same room.

(5) The last step is to merge firing sets into clusters that represent rooms. To initialize, a single cluster is defined as the firing set with the highest weight. We then process the remaining firing sets in a greedy fashion sorted in descending order by weight. Each firing set is merged with a cluster if and only if (a) the merge would not cause the cluster to contain any paired sensors, and (b) the merge would not cause a loop among doorways, which implies any transitive closure of doorways would not contain any paired sensors. Otherwise, the firing set is used to initiate a new cluster. When all firing sets have been processed, the clusters represent individual rooms. Two rooms are defined to be adjacent if the union of their clusters contains two paired sensors.

We apply the technique on our four house deployments, and the percentage of correct room adjacency links detected over 21 days is shown in Figure 5 for Houses A, B, and D. The results show that accuracy fluctuates during the first week but stabilizes

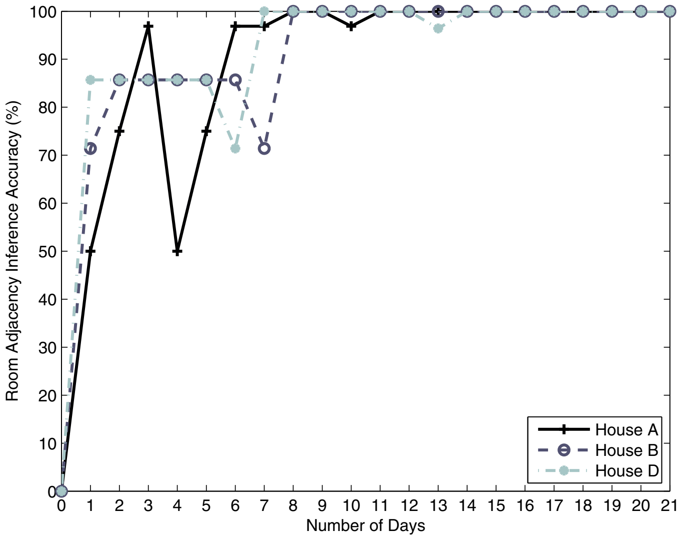
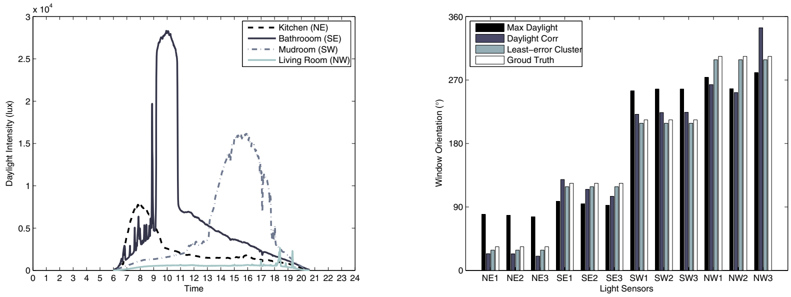


Fig. 5. In homes A, B, and D, the system infers the correct number of rooms and the adjacency of those rooms after collecting motion sensor data for 7-14 days

to the correct number of rooms and room adjacency once it obtains enough data. This means that, for these houses, the system can provide the correct floor plan after a learning period of 1-2 weeks. The sensor clustering for these three houses is illustrated in Figures 7(a), 7(g), and 7(m). Our current algorithm does not work for House C because it has two floors, and the algorithm combines rooms from both floors into a single room. We discuss ways to address this limitation as future work.

6 Inferring Structural Constraints

The number of rooms and their adjacency does not create a fully-specified floor plan, and other structural constraints must also be measured or inferred through other means. In our case study, the application needs a light sensor on every window to manage lighting levels and solar gain. We use these sensors to infer the angle of the windows with respect to true north, based on the patterns of sunlight that are detected throughout the day. In previous work, we noted that light sensors clearly differentiate direct and indirect sunlight, and that direct sunlight occurs at different times of day based on the angle of a window. Therefore, the time of day of the maximum light reading can be used to infer the angle of a window to within a few degrees, when exposed to direct sunlight [17]. Figure 6(a) shows ideal cases in House A for light measurements on windows at different angles. In a home setting, however, the direct sunlight can be consistently blocked by trees or neighboring houses, skewing the angle measurements. We found this to skew some angle estimates enough that windows appeared to be on the wrong wall.



(a) Light Patterns at Different Windows (b) Window Orientation Accuracy

Fig. 6. Light sensors are needed to manage daylight harvesting and solar gain. We can *also* infer the angle of orientation for each window using patterns in the sunlight data. Our algorithm accurately inferred the angle of all twelve instrumented windows in House A.

In order to overcome this challenge, we use two new techniques. First, we do not simply look at the maximum light reading. Instead, we calculate the expected curve for all possible window orientations, and assign each window the orientation with which its light readings have the highest correlation coefficient. This reduces the impact of partial shade that causes short-term abnormalities in the daily light response curves. Second, we assume that houses are built primarily with right angles, and so the windows point in only four directions. We first use all windows to estimate the angle of orientation of the entire house, and subtract this from the estimated window angles. Then, we cluster the window orientations so as to minimize the difference between the cluster average and the four cardinal directions.

We apply these techniques to light sensors deployed in House A, and the results are shown in Figure 6(b). The naive technique described above (*Max Daylight*) produced 37.1 degrees of error on average, and did not clearly differentiate the SW and NW windows. Using correlation coefficients (*Daylight Corr*) approach improved the absolute error, but did not improve differentiation. Finally, our clustering approach (*Least-error Cluster*) correctly assigned each window to the correct wall, by looking at all windows in aggregate instead of each window individually.

These experiments analyze light sensors on windows, but similar approaches could also be used on the interior of a room: the highest light levels in a room will often be correlated with the side of the house that the room is on. Otherwise, devices in the home can alternatively be equipped with magnetometers. Magnetometers are commonly used as in cell phones, GPS receivers and vehicles to measure orientation with respect to magnetic north. They are small, inexpensive, and low-power devices that can easily be integrated into a sensor package such as our doorway enclosure. In our pilot deployment, we used the magnetometer in an Android cell phone to measure the doorway angle by holding the device up to the wooden doorway trim. This process caused some noise in the angle measurements because the trim is not perfectly straight or the phone is not exactly level. Table 2 lists the measurements of the doorways in House A, which are accurate to within 10-15 degrees. We would expect the same causes of noise to affect values measured by a magnetometer actually embedded into a surface mounted enclosure.

7 Searching the Space of Floor Plans

The goal of our search algorithm is to generate a set of floor plans that are most consistent with the structural constraints that were inferred from our sensors. The input to the search algorithm is a number of rooms R, a $R \times R$ adjacency matrix that indicates the adjacency of rooms, and the angle of orientation for doors and/or windows. The output of the algorithm is a fully-specified topology with (x, y) coordinates for each room.

7.1 Defining the Search Space

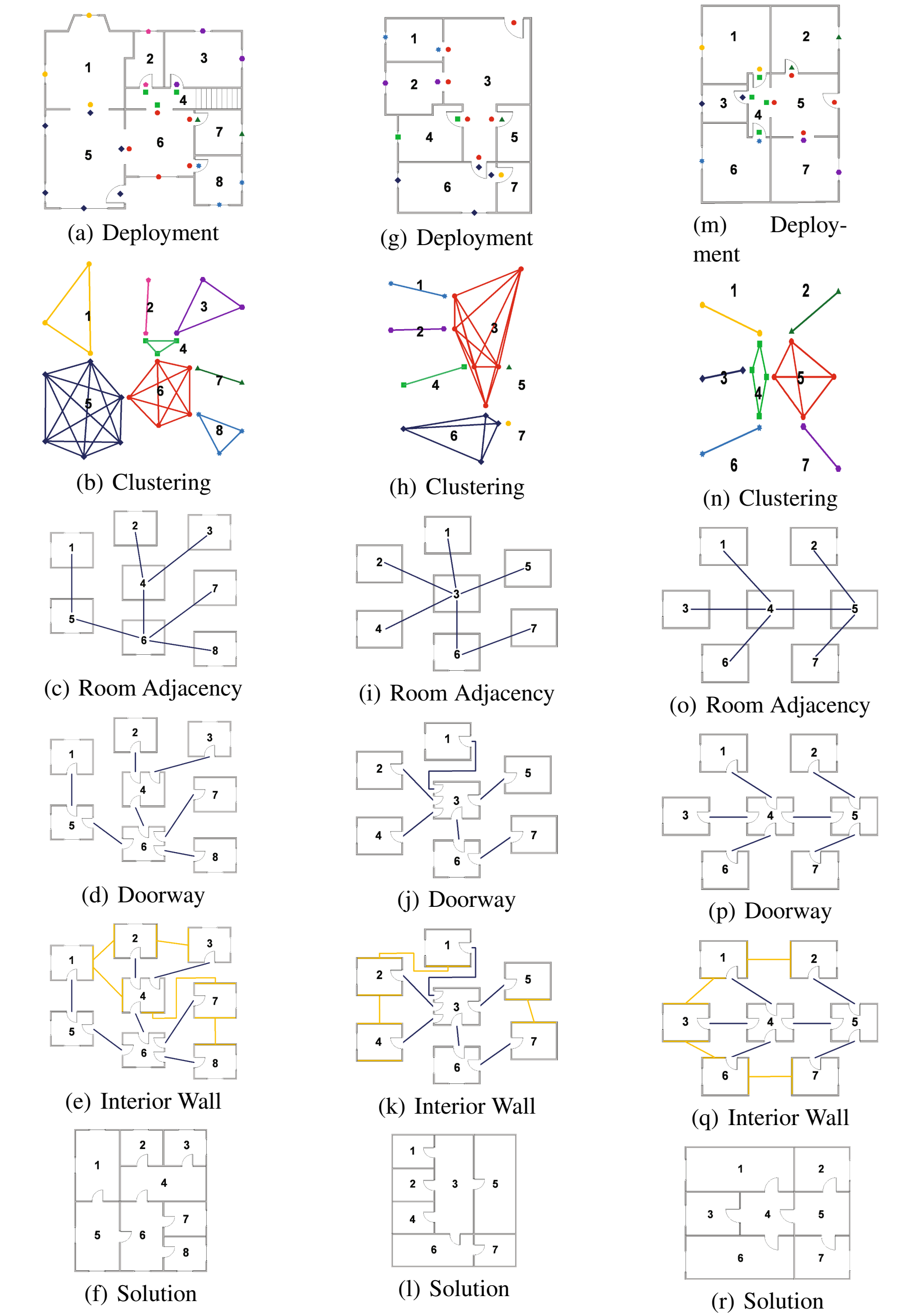
Table 2. Clustering magnetometers reduces orientation error to within $\sim 10^\circ$

Doorway	Measured
Dining Room	322
Kitchen	230
Hallway	221
Bathroom	221
Bedroom	225
Mudroom	324
Nursery	327

First, we define the search space. We illustrate each stage of this search space using Figure 7. In this section, we only discuss Houses A, B and D, where our system is capable of inferring the correct room adjacency. The root of the search tree for each house is the adjacency matrix, which is derived from the sensor clustering performed in a previous step, as illustrated in Figures 7(c), 7(i), and 7(o). The second level of the search tree is the set of all possible doorway assignments, each of which defines a different set of directions for each doorway. Figures 7(d), 7(j), and 7(p) show the correct doorway assignments for each house. For every pair of adjacent rooms, there are four possible orientations for each doorway (North, South, East, and West). Thus, the total number of doorway assignments is 4^D, where D is the number of doorways. For each doorway assignment, the search algorithm explores all possible interior wall assignments. These assignments define whether any pair of rooms is adjacent by an interior wall (but not a doorway). Figures 7(e), 7(k), and 7(q) show the correct doorway assignments for each house. Any pair of two rooms that are not connected by a doorway can possibly be adjacent through an interior wall, and the total number of such room pairs is $n_{pairs} = \frac{R*(R-1)}{2} - D$. Finally, given a set of doorway wall assignments and interior wall assignments, we generate the size of each wall by formulating a linear program that minimizes the length L_i of each wall i subject to the constraints that $L_i \geq 1$ and $L_i = \sum_j L_j : adjacent(L_i, L_j)$. Once the wall sizes are defined, the algorithm tiles the rooms together in sequential fashion to generate the (x, y) coordinates for each room.

7.2 Efficiently Searching the Space

The size of this search space is different for each house, but in general it is too large to search exhaustively. The total number of possible topologies is $4^D * \left\{\sum_{i=1}^{n_{pairs}} 4^n * \binom{n_{pairs}}{i}\right\}$, where D is the number of doorways, n_{pairs} is the number of room pairs that are connected via interior walls, and the number of 4 indicates the four possible directions in which two room can be connected. The maximum value of n_{pairs} is $\frac{R*(R-1)}{2} - D$. The second row in Table 3 titled `total space size` shows the total size of the search space for each of the three houses discussed.



(a) Deployment (g) Deployment (m) Deployment
(b) Clustering (h) Clustering (n) Clustering
(c) Room Adjacency (i) Room Adjacency (o) Room Adjacency
(d) Doorway (j) Doorway (p) Doorway
(e) Interior Wall (k) Interior Wall (q) Interior Wall
(f) Solution (l) Solution (r) Solution

Fig. 7. This figure shows each step of our algorithm (in each row) for all three houses: (1) sensors are clustered together based on similar activity patterns (2) these clusters are converted into rooms and adjacency between rooms (3) doorways are assigned to specific walls (N,S,E,W) (4) interior walls for room pairs are defined to be adjacent (5) spatial layout of the floor plan

In order to more efficiently search this state space, we use three different pruning techniques. First, if the window and/or door orientations are known, we do not explore topologies that are inconsistent with these orientations. In other words, if a doorway is known to be North-South oriented, we do not explore topologies that put the doorway on an Eastern or Western wall of a room. Row 3 in Table 3 shows the size of the search space after constraints on window and doorway orientations are applied. Doorway constraints reduce the search space by two orders of magnitude, window constraints reduce it by five orders of magnitude, and applying both simultaneously reduces it by seven orders of magnitude.

The second and third pruning techniques rely on a non-monotonically increasing score function that we use to assign each node a value. This score function is discussed in the next section. Based on this score function, we use traditional $\alpha - \beta$ pruning: once a leaf node with a low score is found, the search algorithm no longer searches sub-trees where the root of the sub-tree has a higher score. We also use the score function for early termination: when exploring the bottom level of the tree, if the algorithm detects that adding additional interior walls always increases the score of a node, it terminates and backtracks. This pruning technique is based on the observation that the sub-tree is already over constrained. Row 4 indicate that these pruning techniques reduce the number of explored nodes by 7-15 orders of magnitude from the size of the entire search space, dramatically improving search efficiency. Row 4 shows both the number of doorway assignments explored and the number of interior wall assignments explored.

7.3 Choosing the Best Topologies

The pruning techniques described above reduce the number of explored nodes down to the millions or billions. This is computationally tractable, but not acceptable in terms of search accuracy. To be successful, we need to leverage the structural constraints to choose the best out of millions or billions of alternative feasible topologies. To do this, we start with two simple heuristics:

– Minimize the number of doors and neighboring rooms that share a wall with a window.
– Minimize the number of corners of the building.

The first heuristic is based on the observation that most windows are on exterior walls, and it is rare for a room to have a window and an interior door on the same wall, or a window and another adjacent room on the same wall. The second heuristic is based on the observation that most buildings are square, and all squares have four corners. Some houses do have extensions, sheds, or other outcroppings, all of which lead to additional corners. However, the number of corners is typically minimized for cost, real estate efficiency, and heating/cooling efficiency. Therefore, the score of a floor plan increases by one for each corner that it has beyond four. Rows 5 and 6 of Table 3 show that these two cost functions eliminate all but thousands or even tens of floor plan candidates.

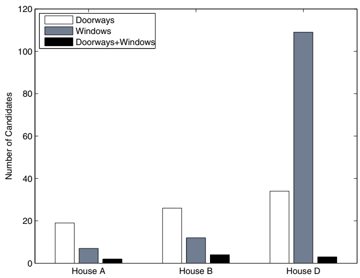
Of the remaining candidate floor plans, most can be eliminated because they are self-inconsistent. These inconsistent floor plans are akin to Mobius strips or Escher stairways: due to computational complexity, our search algorithm generates all possible

Table 3. The total size of the floor plan search space is very large (row 2). This space can be reduced using structural information (row 3), cost-based pruning (row 4), score functions (rows 5-6), and filter functions (rows7-12). Door and window orientation helps to different degrees for each house.

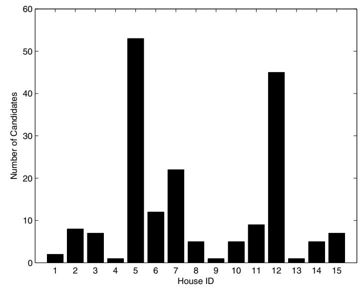
	House A			House B			House D		
Total Space Size	6.1×10^{23}			4.9×10^{22}			4.9×10^{22}		
House Constraints	Doorways	Windows	Windows +Doorways	Doorways	Windows	Windows +Doorways	Doorways	Windows	Windows +Doorways
	4.8×10^{21}	3.8×10^{17}	2.9×10^{15}	7.6×10^{20}	8.2×10^{16}	1.3×10^{15}	7.6×10^{20}	8.2×10^{16}	1.3×10^{15}
#Doorway	128	420	7	64	1728	12	64	3888	64
#Interior Wall	652,668,160	4,615,837,808	60,886,092	71,793,408	2,305,009,408	17,790,744	127,705,728	5,961,165,596	127,705,728
Constraints Conflicts	56,750,918	269,426	1,856	10,969,360	14,802,389	250,682	39,119,820	265,569,366	5,678,122
#Corners Rooms	4,740,246	13,700	49	1,148,492	733,174	12,090	6,489,208	28,248,664	491,038
Room Circles	938,212	4,148	18	74,168	35,562	2,517	456,888	2,132,489	27,436
Room Sizes	897,310	4,048	18	60,446	32,484	2,146	311,628	1,856,657	22,059
Overlapped Rooms	163,336	1,148	16	18,615	5,995	513	81,366	367,957	4,994
Room Adjacency	88	7	2	1,076	255	108	176	1,076	11
Duplicate Floor Plans	76	7	2	588	219	78	98	831	11
Duplicate Shapes	19	7	2	26	12	4	34	109	3

floor plans without checking global consistency properties, such as multi-room cycles or 2D planarity. Rows 7-12 show the effect of a number of filters that we perform to eliminate this type of floor plan.

– Room Cycles: filter floor plans in which two connected rooms are connected to the opposite walls of a third room. This topology is not feasible in a 2D plane.
– Room Sizes: filter floor plans with for which room sizes cannot be solved by the linear program, due to inconsistent room adjacency.
– Overlapped Rooms: filter floor plans that have overlapping rooms. This topology is not feasible in a 2D plane.
– Room Adjacency: filter cases where two rooms are physically adjacent, but the interior wall assignment indicates that they are not adjacent.
– Duplicate Floor Plans: filter floor plans that are duplicates of others. These plans are underspecified.
– Duplicate Shapes: filter floor plans where the size and location of rooms are identical to others, even if the identities of the rooms are different. Only a single shape can be presented to the user, and the room identities can be expanded in a separate step.



(a) Accuracy of Real Deployments



(b) Accuracy of Downloaded Floor Plans

Fig. 8. Accuracy is the number of candidate floor plans chosen, which determines how easy it is for a user to recognize their own house. (a) Houses A and B are helped more by window orientation and House D by doorway orientation. (b) Most of the 15 topologies downloaded from the Internet have fewer than 10 candidates. All candidate pools include the correct house topology.

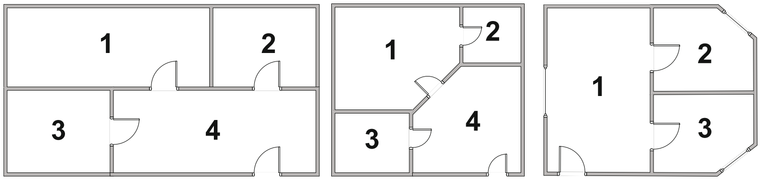
After these filters are applied, the remaining candidate floor plans are presented to the user. The user can select the correct floor plan from this pool of candidates.

8 Evaluation

Figure 8(a) shows the number of candidate floor plans that are generated for each of the three houses using a different set of structural constraints: door orientation, window orientation, or both. All candidate pools contain the correct house topology. The combination of doorway and window orientations has the smallest pool of candidates for all the three houses: 2, 4, and 3, respectively. In Houses A and B, we observe that window orientation is more effective than doorway orientation. This matches our analysis in Table 3, where window orientation shows more decrease in topology search space than doorway orientation. However, House D has a different trend where window orientation has the largest pool size, with 109 total candidate floor plans. The explanation for this anomaly is that House D only has windows on two walls, as indicated in Table 1. Thus, window orientation alone is not informative enough to filter the valid but incorrect topologies. In general, houses with more doors and windows will benefit more from information about their orientations.

Due to the difficulty of deploying large-scale sensor systems, our evaluation is limited to only four houses. Although the floor plans of the houses are highly varied, it is difficult to draw statistical conclusions from the results. To increase our sample set size, we collect another 15 house plans that are publicly available online[2]. These floor plans comprise a variety of single-bedroom and multiple-bedroom apartments and houses from various parts of U.S. By analyzing the blueprints, we manually identified room adjacency, doorway orientations, and window orientations in order to evaluate the efficacy of our search heuristics on these topologies. Figure 8(b) indicates the size of candidate

[2] `http://www.plansourceinc.com/apartmen.htm`



(a) Unaligned Walls (b) Angled Doorway (c) Angled Window

Fig. 9. Our system does not yet capture floor plans with non-conventional angles or unaligned walls

pool that our system produced for each house. In two cases, the candidate pool included approximately 50 floor plans. However, the average number of candidates is 12.2, and the candidate pools do contain the correct topology in all 15 test cases. This indicates that our search heuristics are effective for a large range of topologies and can be used to provide a reasonably small set of candidate floor plans to users.

9 Limitations and Future Work

The sensors, data analysis, pruning algorithms, and scoring heuristics that we use in this paper are designed to be generally applicable to a large number of houses. However, they are not applicable to all houses, and the floor plans of many houses will either not be found, or will be found but a large number of other candidates will be given higher rank. House C in our test set is one of these houses. In future work, we believe we can address most if not all of these limitations by extending our algorithms.

Our system performed poorly on the topology shown in Figure 4(c) for two reasons. First, the house has multiple floors, and the stairway sensors covered two floors. This caused the sensor clustering algorithm to group rooms on separate floors as a single room. Furthermore, this house has a very modern and open floor plan, which exacerbates the problem of overlapping sensing ranges that cover multiple rooms. In future work, we will need to address both multiple story buildings as well as open floor plans by developing new sensors and/or analysis algorithms.

Some other floor plans that we currently do not support are illustrated in Figure 9. These include floor plans with unaligned walls, doorways or windows that are angled, and rooms that do not adjoin at 90° angles. Of our four testbeds, two of the houses actually do have unaligned walls and our system generated a reasonable approximation of their floor plans. In future work, we will build on prior work in architecture [13] to specify floor plans using more general formalisms in order to efficiently search over a wider range of topologies.

The evaluation presented in this paper is based on the sensors available to a specific form of home energy management system: light, temperature, and motion sensors. The results are promising, despite the limited spatial constraints that were extracted from these sensors. In current work, we are exploring new ways to extract spatial constraints from these and other sensors. For example, we are exploring whether light sensors can be used to identify room locations based on artificial lighting patterns, and whether light sensors on the interior of a building can identify direction based on indirect sunlight levels. We are also exploring techniques to extract spatial constraints from more complex sensors such as microphones, tracking systems, and water/electrical usage.

10 Conclusions

In this paper, we present the Smart Blueprints system that simplifies the configuration of smart homes by automatically inferring the floor plan of a home and the locations of sensors in the house. This inferred information serves as physical context to better interpret sensors and configure actuators. We evaluate this approach using a combination of motion sensors and light sensors, and present novel inference techniques to infer their physical context within the home. We evaluate the system by deploying in four homes and by analyzing 15 typical floor plans found online. Our results indicate that our system can narrow the floor plan to a small number of candidates: between 2 and 4 for our deployment testbeds. Our approach requires a short period of data collection to automatically infer the floor plan and in our experiments this period ranged from 1 to 2 weeks. During this training period, we expect users to use a basic list-based interface until the graphical interface is ready. As an early stage prototype, our system does have several limitations in terms of the house and floor plans on which it can operate. It could not infer the topology of House C because it failed to generate accurate sensor clusters due to the modern and open floor plan. Similarly, it does not yet address non-conventional angles or alignment. In this paper, we present a proof of concept demonstration and indicate several open questions and new directions of research that could improve these results in the future. We believe the concepts can be extended to other applications, either by using the exact same sensor configurations or by generalizing the principles learned from our pilot study to other types of sensors that are needed for other applications.

Acknowledgements. Special thanks to our reviewers for very helpful feedback. This work is based on work supported by the National Science Foundation under Grants No. 1038271 and 0845761.

References

1. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Reconstructing Building Interiors from Images. In: IEEE 12th International Conference on Computer Vision, pp. 80–87 (2009)
2. Guo, W., Filer, N., Barton, S.: 2D Indoor Mapping and Location-sensing Using an Impulse Radio Network. In: ICU 2005: 2005 IEEE International Conference on Ultra-Wideband, pp. 296–301 (2005)
3. Surmann, H., Nüchter, A., Hertzberg, J.: An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. Robotics and Autonomous Systems 45(3), 181–198 (2003)
4. Schindler, G., Metzger, C., Starner, T.: A Wearable Interface for Topological Mapping and Localization in Indoor Environments. In: Hazas, M., Krumm, J., Strang, T. (eds.) LoCA 2006. LNCS, vol. 3987, pp. 64–73. Springer, Heidelberg (2006)

5. Harle, R., Hopper, A.: Using Personnel Movements for Indoor Autonomous Environment Discovery. In: PerCom 2003: the 1st IEEE International Conference on Pervasive Computing and Communications, pp. 125–132 (2003)
6. Mozer, M.: The Neural Network House: An Environment that Adapts to its Inhabitants. In: Proc. of the AAAI Spring Symposium on Intelligent Environments, pp. 110–114 (1998)
7. Kidd, C., Orr, R., Abowd, G.D., Atkeson, C., Essa, I., MacIntyre, B., Mynatt, E., Starner, T., Newstetter, W.: The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In: Yuan, F., Hartkopf, V. (eds.) CoBuild 1999. LNCS, vol. 1670, pp. 191–198. Springer, Heidelberg (1999)
8. Wolff, S., Larsen, P., Lausdahl, K., Ribeiro, A., Toftegaard, T.: Facilitating Home Automation Through Wireless Protocol Interoperability. In: WPMC 2009: the 12th International Symposium on Wireless Personal Multimedia Communications (2009)
9. Ferrari, G., Medagliani, P., Piazza, S., Martalò, M.: Wireless Sensor Networks: Performance Analysis in Indoor Scenarios. EURASIP Journal on Wireless Communications and Networking 2007(1), 41 (2007)
10. Schor, L., Sommer, P., Wattenhofer, R.: Towards a Zero-configuration Wireless Sensor Network Architecture for Smart Buildings. In: BuildSys 2009: the 1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp. 31–36. ACM (2009)
11. Granzer, W., Kastner, W., Neugschwandtner, G., Praus, F.: A Modular Architecture for Building Automation Systems. In: WFCS 2006: 2006 IEEE International Workshop on Factory Communication Systems, pp. 99–102 (2006)
12. Reinisch, C., Granzer, W., Praus, F., Kastner, W.: Integration of Heterogeneous Building Automation Systems Using Ontologies. In: IECON 2008: the 34th Annual Conference of the IEEE Industrial Electronics Society, pp. 2736–2741. IEEE (2008)
13. Charman, P.: A constraint-based approach for the generation of floor plans. In: Proc. of the 6th International Conference on Tools with Artificial Intelligence, pp. 555–561. IEEE (1994)
14. Lu, J., Whitehouse, K.: SunCast: Fine-grained Prediction of Natural Sunlight Levels for Improved Daylight Harvesting. In: IPSN 2012: the 11th ACM/IEEE Conference on Information Processing in Sensor Networks (April 2012)
15. Hnat, T., Srinivasan, V., Lu, J., Sookoor, T., Dawson, R., Stankovic, J., Whitehouse, K.: The Hitchhiker's Guide to Successful Residential Sensing Deployments. In: SenSys 2011: the 9th ACM Conference on Embedded Networked Sensor Systems, pp. 232–245. ACM (2011)
16. Srinivasan, V., Stankovic, J., Whitehouse, K.: Protecting your Daily In-home Activity Information from a Wireless Snooping Attack. In: UbiComp 2008: the 10th International Conference on Ubiquitous Computing, pp. 202–211. ACM (2008)
17. Lu, J., Birru, D., Whitehouse, K.: Using Simple Light Sensors to Achieve Smart Daylight Harvesting. In: BuildSys 2010: the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, pp. 73–78. ACM (2010)