A Genetic Algorithm approach to motion sensor placement in smart environments

Brian L. Thomas¹ · Aaron S. Crandall¹ · Diane J. Cook¹

Received: 25 March 2015 / Accepted: 17 November 2015 / Published online: 8 December 2015
© Springer International Publishing Switzerland 2015

Abstract Smart environments and ubiquitous computing technologies hold great promise for a wide range of real-world applications. The medical community is particularly interested in high-quality measurement of activities of daily living. With accurate computer modeling of older adults, decision support tools may be built to assist care providers. One aspect of effectively deploying these technologies is determining where the sensors should be placed in the home to effectively support these end goals. This work introduces and evaluates a set of approaches for generating sensor layouts in the home. These approaches range from the gold standard of human intuition-based placement to more advanced search algorithms, including Hill Climbing and Genetic Algorithms. The generated layouts are evaluated based on their ability to detect activities while minimizing the number of needed sensors. Sensor-rich environments can provide valuable insights about adults as they go about their lives. These sensors, once in place, provide information on daily behavior that can facilitate an aging-in-place approach to health care.

Keywords Smart environments · Sensor placement · Machine learning · Genetic algorithms

This material is based upon work supported by the National Science Foundation under Grants 1262814 and 0900781.

✉ Brian L. Thomas
bthomas@eecs.wsu.edu

Aaron S. Crandall
acrandal@wsu.edu

Diane J. Cook
cook@eecs.wsu.edu

¹ School of Electrical Engineering and Computer Science, Washington State University, Pullman, USA

1 Introduction

Smart environments, smart homes, and general pervasive computing technologies are rapidly becoming feasible and useful technologies for many applications. These technologies all rely on sensors to provide evidence for building machine learning models and giving data to their users. Selecting appropriate types of sensors to use and strategic locations to put them for effective observation are difficult challenges. This work explores techniques to identify “reasonable” sensor placement plans within the smart home context. The goal of these tools is to generate placements with little user input. The placements should optimize activity recognition performance while minimizing the number of sensors that are required.

1.1 Background

There has been a variety of work done on sensor placement in both specific and general contexts. These works have been applied to the fields of wearable devices [3], transportation [9], whole building health [4], and agriculture [1], among others. The goal of these systems is to monitor routine activities and behavior, which requires careful selection and placement of sensors. With costs ranging from \$20 to over \$100 USD per sensor today, it is important to minimize the number of sensors needed to accurately observe the space.

While long lived, low priced, wireless sensor networks have only truly been widely available for the last decade, the term “sensor network” has been around for well over 30 years [2]. A recent literature review on context-aware systems (systems that observe, interpret, and utilize the contextual information gathered with sensors to adapt their functionality to the current context [15]) from 2008 [19] reveals a dramatic

increase in research activity over a seven-year period, 2000 through 2007, in this field.

Most classic and many modern approaches to sensor placement perceive the problem as a need for sensor coverage over a region, and are derived from the NP-hard Art Gallery Problem [31]. The Art Gallery Problem states: Find the minimum set of guards placed in an enclosed region such that every point within the region has direct line-of-sight to at least one guard.

More recently, the addition of probabilistic models, such as Gaussian Processes [28], and Gaussian Matrices [27] have been utilized for improved optimizations [5], resulting in several new techniques for reducing the total number of sensors required to cover a space while still maintaining the same level of contextual knowledge about the space. One of these new optimizations [18] leverages *mutual information*. This is the process of finding sensor placements that are the most informative about unsensed locations [26]. Building upon this, Andreas Krause presents a data-driven algorithm for Sensor Placements at Informative and cost-Effective Locations (pSPIEL) [21] that optimizes for sensor locations, as well as communication within the wireless sensor network.

Moving away from the Art Gallery Problem, a group at the MIT Media Lab recently argued that if sensors become so pervasive that everyday life is saturated by sensors, then coverage problems are no longer much of an issue [23]. Building on the hypothesis that if sensors can be made unobtrusive enough then the technology will generally go unnoticed, this group attempted to build wireless sensors into everyday objects that are generally found throughout a home or office environment. What resulted was the MIT “Plug,” a wireless sensor node built in the form factor of a common electrical power outlet strip. The Plug integrated the following sensors: individual outlet current, passive infrared (PIR), light, vibration, and microphone. Automation and feedback were also integrated into the Plug: outlet relays, LED indicators, and a 1.5 W speaker. The Plug is an example of the rise of the Internet of Things (IoT) concept [8, 24], where ubiquitous networks operating both in Machine to Machine (M2M) and Machine to Human (M2H) modes provide services and monitor spaces with a high density of sensor deployments.

These two approaches show different major strategies to monitoring the home. One tries to optimize the number of sensors needed through specific placement, while the other seeks to saturate the space and mine out the important information after the fact. This work follows the former path by seeking methods to place sensors in effective locations. The goal is to produce tools that may be used to automatically evaluate where sensors should go in a home without an expert’s input. Choosing locations based on how effective they are at supporting end goals, such as detection of activities of daily living (ADL) [20], while limiting sensor counts to keep down monetary costs are important considera-

tions. Recent works on establishing caregiver needs [36–38] from smart environments have shown that a combination of ADL information [22] with change point alarms [17] would provide strong tools for the health care community when deploying IoT-based systems for in-home monitoring.

Current algorithms for ADL detection are already able to achieve very high levels of accuracy in human-designed sensor layouts [22]. The primary goal of this work is to remove the human designer from the process as much as possible while still achieving these levels of accuracy. Continued work to automatically design and deploy sensor networks are necessary steps in making them deployable on an IoT scale.

Data from real human performing activities in a monitored smart home testbed with camera-based verification of the ground truth are used to determine location and intent of the people. This data can then be used in an emulator to evaluate different sensor layouts for their effectiveness at successfully determining what the people are doing and when. The result of emulating human intent is a set of sensor event data that closely models sensor events generated from real-world human activity observation. Following the data flow in Fig. 1, Sects. 2, 3, and 4 discuss how the proposed system utilizes real-world data to evaluate proposed sensor layouts. Five different approaches to sensor placement are evaluated and compared for future use. Finally, one of the top-performing sensor layouts is installed in parallel with the baseline sensor layout and the resulting performance is discussed.

2 Data sources and collection

The data used for this project came from real humans performing activities in an instrumented smart home testbed. This corpus of sensor events was annotated by humans who tagged individual events with corresponding labels from a set of predefined activities. Finally, the data were processed to determine the ground truth paths the people followed while they performed their activities. These data sets, consisting of peoples’ paths paired with the activities being performed, could then be run through a simulator to evaluate hypothesized sensor placement strategies.

2.1 Testbed description

The CASAS testbed (Fig. 1, Node 1) that generated the data used for the experiments came from a townhouse apartment located on the Pullman campus of Washington State University. The facility, code named “Kyoto,” is equipped with a semi-grid of infrared motion detectors, light sensors, door sensors, temperature sensors, light switch monitors, and object shake sensors.

Fig. 1 Overall data flow for evaluating sensor layouts

The *Kyoto* testbed is the primary research facility for the CASAS project. The floor plan and sensor layout for this three-bedroom apartment is shown in Fig. 2. *Kyoto* is designed to be a sensor-rich space designed for capturing as many ADLs and behaviors as possible.

Since its initial installation in 2007, this smart home testbed has undergone a series of improvements. These have primarily been software updates, but over time new sensors and interactive technologies have also been deployed with the focus of supporting the CASAS research objectives, such as early onset dementia evaluation [29], aging-in-place technologies [13], and the associating of activities with

Fig. 2 Map of current motion sensor locations in the CASAS *Kyoto* testbed

energy consumption [6, 7]. This testbed has proven highly successful at gathering rich and well-documented data sets, many of which are available publicly [35]. A more detailed description of this testbed can be found in other CASAS works [11].

This testbed has also lead to other strategies for sensor placement in CASAS smart environments. One of the new strategies that has been employed establishes sensor locations in relationship to key items and special locations instead of through regular or algorithmic layouts. This approach is utilized in the CASAS Smart Home in a Box (SHiB) projects [10, 12]. The results from these projects are promising, though there are significant tradeoffs in ease of installation and success of modeling based upon the available sensors. This work on algorithmic designs competes with the *Kyoto* design because the concept of complete coverage is more prevalent in the field. Future works will seek to compare coverage with key locations SHiB style approaches and their impacts on modeling, cost, and installation ease.

Table 1 Example sensor data with annotation tag

Date	Time	Location	Message	Tag
2008-03-03	14:13:58.873547	M014	ON	Cook Begin
2008-03-03	14:13:58.997577	M013	ON	
2008-03-03	14:13:59.144543	M009	OFF	
2008-03-03	14:13:59.984724	D007	OPEN	
2008-03-03	14:14:01.033088	M014	OFF	
2008-03-03	14:14:04.578001	I008	ABSENT	
2008-03-03	14:14:09.120758	M013	OFF	Cook End
2008-03-03	14:14:09.622629	M016	ON	Eat Begin
2008-03-03	14:14:12.413874	M016	OFF	

2.2 Data event format

Data collected from the sensor platform are stored as a four-tuple: <Date, Time, Location, Message>, and are individually referred to as “events” (Fig. 1, Node 2). An example of this data format can be seen in Table 1. These four fields are augmented by a fifth when tagged, also known as annotated, with the activity being performed.

2.3 Data collection

Data used in the current study were generated by people performing an augmented set of scripted activities, which are based upon the common Activities of Daily Living (ADL) set from the gerontechnology field [33]. The set of activities appearing in this dataset includes both cued [30], via an intercom system, and uncued tasks [14], as listed in Fig. 3.

Fig. 3 Activities of daily living performed in the data set

Uncued Tasks

1. Sweep the kitchen and dust the living room.
2. Obtain a set of medicines and a weekly medicine dispenser, fill as per directions.
3. Write a birthday card, enclose a check and address an envelope.
4. Find the appropriate DVD and watch the corresponding news clip.
5. Obtain a watering can and water all plants in the living space.
6. Answer the phone and respond to questions pertaining to the video from task 4.
7. Prepare a cup of soup using the microwave.
8. Pick a complete outfit for an interview from a selection of clothing.

Cued Tasks

9. Check the wattage of a desk lamp and replace the bulb.
10. Wash hands with soap at the kitchen sink.
11. Wash and dry all kitchen countertop surfaces.
12. Place a phone call to a recording and write down the recipe heard.
13. Sort and fold a basketful of clothing containing men’s, women’s, and children’s articles.
14. Prepare a bowl of oatmeal on the stovetop from the directions given in task 12.
15. Sort and file a small collection of billing statements.
16. Setup hands for a card game, answer the phone and describe the rules of the game.

3 Data pre-processing

For this experiment to work, the raw data collected in Sect. 2 were not enough. The sensor placement algorithms require the true physical paths of the people (Fig. 1, Node 8), paired with the activities they were performing (Fig. 1, Node 4). To derive this ground truth trajectory data, the two-part CASAS Augmentable Modular emulation System (CAMS) tool was developed. Input to the CAMS tool consists of a detailed description of the site in which the data were gathered (Fig. 1, Nodes 5 and 7). With a good description of the site, CAMS Human Path Generation (Fig. 1, Node 6) uses its algorithm to derive the original path in terms of the absolute coordinates that were visited and the walking speeds that were used.

3.1 XML Home Definition File

To create the XML Home Definition file (Fig. 1, Node 7), a 1/8 m grid is superimposed on a floor plan of the site. All X,Y measurements use a 0 for the first grid location index, with the origin in the upper left corner. It is assumed that walls block the view of sensors and people cannot walk through them. People can walk through lintels, the small piece of wall above most doors, while sensors on the ceiling can only “see” past them if they are more than 1/2 m from the lintel. Furniture does not obstruct the viewing area of sensors, though people cannot walk through it. For example, a couch would be described as the back and two arm rests, as a person can occupy the couch but it is assumed the person will not likely jump over the back. As seen in the shortened example site configuration of Fig. 4, there are two important attributes in the root node of the XML structure, *max_width* and *max_height*, which describe the absolute maximum size

```

<site name="kyoto" max_width="43"
      max_height="68">
  <floors>
    <floor level="1">
      <spaces>
        <space name="kitchen">
          <rectangle x="24" y="33"
                    width="18" height="22" />
          <rectangle x="28" y="55"
                    width="14" height="4" />
          ...
        </space>
        ...
      </spaces>
      <walls>
        <wall x="0" y="0"
              width="29" height="1" />
        <lintel x="29" y="0"
               width="8" height="1" />
        ...
      </walls>
      <furniture>
        <rectangle x="10" y="8"
                  width="5" height="10"
                  name="table" />
        <rectangle x="22" y="1"
                  width="1" height="16"
                  name="couch" />
        ...
      </furniture>
      <off_limits>
        <rectangle x="24" y="33"
                  width="4" height="16" />
        ...
      </off_limits>
      <sensors>
        <motion id="M001" x="12" y="31" />
        <motion id="M018" x="32" y="52" />
        ...
        <door id="D001" x="35" y="61" />
        ...
        <item id="I004" x="36" y="44" />
        ...
      </sensors>
    </floor>
    ...
  </floors>
</site>

```

Fig. 4 An abbreviated XML Home Definition file

in scaled grid coordinates to which the site will expand in the *X* and *Y* directions, respectively.

The *off_limits* definitions allow for identifying ceiling features that sensors cannot be attached to. This also presents a feature for future cases where a person may not want sensors to be installed in certain rooms. The algorithms can then do their best to work around this deficit and optimize with what locations they have available.

3.2 CAMS Human Path Generator

The CAMS tool consists of two parts, the generator and emulator, Nodes 6 and 10 in Fig. 1, respectively. CAMS

```

def find_next_valid_event(events, cur_event):
    events = all_input_data_events
    i = cur_event
    j = i + 1
    while ((dist[i][j] / (events[j].t_stamp -
                        events[i].t_stamp)) > 10):
        j += 1
    return[j]

```

Fig. 5 Python pseudo code for filtering out of order data events as noise

Human Path Generation requires as input the Annotated Sensor Events data file (Fig. 1, Node 4), the XML Home Definition file (Fig. 1, Node 7), and the sensor layout (Fig. 1, Node 5) used to capture the Annotated Sensor Events.

The Human Path Generator takes as input an annotated data file of a person's scripted activities in the smart apartment, an XML home definition file, and the original sensor layout file. From the XML Home Definition file and sensor layout file, the generator builds a map of the space including walls, lintels, furniture, and sensors. Using this map, the generator knows where the various sensors are located and where a person can walk. From this initial map, the generator also creates a fully connected graph where every vertex is an *<x,y>* coordinate on the map and every edge is a path that the person can take to traverse the map. The generator now builds a list of valid paths that the person can take. In the current model, the generator assumes when a motion sensor is tripped, the person is directly under said sensor. Using this assumption, the traversal graph, and Dijkstra's algorithm [16], an optimal move by the person can be modeled.

The original data have time stamps for every event. Starting from one event and looking to the next, the speed at which the person was observed to traverse the path within the given time is calculated using the distance between the sensors divided by the time difference between the events. If the resulting movement speed is under 10 m/s, then this move from one sensor to another was added to the path of the person.

Alternatively, if the speed of the current possible path is too great, then the move to the adjacent sensor event is ignored. Instead, the generator proceeds to attempt a move from the previous location and time to the next available event. The generator will continue skipping events until one is reached such that the speed traveled is less than 10 m/s. This algorithm is shown as pseudo code in Fig. 5. Roughly 1 % of the data had some form of noise that resulted in a person moving faster than 10 m/s, most often because of sensor communication errors that re-ordered events or other jitter in data collection traffic.

Fig. 6 A snippet of a single chromosome translated to sensors by the simulator. Each ‘1’ in the binary string becomes a single motion sensor

The output of the CAMS Human Path Generator is a data file (Fig. 1, Node 8) representing the movements of a person consisting of: $\langle timestamp, residentID, X, Y, speed, annotation \rangle$. By generating these “movement” files representing the ground truth of a person’s location and activity, different sensor layouts for the site can now be explored. This process hides the original “real world” sensor data files from the CAMS emulation, sensor layout, and accuracy calculation algorithms. Once these movement files representing the original ground truth of people moving through the space have been generated, generation and evaluation of different sensor layouts may begin.

4 Sensor layout definition and evaluation

After combining the XML Home Definition described in Sect. 3.1 (Fig. 1, Node 7), the sensor layout described in Sect. 4.1 (Fig. 1, Node 5), and human path traversal data from the CAMS Generator in Sect. 3.2 (Fig. 1, Node 8), hypothesized sensor layouts (Fig. 1, Node 9) may now be evaluated and compared. This process is shown in Fig. 1 Node 10, starting with “CAMS Emulation”.

4.1 Sensor layout definition

A sensor layout is defined as a fixed-length string of ‘1’s and ‘0’s where the length is $max_width * max_height$, as defined in the site description file from Sect. 3.1. Each ‘1’ represents a motion sensor placed at that location and each ‘0’ represents no sensor at that location. A subset of the a layout and the resulting sensor placements is shown in Fig. 6. All algorithms evaluated in this paper utilize this same sensor layout format.

4.2 Layout evaluation

Each layout is evaluated in a 2-step process. First, the layout is given to an emulator as defined in Sect. 4.2.1. Utilizing the sensor model defined in Sect. 4.2.2, the emulator uses

the provided layout with the set of human walking paths (Fig. 1, Node 8) to generate an emulated event stream (Fig. 1, Node 11). This event stream is then evaluated, as defined in Sect. 4.2.3, resulting in a score for the provided layout (Fig. 1, Node 15).

4.2.1 CAMS emulator

To evaluate how well a given hypothesized sensor layout might perform, a smart home emulator was developed. This emulator was designed to approximate a living space instrumented with various sensors and to output an approximated data set of events, as if a given sensor layout were to be actually installed in the home. By building an emulation of the home, instead of physically installing sensor layouts repeatedly, a large number of hypothesized sensor layouts may be tested in rapid succession.

This emulation tool requires three data files. The tool outputs a new data file with emulated sensor events and the activities that were performed, as determined by the original human annotators. The input files, which are also shown in Fig. 1, consist of:

- *XML Home Definition*: A file, as described in Sect. 3.1, which defines features of the walls, ceilings, floors, and furniture to build a model of the home.
- *Sensor Layout*: A file, as described in Sect. 4.1, defining where and which types of sensors it should simulate.
- *CAMS Generated Human Paths*: A movement file as formatted by the CAMS Generation tool from Sect. 3.2.

With the layout of the home, the location of sensors and a movement file, the emulator can then output a data set of events. As the simulated person moves throughout the space, according to the movement file, and occupies the areas that the emulated sensors can see (described in Sect. 4.2.2), events are generated as if they were in a real smart environment. These events come in the form $\langle date, time, location, message \rangle$, with an optional 5th field for activity beginning and endings, as shown in Table 1.

This output also includes which activities were being performed by the person, whether or not emulated sensors generated events during that time. By including the activities even if they were not “seen” by the sensors, the algorithm for scoring the quality of the hypothesized sensor layout can calculate false negatives.

Once the whole movement file has been replayed once through, the emulator terminates. The resulting data file can then be scored by how well other tools, such as the activity recognition algorithm, accurately detect activities being performed by the resident.

4.2.2 CAMS sensor model

The emulated motion sensors are designed to replicate the ceiling-mounted sensors already installed in *Kyoto*. Each sensor can detect motion with a radius of 0.75 m of the given location. However, the sensor cannot view motion through walls or past lintels within 0.25 m of the sensor. Once a motion path encounters a sensor's detectable area, the motion sensor will send an "ON" event, only sending an "OFF" event once there has been no motion in the detectable area for 2.5 seconds.

4.2.3 Scoring the hypothesized sensor layout for activity recognition

Sensor layouts were scored using a tool designed to detect the ADLs. This Activity Recognition (AR) tool [32] uses a Naïve Bayes model to determine which activities are being performed based on evidence from the sensors. The AR tool is used to score a given sensor layout by how accurately it is able to classify activities over time, by processing the emulated data as a continuous stream.

5 Sensor placement algorithms

This work uses five algorithms for determining sensor layouts. These include:

1. Human Intuition-Based
2. Monte Carlo-Based
3. Two-Dimensional Uniform Placement (Grid)
4. Hill Climbing
5. Genetic Algorithm

It is hypothesized that the Genetic Algorithm will provide the most effective approach without requiring an exhaustive search. With 2388 possible sensor locations, an exhaustive search would require evaluating 2388! layouts. To approach an exhaustive search, the Hill Climbing method, described in Sect. 5.4 and discussed in Sect. 7.1.3, was developed to provide a comparison closer to an exhaustive search. The next sections describe each of these algorithms in detail. A discussion of their behaviors and results follows in Sect. 7.

5.1 Human Intuition-Based placement

Utilizing current smart home industry standard techniques, sensor locations are selected for the environment by human researchers. A simulation of the existing *Kyoto* smart home testbed sensor layout, as determined by human engineers, was used as a basis of comparison. The sensor placement in *Kyoto* was designed to support multiple research objectives, including ADL detection, tracking, identification, and power

monitoring. The design was defined by human researchers relying on their own intuition and knowledge of the deployed technologies as to what might be a good sensor layout strategy. What resulted was a relatively high-density configuration of motion detectors, some 26 in all, as shown in Fig. 2. The simulated sensor layout evaluated here is a replication of the same layout used to gather the initial data used in this work, as well as for other projects.

5.2 Monte Carlo-Based placement

A Monte Carlo-based location selection algorithm was designed and tested as an algorithmic baseline for comparison. The Monte Carlo algorithm randomly selects n distinct sensor locations from all possible locations in the home. The value for n was evaluated for each entry in the set: {10, 20, 26, 30, and 40}. For each value of n , the algorithm was run 100 times with leave-one-out cross validation.

5.3 Two-Dimensional Uniform Linear Distribution Placement (Grid)

The sensor layout in *Kyoto* is based around a relatively simplistic fixed interval grid of sensors in each room. This approach is fairly common in smart home implementations that use many of the same sensor type in a single room. As a means to evaluate this kind of uniform linear distributions of sensors, all possible permutations of two-dimensional intervals ranging from 1/4 to 2.5 m in 1/8 m increments were tested. Each interval was also tested for $n \times m$ offsets from the origin, where $0 \leq n$ and $m < \text{interval}$. All generated layouts were evaluated with a 30-fold holdout validation.

5.4 Hill Climbing-Based placement

Hill Climbing algorithms apply a greedy approach to finding reasonable solutions in the search space. In this instance, Hill Climbing was applied by incrementally adding more sensors. A pseudo code outline of the implementation used for this work is shown in Fig. 7. Starting with the 2388 possible sensor locations, the algorithm iterates over all possible sensor locations, generating a layout for each one. Each layout is then evaluated and scored with the ADL-based fitness function, described in Sect. 4.2.3, to determine the top performer for each activity. For the top performer in each activity, the algorithm iterates over that layout in the next generation. This process is repeated for "depth" iterations.

5.5 Evolutionary placement via Genetic Algorithms

A Genetic Algorithm (GA) uses a form of parallel search to explore the search space quickly [25]. GAs do not guarantee an optimal solution, but the reduction in computation

```

def hill_climbing(layouts, depth):
    events = all_input_data_events
    activities = different_activities
    for i in sensor_locations:
        score = eval(i, events)
        results = []
    for a in activities:
        results.append(get_top_score(a))
    if (depth > 0):
        results = hill_climbing(results, depth-1)
    return results

```

Fig. 7 Python pseudo code of the Hill Climbing placement algorithm

time compared with exhaustive search algorithms is valuable. Given the high dimensionality of this problem's space, using a GA is a natural fit. Over many generations within the GA operation, the algorithm is designed to iteratively discover more effective sensor layouts.

5.5.1 Chromosome and phenome

The chromosome for the Genetic Algorithm utilizes the same sensor layout definition as provided in Sect. 4.1. This translation from a binary string to sensor locations is how it expresses its Phenome, as shown in Fig. 6. The population member data structure additionally stores information about itself, such as the generation it was created, its calculated accuracy for each individual activity, and overall fitness score.

5.5.2 Fitness function

The fitness function utilizes three features of the evaluated chromosome, which are shown in Fig. 8's pseudo code of this calculation. The fitness score is initialized to the average recognition accuracy of all activities as classified utilizing a specific sensor layout. Next, a sensor cost penalty is removed from the fitness for every sensor in the chromosome, to bias the population towards a smaller sensor distribution. The cost penalty value is calculated as the number of sensors in a given chromosome divided by 20, which appears in Fig. 8's pseudo code as (mNumSensors/20). The divisor value of 20 was discovered through experimentation. Higher values for the divisor did produce lower sensor counts, but applied too much pressure towards reduced coverage in the space, resulting in lower accuracies in detecting ADLs. Lower divisor values allowed for increased accuracies in ADL detection, but resulted in sensor counts well above 200. Next, the accuracy for each individual activity is compared to the current population's global average accuracy for that activity. For each activity, if the individual's accuracy is greater than the population's average, then the difference is added to the fitness value. This final fitness value is then compared against

```

def calculate_fitness():
    gADL = Global_Population_ADL_Accuracies
    mADL = My_ADL_Accuracies
    mNumSensors = Number_Sensors_In_Hypothesis
    fitness = average(mADL.all_accuracies)
    for i in ADL:
        if mADL[i].accuracy > ADL[i].accuracy:
            fitness += (mADL[i].accuracy
                       - ADL[i].accuracy)
    fitness -= mNumSensors/20
    return fitness

```

Fig. 8 Python pseudo code for calculating a chromosome's final fitness score

other chromosomes in the population for reproductive selection.

5.5.3 Reproduction

During reproduction, the top 40 % of the population by fitness score are selected for reproduction. A pseudo random 4 multi-point crossover is performed with each breeding pair. A mutation operator is applied where the new child's sensor locations have a 0.2 % chance of inverting their value. Finally, the top 30 % of the parent generation are allowed to survive into the next generation. These values were arrived at after extensive experimentation to allow retaining of important sensor locations without hindering the exploration of new sensor combinations and locations by the population.

5.5.4 Population configuration

The population is set and initialized with 100 members. Each member's chromosome is initialized with a seed size of 5, utilizing a pseudo random number generator to select locations for five sensors in the otherwise all zeros chromosome. The Genetic Algorithm then runs for 300 generations before terminating.

5.6 Algorithms summary

These different algorithms provide a range of diverse underlying methodologies for sensor selection. These algorithms were selected to give a good evaluation of the current approaches to sensor placement and contrast those by introducing more successful and scalable tools to help with real-world deployments.

6 Evaluation

Using the data described in Sect. 2.3, each algorithm was individually evaluated with a leave-one-out cross validation method utilizing data from 30 different participants. The

Fig. 9 An example of how time-based accuracies are calculated

results for each algorithm provide a mean and standard deviation to determine statistical significance of the results.

Many activity detection algorithms are scored on an event-by-event basis. Because the ground truth events are effectively hidden from the final AR classification during the CAMS simulation phase, the final AR output does not “line up” with the ground truth input files on an event-by-event basis. This means a mechanism other than sensor event count needs to be used for accuracy evaluation. Instead, this work’s accuracy metric was calculated using *total time correct* and not by sensor event count.

The ground truth data files included annotations of when activities began and ended at distinct times. The goal of a placement algorithm is to place sensors such that the activity recognition algorithm outputs activity begin and end times that match the ground truth. An example of the ground truth not fully lining up with the classified output is shown in Fig. 9. This method of accuracy calculation allows the tool to determine the correct outcomes True Positive (TP) and True Negative (TN), as well as Type I False Positive (FP) and Type II False Negative (FN) errors.

Each fold of the evaluation was run through the CAMS emulator as described in Sect. 4.2.1. The output from the emulator is a data file of events paired with the ground truth activities to be classified by the AR Tool [33]. The output from the AR tool is an ADL label for each event. The ADL tagged events are then processed to determine begin and end times for activities. The final list of ADL begin and end times can then be used to calculate the accuracy of the given sensor layout.

Determining accuracy is done by comparing ground truth to the AR output at a millisecond resolution. At each millisecond, the ADL guess is compared to the ADL ground truth. The score for each ADL is kept separately with the true positive, false positive, true negative, and false negative values. A fold’s accuracy for a given algorithm is the average of the true positive rate minus the false positive rate for all ADLs. All 30 folds are used to generate the algorithm’s accuracy mean and standard deviation.

7 Results

The overall accuracy of these placement strategies is summarized in Fig. 10. The Monte Carlo approach with 26 sensors

Fig. 10 Overall accuracy for all algorithms. Error bars two standard deviations from the mean

and the grid 10×10 layout are shown in this chart. The error bars show two standard deviations from the mean. As the figure shows, all five algorithms exhibit varying degrees of success. While Hill Climbing gets very high classification accuracies, only the GA is significantly better ($p < 0.05$) than the baseline algorithms.

7.1 Results analysis

To analyze the results from these tools in greater depth, we examine individual accuracies and the associated physical layouts. The three baseline algorithms, human, Monte Carlo and grid, exhibited notably different approaches to sensor placement than the GA and Hill Climbing approaches. The various tools also leveraged temporal and physical layouts in different ways.

Looking across all activities for accuracy in Fig. 11, there is notable variation in the ability to accurately recognize alternative activities. Previous work on ADL detection has noted similar issues with classifying highly mobile and subtle activities such as cleaning, exercise and socialization [32, 34]. The GA and Hill Climbing approaches overcome some of this difficulty by discovering the best places to detect these subtle divisions between activities.

7.1.1 Baseline Algorithms results discussion

The baseline algorithms generated layouts that covered more of the space, including areas that are not utilized by the tar-

Fig. 11 Per ADL accuracy for all algorithms

Fig. 12 Best baseline layouts

geted ADLs, as shown in Fig. 12. In contrast, the GA and hill climbing approaches clustered sensors around the areas that physically divide activities, as shown in Figs. 14 and 17.

The Human and Grid-based approaches both share a common underlying theory of covering as much of the home as possible to capture activity. Since the Human-designed installation was limited by monetary and physical installation costs, it was a balance between cost and resolution of the final data received from the sensor platform. The simulated Grid approach did not have these limitations. For example, the 2×2 grid simulates an environment with sensors placed every $1/4$ m, leading to this test space having 630 sensors. This is too monetarily expensive for most home installations.

By charting the Grid resolution sizes against accuracy, and aligned with sensor count in Fig. 13, the 10×10 grid spacing is a definite point of interest. The 10×10 grid, which is at a roughly 1.5m resolution, has the maximal accuracy benefit

Fig. 13 Accuracy and sensor count plotted against grid layout sensor resolution. The *black line* is the average accuracy, while the grey field extends shows the variation in results at the given grid resolution

per sensor. Adding more sensors by reducing the interval distance between them has tiny incremental improvements at a notable sensor count increase.

The 10×10 spacing is very similar to the human-designed layout, in both space between sensors and general placement. The result is also very similar in quantity of sensors to the best GA, which runs at around 24–26 sensors.

7.1.2 Genetic Algorithm results discussion

Examples of some GA run results in Fig. 14 show that sensors in the center of the living room and kitchen are effective zones for providing evidence to differentiate between activities. Evidence of the person in these regions is important for providing information to the activity recognition system that helps to differentiate subtly different activities. As expected, the GA improved its overall accuracy with the number of generations, as seen in Fig. 15. Additionally, in Fig. 16, it can be noted that each ADL accuracy improved before stabilizing, where one ADL may drop in accuracy to benefit the rest before the algorithm finds a way to bring all accuracies back up.

The GA algorithm included a sensor count penalty in the fitness function. For every sensor, the final fitness score was

Fig. 14 Example GA layouts

Fig. 15 Genetic Algorithm accuracy by run generations

Fig. 16 Per ADL accuracy by run generations

Fig. 17 Examples of Hill Climbing produced layouts

reduced by (Sensors Count/20), as noted in Sect. 5.5.2. This placed a downward pressure on adding more sensors and forced the population to optimize over adding more sensors. This pressure tapered the introduction of new sensors and the best balance of sensors to accuracy leveled off at 26 sensors for most runs of the algorithm.

7.1.3 Hill Climbing results discussion

Similar to the GA layouts, an example of a Hill Climbing generated layout in Fig. 17a also highlights how sensors targeting the physical spaces between where activities take place provides significant benefits in detecting context switches from one activity to another. We hypothesize that if the Hill Climbing algorithm continued to add more sensors, they would have begun to focus on the locations where multiple activities physically overlap to provide more discriminatory power to the AR tool.

Figure 17b shows a summation of all 30 runs of the Hill Climbing algorithm. The Hill Climbing results also show benefits to exploiting temporal information about *when* residents are likely to be at certain locations. Overall, the

algorithm targeted a similar area to the GA approach, though it avoided the kitchen.

The results from the Hill Climbing tool seem counter intuitive at first, given that the sensors are clustered by this algorithm in an open area, and are rarely placed in the kitchen as was prevalent with the GA approach. After inspecting the interaction of the AR tool and the simulated output from the Hill Climbing layouts, this method is observed deriving such high accuracies due to the nature of the human paths produced during the experiments. Many of the people participating in the experiments walked through the living room, directly under the locations chosen by the Hill Climbing tool, to get instructions about performing the current and next activities from a two-way communication system. Combining this small evidence of movements from a few sensors with temporal information is enough for the AR tool to determine which activity is currently being performed. In a less scripted environment, this sensor placement behavior is unlikely to be successful.

Hill Climbing was surprisingly efficient at exploiting the nature of the data and the ADL recognition. The results are very good, but expensive to obtain. Because our Hill Climbing algorithm has a significant branching factor in the search space with each iteration, to the order of $(2388 - \text{NumSensors}) \times \text{length (ADL list)}$, it took weeks running to generate these results, compared to much faster evaluations with the other algorithms. The *depth* of the algorithm was initially set to 26, but after two weeks execution, the algorithm had only reached a depth of 4. This extended calculation time is infeasible for deploying sensors quickly in a home, unless week-plus wait times are acceptable.

7.2 Results summary

The GA algorithm's layouts are more efficient to generate than Hill Climbing, in addition to being significantly more accurate than the baseline algorithms tested here. The GA converged to a similar number of sensors for this home and data as the human-driven and Grid-based solutions, which indicates something about the nature of the space, behaviors and ADL detection software.

7.3 Validation of results

One of the top-performing GA layouts was installed in the *Kyoto* testbed, shown in the middle image of Fig. 14. A photo showing the ceiling of the *Kyoto* living room with both the human-designed sensor layout and the GA sensors is shown in Fig. 18. The sensors highlighted in red are from the GA's design.

Data were collected for three months while new experiments were carried out. A subset of the activities performed that matched the original experiment were utilized to mea-

Fig. 18 Image of the GA sensor placement installed in parallel with the human placement. GA sensors have been highlighted in red

Table 2 Accuracy of ADL activities from real-world installations of layouts

ADL	#	Baseline	GA
Sweeping and dusting	1	98.795880	98.197847
Water plants	5	98.852401	98.415053
Cooking	7	99.518364	99.816572
Hand washing	10	94.118849	99.441648
Wash countertop	11	96.482625	98.782685
Recipe	12	61.868240	82.014987
Oatmeal	14	89.929476	97.710546
Total		91.366548	96.339905

sure and compare the accuracy of AR on both the baseline and the installed GA layout. The AR accuracy on the baseline was 91.3665 %, and the GA achieved 96.3399 %. While not statistically significant, this does show that in real-world conditions, the GA layout was not outperformed by the baseline layout. Per ADL accuracies can be found in Table 2.

8 Conclusion

There is a need for tools and techniques to strategically place sensors in smart homes. These processes need to be

scalable, repeatable, and ideally invisible to users of the system. Appropriate placement heavily influences the ability for modeling tools to perform their tasks, which directly affects the performance of the smart home's monitoring of people. For applications such as medical monitoring, highly accurate models are a must for their clinical utility in the field.

Current approaches for sensor placement address some of the issues faced in smart home applications, but not all. The competing concerns of modeling accuracy, monetary cost, maintenance overhead and installation complexity need to all be considered. Finding algorithms that are quick, effective and sensitive to these issues are vital in moving towards larger scale deployments.

While the Hill Climbing approach generated very efficient layouts, it also overfitted the data. Additionally, it is very computationally expensive, often taking weeks to generate results where the GA took hours. This kind of exhaustive search is too unwieldy for real-world deployments, especially as the size of the monitored space grows.

The algorithms tested here represent the current approaches used by much of the smart home community and demonstrate that there is a place for algorithmic tools in effective sensor placement. The results from this work show a significant improvement over baseline algorithms using a Genetic Algorithm. Given the effective size of the search space, heuristic and clever algorithms such as the Genetic Algorithm are needed to make these tools useful in real-world applications.

9 Future work

This work scratches the surface of automatically placing sensors in smart homes. The simulator should be expanded to handle a much wider variety of sensor types and positioning. Data from multi-inhabitant sites with more varied activities should be included.

There are also plenty of other algorithms available for this kind of search. Gaussian Processes, simulated annealing, and others could be applied to this problem.

A wider range of fitness functions could also be used to compare which kinds and differing sensor layouts are needed for various end goals of the system. For example, changing from a simple Naïve Bayes ADL detection algorithm to the latest tools, or to use algorithms that attempt to track and identify individuals by behavior, or evaluate energy consumption.

Given a good model of how humans live in their homes, a set of generic virtual humans could be used to evaluate the system instead of expensive real-world data. For a new home, providing only the floor plan and appliance locations could be enough to run a long simulation with a set of generic virtual humans. This could be done before the smart home instal-

lation begins, and give the residents a good layout without requiring initial path tracking and activity data collection.

Eventually, when enough home floor plans have been tested, generic rules may be extracted. For example, “place one motion detector 1 foot inside each room’s doorway” or “the kitchen needs at least 3 sensors.” This kind of generic rule may get “good enough” for most home installations and would require notably less processing power.

References

1. Abbasi AZ, Islam N, Shaikh ZA et al (2014) A review of wireless sensors and networks' applications in agriculture. *Comput Stand Interf* 36(2):263–270
2. Agency, U.S.D.A.R.P., Dept, C.M.U.C.S.: Distributed sensor nets: proceedings of a workshop held at Carnegie-Mellon University. Computer Science Department, Carnegie-Mellon University (1978)
3. Atallah L, Lo B, King R, Yang GZ (2010) Sensor placement for activity detection using wearable accelerometers. In: 2010 international conference on body sensor networks (BSN). IEEE, New York, pp 24–29
4. Bhuiyan MZA, Wang G, Cao J, Wu J (2014) Sensor placement with multiple objectives for structural health monitoring. *ACM Trans Sens Netw (TOSN)* 10(4):68
5. Cerpa A, Wong JL, Kuang L, Potkonjak M, Estrin D (2005) Statistical model of lossy links in wireless sensor networks. In: Proceedings of the 4th international symposium on information processing in sensor networks, IPSN '05. IEEE Press, Piscataway. doi:[10.1109/IPSN.2005.1440900](https://doi.org/10.1109/IPSN.2005.1440900)
6. Chen C, Cook DJ (2010) Novelty detection in human behavior through analysis of energy utilization. In: Human behavior recognition technologies. IGI Global, Pennsylvania, pp 65–85. doi:[10.4018/978-1-4666-3682-8.ch004](https://doi.org/10.4018/978-1-4666-3682-8.ch004)
7. Chen C, Das B, Cook D (2010) Energy prediction based on resident's activity. In: Proceedings of the international workshop on knowledge discovery from sensor data, SensorKDD '10 (2010)
8. Chen YK (2012) Challenges and opportunities of internet of things. In: Asia and South Pacific design automation conference, ASP-DAC, pp 383–388. doi:[10.1109/ASPDAC.2012.6164978](https://doi.org/10.1109/ASPDAC.2012.6164978)
9. Cheng X, Du DZ, Wang L, Xu B (2008) Relay sensor placement in wireless sensor networks. *Wirel Netw* 14(3):347–355
10. Cook DJ, Crandall AS, Thomas BL, Krishnan NC (2013) CASAS: a smart home in a box. *IEEE Comput* 46(7):26–33. doi:[10.1109/MC.2012.328](https://doi.org/10.1109/MC.2012.328)
11. Crandall AS (2011) Behaviometrics for multiple residents in a smart environment. Ph.D. thesis, Washington State University
12. Crandall AS, Cook DJ (2012) Smart home in a box: a large scale smart home deployment. In: Workshop proceedings of the 8th international conference on intelligent environments, IE, vol. 13. IOS Press, Amsterdam, p 169 (2012). doi:[10.3233/978-1-61499-080-2-169](https://doi.org/10.3233/978-1-61499-080-2-169)
13. Das B, Krishnan NC, Cook DJ (2012) Automated activity interventions to assist with activities of daily living. In: Agents and ambient intelligence. IOS Press, Amsterdam, pp 137–158. doi:[10.3233/978-1-61499-050-5-137](https://doi.org/10.3233/978-1-61499-050-5-137)
14. Dawadi P, Cook D, Parsey C, Schmitter-Edgecombe M, Schneider M (2011) An approach to cognitive assessment in smart home. In: Proceedings of the 2011 workshop on data mining for medicine and healthcare, DMMH '11. ACM, New York, pp 56–59. doi:[10.1145/2023582.2023592](https://doi.org/10.1145/2023582.2023592). <http://doi.acm.org/10.1145/2023582.2023592>

15. Dey AK (2001) Understanding and using context. *Pers Ubiquit Comput* 5(1):4–7. doi:[10.1007/s007790170019](https://doi.org/10.1007/s007790170019)
16. Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271. doi:[10.1007/BF01386390](https://doi.org/10.1007/BF01386390)
17. Feuz KD, Cook DJ, Rosasco C, Robertson K, Schmitter-Edgecombe M (2014) Automated detection of activity transitions for prompting. *IEEE Trans Human-Mach Syst* 99:1–11. doi:[10.1109/THMS.2014.2362529](https://doi.org/10.1109/THMS.2014.2362529)
18. Guestrin C, Krause A, Singh AP (2005) Near-optimal sensor placements in gaussian processes. In: *Proceedings of the 22nd international conference on machine learning, ICML '05*. ACM, New York, pp 265–272. doi:[10.1145/1102351.1102385](https://doi.org/10.1145/1102351.1102385)
19. Hong J, Suh E, Kim S (2009) Context-aware systems: a literature review and classification. *Exp Syst Appl* 36(4):8509–8522. doi:[10.1016/j.eswa.2008.10.071](https://doi.org/10.1016/j.eswa.2008.10.071)
20. Katz S (1983) Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. *J Am Geriatr Soc* 31(12):721–727
21. Krause A, Guestrin C, Gupta A, Kleinberg J (2006) Near-optimal sensor placements: maximizing information while minimizing communication cost. In: *Proceedings of the 5th international conference on information processing in sensor networks, IPSN '06*. ACM, New York, pp 2–10. doi:[10.1145/1127777.1127782](https://doi.org/10.1145/1127777.1127782)
22. Krishnan NC, Cook DJ (2014) Activity recognition on streaming sensor data. *Perv Mob Comput* 10:138–154. doi:[10.1016/j.pmcj.2012.07.003](https://doi.org/10.1016/j.pmcj.2012.07.003)
23. Lifton J, Feldmeier M, Ono Y, Lewis C, Paradiso JA (2007) A platform for ubiquitous sensor deployment in occupational and domestic environments. In: *The international conference on information processing in sensor networks, IPSN '07*. ACM, New York, pp 119–127. doi:[10.1145/1236360.1236377](https://doi.org/10.1145/1236360.1236377)
24. Miorandi D, Sicari S, Pellegrini FD, Chlamtac I (2012) Internet of things: vision, applications and research challenges. *Ad Hoc Netw* 10(7):1497–1516. doi:[10.1016/j.adhoc.2012.02.016](https://doi.org/10.1016/j.adhoc.2012.02.016)
25. Mitchell M (1998) *An introduction to genetic algorithms (complex adaptive systems)*. A Bradford Book, Cambridge (1998)
26. Nott DJ, Dunsmuir WTM (2002) Estimation of nonstationary spatial covariance structure. *Biometrika* 89(4):819–829. doi:[10.1093/biomet/89.4.819](https://doi.org/10.1093/biomet/89.4.819)
27. Ranieri J, Chebira A, Vetterli M (2014) Near-optimal sensor placement for linear inverse problems. *IEEE Trans Signal Process* 62(5):1135–1146. doi:[10.1109/TSP.2014.2299518](https://doi.org/10.1109/TSP.2014.2299518)
28. Rasmussen CE (2004) Gaussian processes in machine learning. In: *Advanced lectures on machine learning. Lecture notes in computer science*, vol 3176. Springer, Berlin, pp 63–71. doi:[10.1007/978-3-540-28650-9_4](https://doi.org/10.1007/978-3-540-28650-9_4)
29. Schmitter-Edgecombe M, Parsey C, Cook DJ (2011) Cognitive correlates of functional performance in older adults: comparison of self-report, direct observation, and performance-based measures. *J Int Neuropsychol Soc* 17(5):1–12. doi:[10.1017/S1355617711000865](https://doi.org/10.1017/S1355617711000865)
30. Seelye AM, Schmitter-Edgecombe M, Cook DJ, Crandall A (2013) Naturalistic assessment of everyday activities and prompting technologies in mild cognitive impairment. *J Int Neuropsychol Soc* 19:442–452. doi:[10.1017/S135561771200149X](https://doi.org/10.1017/S135561771200149X)
31. Shermer TC (1992) Recent results in art galleries [geometry]. *Proc IEEE* 80(9):1384–1399. doi:[10.1109/5.163407](https://doi.org/10.1109/5.163407)
32. Singla G, Cook DJ (2009) Interleaved activity recognition for smart home residents. In: *Proceedings of the international conference on intelligent environments, IE '09*, vol 2. IOS Press, Amsterdam, pp 145–152. doi:[10.3233/978-1-60750-034-6-145](https://doi.org/10.3233/978-1-60750-034-6-145)
33. Singla G, Cook DJ, Schmitter-Edgecombe M (2009) Tracking activities in complex settings using smart environment technologies. *Int J BioSci Psychiatry Technol* 1(1):25–36
34. Singla G, Cook DJ, Schmitter-Edgecombe M (2010) Recognizing independent and joint activities among multiple residents in smart environments. *J Ambient Intell Human Comput* 1(1):57–63. doi:[10.1007/s12652-009-0007-1](https://doi.org/10.1007/s12652-009-0007-1)
35. WSU CASAS (2015) Center for Advanced Studies in Adaptive Systems (CASAS) shared smart home datasets repository. <http://casas.wsu.edu/datasets.html>
36. Zulas AL, Crandall AS (2014) Assessing professional caregiver needs in assistive smart homes. In: *Handbook of smart homes, health care and well-being*. Springer International Publishing, Berlin, pp 1–10. doi:[10.1007/978-3-319-01904-8_14-1](https://doi.org/10.1007/978-3-319-01904-8_14-1)
37. Zulas AL, Crandall AS, Schmitter-Edgecombe M (2014) Caregiver needs from elder care assistive smart homes: children of elder adults assessment. *Proc Human Factors Ergonom Soc Annu Meet* 58(1):634–638. doi:[10.1177/1541931214581150](https://doi.org/10.1177/1541931214581150)
38. Zulas AL, Crandall AS, Schmitter-Edgecombe M, Cook DJ (2012) Caregiver needs from elder care assistive smart homes: nursing assessment. *Proc Human Fact Ergonom Soc Annu Meet* 56(1):125–129. doi:[10.1177/1071181312561003](https://doi.org/10.1177/1071181312561003)