

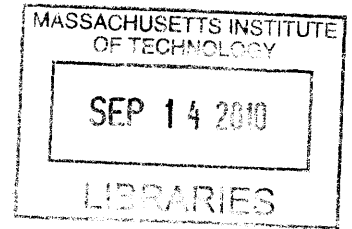
# End-User Modification and Correction of Home Activity Recognition

by

**Edward E. Burns**

B.A., Computer Science

Pomona College, 2004



**ARCHIVES**

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning,  
in partial fulfillment of the requirements for the degree of,

MASTER OF SCIENCE

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

Author

A large, stylized handwritten signature in black ink, written over a horizontal line.

Program in Media Arts and Sciences  
August 31, 2010

Certified by

A handwritten signature in black ink, written over a horizontal line.

Kent Larson  
Principal Research Scientist  
House\_n / Changing Places, MIT Department of Architecture  
Thesis supervisor

Accepted by

A handwritten signature in black ink, written over a horizontal line.

Pattie Maes  
Associate Academic Head  
Program in Media Arts and Sciences



# Abstract


Sensor-enabled computer systems capable of recognizing specific activities taking place in the home may enable a host of “context-aware” applications such as health monitoring, home automation, remote presence, and on-demand information and learning, among others. Current state-of-the-art systems can achieve close to 90% accuracy in certain situations, but the decision processes involved in this recognition are too complex for the end-users of the home to understand. Even at 90% accuracy, errors are inevitable and frequent, and when they do occur the end-users have no tools to understand the cause of errors or to correct them. Instead of such complex approaches, this work proposes and evaluates a simplified, user-centric activity recognition system that can be understood, modified, and improved by the occupants of a context-aware home. The system, named Distinguish, relies on high-level, common sense information to construct activity models used in recognition. These models are transferrable between homes and can be modified on a mobile phone-sized screen. Observations are reported from a pilot evaluation of Distinguish on naturalistic data gathered continuously from an instrumented home over a period of a month. Without any knowledge of the target home or its occupant’s behaviors and no training data other than common sense information contributed by web users, the system achieved a baseline activity recognition accuracy of 20% with 51 target activities. A user test with 10 participants demonstrated that end-users were able to not only understand the cause of the errors, but with a few minutes of effort were also able to improve the system’s accuracy in recognizing a particular activity from 12.5% to 52.3%. Based on the user study, 5 design recommendations are presented.

Thesis Supervisor:  
Kent Larson  
Principal Research Scientist, House\_n/Changing Places  
Department of Architecture

# End-User Modification and Correction of Home Activity Recognition

by  
**Edward E. Burns**

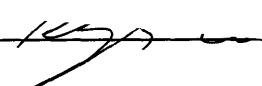
Thesis advisor



---

**Kent Larson**  
Principal Research Scientist  
House\_n / Changing Places, MIT Department of Architecture

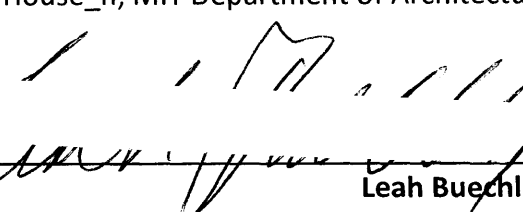
Thesis reader



---

**Stephen Intille**  
Research Scientist  
House\_n, MIT Department of Architecture

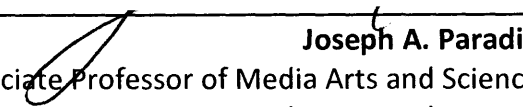
Thesis reader



---

**Leah Buechley**  
Assistant Professor of Media Arts and Sciences  
Program in Media Arts and Sciences

Thesis reader



---

**Joseph A. Paradiso**  
Associate Professor of Media Arts and Sciences  
Program in Media Arts and Sciences



# Acknowledgements

There are many people to whom I owe much thanks and credit. I am greatly indebted to both Kent Larson and Stephen Intille for their support, patient advice, and always-constructive critiques. Without them this work would be less practical, more confusing, and much less impressive. Their advice was the necessary catalyst to break out of many mental blockades, and their constant support was invaluable. Additional great thanks to my readers, Leah Beuchley and Joe Paradiso, whose patience and flexibility were invaluable and whose guidance helped create a much better final product; to Jason Nawyn for providing technical support, advice, critical research data, and the greatest of personal support; to my lab mates Yi Han, Jonathan Lester, Selene Mota, and Fahd Albinali for their kind words, encouragement, and well-considered advice; and to Linda Peterson and Aaron Solle, without whose tireless efforts and patience this thesis would not exist.

I would like to thank my family and Amelia Yu for their never-flagging support and encouragement, and for believing in me even when I did not.

This work is supported by a generous grant from Intel.

## Table of Contents

Abstract .....	3
Introduction .....	8
Problem 1: Dependency on labeled training data .....	8
Problem 2: Size of problem's solution space .....	8
Problem 3: Opaque failure modes .....	9
User-centered design .....	9
Background .....	11
Approach .....	13
Algorithm design .....	15
Data abstraction .....	18
Design limitations .....	19
Generating the common-sense databases .....	20
Modification interface .....	21
Design discussion .....	23
Evaluation and Results .....	25
Common sense survey .....	25
Database construction .....	26
Base recognition results .....	27
Usability study .....	30
Study results .....	33
Quantitative results .....	33
Participant behavior and feedback .....	34
Idealists vs. puzzle solvers .....	34
Oscillating errors .....	34
Strategies and brevity of action .....	35
Relationship to real data .....	36
Data overload .....	37
Extending the system .....	37
Data magnitudes .....	38

Discussion ..... 39

    Speed of correction ..... 41

    Recommendations..... 42

    Limitations of the study ..... 45

Conclusion ..... 46

Bibliography ..... 48

Appendix..... 51

    Appendix A – User interface prototypes ..... 51

        Preliminary survey – free-form text entry ..... 51

        Pick-and-choose survey with specific types of data ..... 52

        Refined pick-and-choose survey with a focus on objects ..... 53

        Final survey ..... 55

    Appendix B – All activities used in survey (total: 51) ..... 56

    Appendix C – All object questions used in survey (total = 108) ..... 57

    Appendix D – Total annotated activities from the source home ..... 61

    Appendix E – User study tutorial script ..... 63

    Appendix F – Study debriefing questions ..... 68

    Appendix G – Study advertisement ..... 68

    Appendix H – Confusion Matrix ..... 69

# Introduction

Activity recognition systems attempt to automatically identify human behavior taking place within a particular space (frequently the home or the office) using sensor data. The nature and granularity of the target activities depend on the application, and may vary from the specific and/or personal (“making my morning cup of chamomile tea”) to the general (“sleeping”). The sensing technology used to perform classification varies wildly, and may include cameras [1], motion sensors [2], wireless location tracking systems [3], microphones, object usage sensors [4], on-body sensors, and electricity and utility sensors [5], among others.

Accurate, practical, and robust activity recognition would make a variety of “context-aware computing” applications feasible. These applications include health monitoring of the elderly through activities of daily living (ADLs) [6], advanced energy optimizations through smart climate and lighting controls, on-demand educational and assistive technologies, and remote-presence and ubiquitous communication systems.

Most recent research on activity recognition has focused on increasing recognition performance. However, even the best-performing systems achieve less than 90% overall accuracy, often with poor performance on some activities of interest and a limited list of possible activities (frequently fewer than 15). The question of what to do in the case of recognition failure, and what this means for the end-user, has not been fully addressed [7]. In addition, there are three main challenges to applying traditional machine learning activity recognition in the home domain, detailed below.

## **Problem 1: Dependency on labeled training data**

Machine-learning techniques require large training sets in order to achieve acceptable performance (for general-purpose recognition, this is on the order of weeks or months of data). In this domain this translates into many weeks of hand-annotated activity data. This annotation can be done either by the occupant of the space [8], an onerous task that few are willing to devote time to, or by a third party, who must be paid for many hours of work. Differences in sensing technologies, home layouts and the locations of sensors within them, the number and type of occupants, and behavior between end-users limits data transferability between environments. This suggests that large corpora of annotated data would have to be created *for each household*. Given the difficulty in obtaining training data, this approach does not appear to scale beyond a handful of experimental homes. Some preliminary work has been done with information-transfer between domains [9, 10], but is still currently limited in its scope and application.

## **Problem 2: Size of problem’s solution space**

Sensor implementations vary between approaches, but some fully instrumented homes contain upwards of 200 sensors [4]. These sensors may be a mixture of different types, from object usage sensors to motion sensors to plumbing flow meters. Incorporating

readings from all of these sensors can quickly lead to a “curse of dimensionality” in traditional machine learning approaches. As the size of the state vector increases, so too does the amount of training data required to achieve acceptable recognition performance. Logan et al. [11] encountered such a problem when analyzing data from the PlaceLab [4] – when excluding roughly half of the involved sensors, recognition performance *increased*. This creates a situation for the future homeowner where adding additional sensors may actually degrade performance, rather than improve it. This problem is further complicated by the choice of activity taxonomy. Changes to the sensor layout or taxonomy (adding or removing activities, splitting activities into sub-activities), would require new training data as well as a lengthy retraining period. If the end-user wants to add a new activity (perhaps something important to them that the normal taxonomy does not capture), they must somehow acquire or generate new training data and then supervise the system’s training process.

### **Problem 3: Opaque failure modes**

With any recognition system, failure is inevitable. Current state-of-the-art systems have yet to reliably break above 90% accuracy on naturalistic data [8, 12, 13], so even if such systems re-label activities once every minute, they would commit an error every 10 minutes. Unfortunately, “soft failure” modes are hard to achieve with traditional machine learning due to the black-box nature of the recognition process. Once trained, such systems are hard even for their creators to debug, and they are completely opaque to the average end-user. Thus, when failure occurs the user will have no way of understanding why it occurred and no real way of correcting it. He or she might provide additional training data, but without understanding how the data are being processed, users are likely to supply noisy or otherwise unhelpful information and continue to receive poor recognition performance. A user’s remaining two options are equally unattractive: try to ignore the errors, or simply turn the system off.

### **User-centered design**

Given these concerns, especially the problem of mysterious failure, this work proposes and evaluates an alternative, user-centered approach to activity recognition. Others have already argued for the involvement of the user in context-aware applications<sup>1</sup> [14-16]. Dey et al. [17] enumerated two requirements for such applications:

- **Intelligibility** – the ability for users to *understand* how a system is currently functioning.
- **Control** – the ability for users to *modify* how the system functions.

This work applies these concepts to home activity recognition and adds a third requirement:

- **Iterability** – most changes by users demonstrably improve the functioning of the system. Improvements to one area rarely degrade performance in other areas, or if so these changes can be easily detected and corrected.

---

<sup>1</sup> Context-aware applications may depend upon activity labels generated by activity recognition systems, but context-aware application evaluation is a distinct field.

An activity recognition system, named Distinguish, was constructed to evaluate the proposed approach. In the Distinguish system, the recognition process has been simplified to such a degree that the end-user can *understand* and *modify* it. Because the cause of errors is transparent to the user, frustration may be less likely. In addition, the user can take corrective measures to improve performance by modifying his or her own sensor layout (e.g. adding missing sensors, moving poorly placed sensors) or by modifying the recognition system so that it no longer generates the error. The user may decide that the problem is not fixable, and disable whatever automated responses depend upon it. These options are only possible when the user is cognizant of how the recognition system actually works. Without such involvement, the user is helpless to do anything except hope the system improves later on or switch the system off completely.

There are three major sources of errors in a recognition system: (1) noise in the sensor input, (2) weak or inappropriate models of the activities, and (3) exceptional or out-of-the-ordinary behavior. In the proposed system, all three types of errors are exposed directly to the user, allowing them to take appropriate corrective action. In the first case, sensor information is abstracted from raw data into high-level labels that are meaningful to end-users, allowing them to understand when sensors are providing noisy or otherwise inaccurate information. The latter two types of error are addressed by employing a voting-based common-sense engine to perform recognition. The models of activity used by this system are designed to be simple enough for end-users to understand and modify. Thus, in the case of error, users can analyze the source of error (sensors vs. recognition) and, if desired, correct it. The resulting system has the following characteristics:

- The recognition process is *comprehensible* to end-users. Users can isolate whether the system is failing because of noisy or inadequate sensor input or inadequate activity models.
- The recognition process is *modifiable* by end-users.
- Recognition can be viewed and modified on a phone-sized interface, allowing for truly mobile and ubiquitous home control.
- “Training data” are stored in a generic, high-level format that is portable between homes and homeowners.
- The curse of dimensionality is mitigated – the system is able to benefit from the inclusion of additional sensors. Additional “training data” is only occasionally required, and can be generated in a couple of minutes.
- The system is *extensible* – end-users can create new activities for the system to recognize. In addition, they can add additional sensors with relatively little effort.
- The system is *reflective* – it is possible for the system to self-analyze. Given a set of training data and a sensing environment (a particular home), the system can inform the user if some activities are indistinguishable from other activities, or if they are impossible to detect altogether due to a lack of appropriate sensors.

# Background

There has been extensive work with using SVMs [18, 19], decision trees [11], CRFs [8], a variety of probabilistic models [8, 12, 13, 20, 21], as well as number of rule-based approaches [22, 23] to perform activity recognition in the home. As has been discussed, the overall best performance of most of these approaches remains below 90%, even when applied to simplified data or a when asked to recognize a small subset of all common home activities. Even if systems could reliably achieve 95% accuracy, errors would occur frequently. These approaches do not currently expose any means of "explanation" to the end-user for their mistakes or provide any mechanisms for users to fix mistakes other than adding more training data, which as discussed above is unrealistic in many cases. In a real deployment, user frustration would be likely [7].

There has been some work that attempts to involve the end-user in the recognition process. Kasteren et al. [8] created a system that allowed occupants of a space to vocally annotate their activities as they performed them, although without the ability to understand or control how this information was being used. Kawsar et al. [24] explored end-user modification of their sensor network, but without completely explaining the operational concerns of the sensors or how their data would be processed to the end-users. As a result, much end-user involvement devolved into reading manuals specifying certain required types of sensors, and following instructions for their placement. It is unlikely that end-users would be able to deduce the cause of sensor error if it was present, or know how to correct it. Some work in image recognition has explored a "guided" system for helping the end-user to supply helpful data [25], but such approaches have yet to be applied to the temporally complex domain of home activities. Single images are much easier to display and categorize than complex actions that take place over a period of time, which may be interleaved with other, unrelated activities.

While end-user involvement in activity recognition remains a relatively new field, there has been considerable research in end-user *programming* of smart environments. MicroCommander [26] demonstrated end-user control of home behavior using simulated logic gates. CAMP [27] and the Jigsaw Editor [28] were a pair of more user-friendly interfaces with novel fridge poetry and jigsaw puzzle metaphors. These systems employed a trigger/action approach that directly linked raw sensor inputs to application responses. This approach is simple for the end-user to understand, but prevents him or her from constructing behaviors triggered by higher-level concepts such as activities. Commercial systems such as X10 [29] and Mr. House [30] can also be setup using a trigger/action metaphor but have failed to gain widespread adoption. When such systems misbehave, their users frequently call tech support instead of correcting the problems themselves [31].

More recently, aCAPpella [32] provided a bridge between activity detection and behavior, allowing end-users to define general categories such as "having a meeting," record demonstrations of the activity, and then link certain behaviors to this activity. However, the

actual recognition process was performed automatically, and without providing the user with an understanding of how it was taking place. End-users were able to label sections of their demonstration as particularly relevant, but it is unclear how they would have been able to determine what data would be most relevant to the training process. Exemplar [33] allowed a similar process where users demonstrated an activity, edited the demonstration to remove irrelevant data, and then reviewed the results of their changes. However, users had to interact with their recordings on the raw sensor level, selecting "good" data from "bad" data on the grounds of abstract line graphs. Again, it is unclear how the end-users were expected to make these decisions. Ubicorder [34] was a diagnostic tool for debugging context-aware environments targeted at trained professionals - its novel interface is extremely powerful, but not appropriate for end-user control due to its complexity. Finally, Activity Designer [35] provided a comprehensive test bed for defining activities, designing behavior around them, and field-testing the resulting system. This approach is appealing because it allows for in-situ debugging and modification as well as high-level definitions of activities, but is targeted primarily at professional designers rather than end-users. The resulting system, while quite powerful, is probably too complex for the average end-user.



# Approach

Most activity recognition systems function on the *sensor level* – raw data is read from sensors and are converted into a set of *features*. These features are then processed by the classifier in order to generate one or more activity labels. The choice of what features are used and how they are generated is an important one. Such choices allow the architect of the system to attempt to inject some representation of higher-level information into the system (for example, a “sittable object has been moved in the last five minutes” feature instead of raw data from chair sensors). However, such features are usually still too complex, or too closely related to the raw data, to be explicable to end-users. This approach affords great accuracy, flexibility, and adaptability, but it requires large training sets and is not comprehensible to the end-user. Raw sensor readings are difficult to interpret, and the black-box reasoning of machine learning is difficult for even pattern recognition experts to analyze, much less end-users.

Instead, Distinguish raises the recognition algorithm from the sensor level to the *user level*. Involving the user solves many outstanding problems with activity recognition, but also imposes a great number of restrictions on the design of the algorithm. First, incoming raw data must be abstracted into a form that is comprehensible to the end-user. Second, the complex mathematics of traditional approaches such as HMMs and decision trees are no longer appropriate. These approaches frequently require graduate-level understandings of computer science and mathematics, involve hundreds of operations for each classification, and are difficult to visualize.

Previously, we have dwelt only on the problems that the home environment creates for recognition algorithms. However, at the same time, the semantic richness of the home provides much more contextual information than might be found in other applications of pattern recognition, such as speech recognition or financial analysis. Most homes are carefully broken up into sections with specific functions (kitchen, bedroom, bathroom). Similarly, the objects they contain are heavy with meaning: what they can be used for, whether they can be moved around, etc. Some activities only take place in certain rooms, others only happen at a certain time of day, while others can only occur until some prerequisite is met (for example, you can’t do the dishes until you’ve made dinner). In theory, such associations should be learnable by standard machine learning algorithms, but in practice the amount of data required to separate the signal from the noise becomes prohibitive. Suppose one person in one home always waters his plants while cooking dinner. Is this a desirable association to learn? What happens when the person forgets to water his plants one night and the system fails to recognize meal preparation? Would it make sense to the user that watering plants and making a meal should be dependent?

It is easy – too easy – to construct scenarios where systems using large training sets in the home learn statistical associations that may work for a while, then break, and then cause intense user frustration because what the system learns does not map onto the user’s mental model of what should be taking place. Instead, a system that allows the user to simply tell it such associations (ironing uses the iron, etc.) would be able to more

accurately capture the user's mental model of what is taking place while avoiding a dependency on statistical quirks that may not generalize to all normal behaviors.

Distinguish uses a common-sense database of home object and activity information in order to perform recognition. This allows us access to the contextual information of the home without the need for large training corpora. The contents of this database are fully modifiable by the end-user, allowing him or her to construct specific or personal definitions if desired (such as the plant watering example).

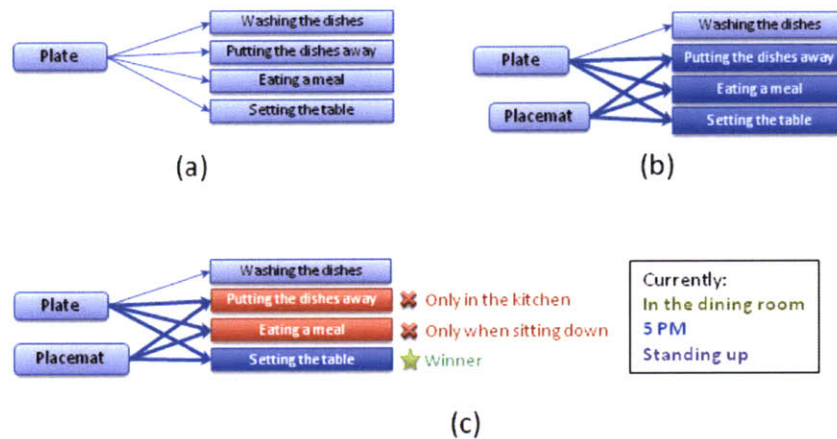
Distinguish was designed to reduce the mental overhead involved in modifying the system as much as possible. In this manner, both the algorithm and the interface have been designed to analyze only the most recent and directly applicable data as follows:

- Primary recognition involves only data from object-usage sensors. This allows the end-user to focus his/her attention on a single source of data. Object usage information was chosen above other data sources due to its specificity of meaning – many objects in the home are involved in only a handful of specific activities, while other sources of information such as current location or the time of day are either less specific (“standing up” could indicate hundreds of possible activities) or much harder to process (a video stream might contain extremely high-fidelity information, but cannot be easily analyzed with modern computer vision algorithms). These other sensor data streams (e.g., user's location inferred from motion sensors and on-body location sensors if present, posture inferred from on-body accelerometers) are used in later stages to improve these results by removing illogical decisions.
- The recognition process is broken up by the rooms in the home, so that recognition taking place in the kitchen is distinct from the one taking place in the living room. These processes are independent and do not affect one another. This reduces the problem space for a user attempting to correct the system – they need only think about data gathered from the room they are currently occupying as opposed from the house in general.
- Recognition is performed on the most recent 5 minutes of data only. The borders of this window are strict – data is either present in the window or it is not. This avoids situations where the user may become confused as to what data the system is using to make its decision, or why some recent data is included but other data is not. Without such knowledge, it is difficult to effectively correct the system, or detect the presence of noisy data that are confusing the recognition algorithm.
- “Fuzzy” weighting metrics are not employed. Weighting metrics further hide from the user exactly how the data is being processed and used by the system. Why does one piece of data have a much greater influence over the results than another although they both occurred around the same time? What is the result if I change the weight of some data point from 0.2 to 0.3? Such weights are difficult to understand and are difficult to relate to real life (Questions such as “Should plate have a weight of 0.53 or 0.66 when associated with *doing the dishes?*” are meaningless to end-users).

## Algorithm design

A standard approach for performing pattern recognition involves generating a specific model for each pattern desired (in this case, different types of activities). When performing activity recognition, each model is matched against recent data and the model that matches the “best” is selected (for some given definition of “best”). This approach is common in many forms of pattern recognition, including dynamic graphical models (e.g., HMMs) and rule-based systems. However, as the accuracy of these models increases, so do their complexities. For example, consider a list of all the possible objects that might be included in the “cleaning up” activity: broom, mop, paper towel, Windex, dust pan, rag, garbage bag, sponge, feather duster, and faucet, among others. A single instance of “cleaning up” might only involve one or two of these objects, but a complete model should include them all (and quite possibly many more, such as any objects that someone might want to actually clean). When evaluating these models against incoming data, the system must process each lengthy model in turn. This is usually of little concern for modern-day processors, but the resulting chain of logic becomes an onerous one for the end-user to understand or debug. In order to “follow along”, they may be forced to make hundreds of calculations or comparisons, and the majority of these comparisons will most likely have no bearing on the actual situation and simply confuse the user. For example, the user might wonder, “Why is it worrying about ‘mop’ when I wasn’t anywhere near a mop?”.

By contrast, the design of Distinguish’s algorithm seeks to more closely imitate the decision process that a human might go through if asked to label activity data. A single piece of data might be evidence for a number of different activities (Fig. 1a). However, as more data are gathered, a consensus may begin to form (Fig. 1b). Finally, if additional information is known about the state of the environment (the current time, the current room, etc.), an even more accurate guess can be made (Fig. 1c).



**Fig. 1: Decision model.** (a) Plate is associated with a number of possible activities. (b) Plate and placemat are both associated with three of these four activities, suggesting that they are more likely to be taking place. (c) Two of the remaining three activities can be ruled out through the use of other information, leaving only one left to choose.

Distinguish attempts to create a recognition model that may mirror the user’s common sense, object-based mental model of the situation by reasoning via *data voting rounds*. During recognition, each recent piece of object usage data “votes” for the activities that it might indicate. For example, a recent “plate” object activation might vote for *setting the*

*table, eating a meal, and doing the dishes.* The results of this vote are tallied and the possible activities are ordered by their score. Finally, nonsensical candidates are pruned using other information about the current environment, including the current room, time of day, etc. For example, “taking a shower” would be pruned if the user were currently located in the kitchen.<sup>2</sup>

The specific steps that the algorithm takes are:

1. All object activations within the time window “vote” for any activities that might have caused them.
2. The activity with the most votes “wins” and all data that voted for the winning activity is removed from the voting pool.
3. If there is still data remaining in the pool, another voting round takes place.
4. After each vote, nonsensical choices (such as taking a shower in the kitchen) are filtered out using rules generated from the common-sense database.

In this approach, only the most recent and applicable information is used in order to perform recognition (i.e. recent data and all activities that might be associated with that data). Note that, through the use of multiple rounds, this system is capable of detecting simultaneous and interleaved activities. However, it has no knowledge of the number of current occupants of a space, and thus cannot take into account possible activity exclusions (e.g. if only one person is present, they cannot be both taking a shower and cooking a meal). In addition, because a data point can only “vote” for a winner once before it is removed from the voting pool, this reduces the system’s ability to detect overlapping activities that involve the same object (such as washing the dishes and then putting them away in a cabinet).

Distinguish uses two common-sense databases to perform recognition: *object associations* and *activity restrictions*. These databases were not hand-coded, but were instead generated from the results of a web survey (described shortly in “Generating the databases”). In this way, end-users are involved in the complete construction of the system, from rule generation to rule modification. The **object associations** database is a simple mapping of *object* → *possible activities*. For example, one entry might be:

*hand soap* → *washing hands, taking a shower*

These activities are not ordered or ranked. The **activity restrictions** database consists of a collection of common-sense restrictions for each activity. Distinguish currently uses three types of restrictions: the rooms that an activity can take place in (*where*), the times of day that it can occur (*when*), and the postures involved in performing the activity (*how*). Additional types of restrictions could be added without modifying the algorithm (seasons, activities that must have occurred previously, etc.). Thus, restrictions are a form of common-sense rule, but must remain straightforward enough for end-users to understand.

---

<sup>2</sup> The question of whether this approach is, in fact, an intuitive one for the end-user was evaluated during user testing (see: Evaluation section).



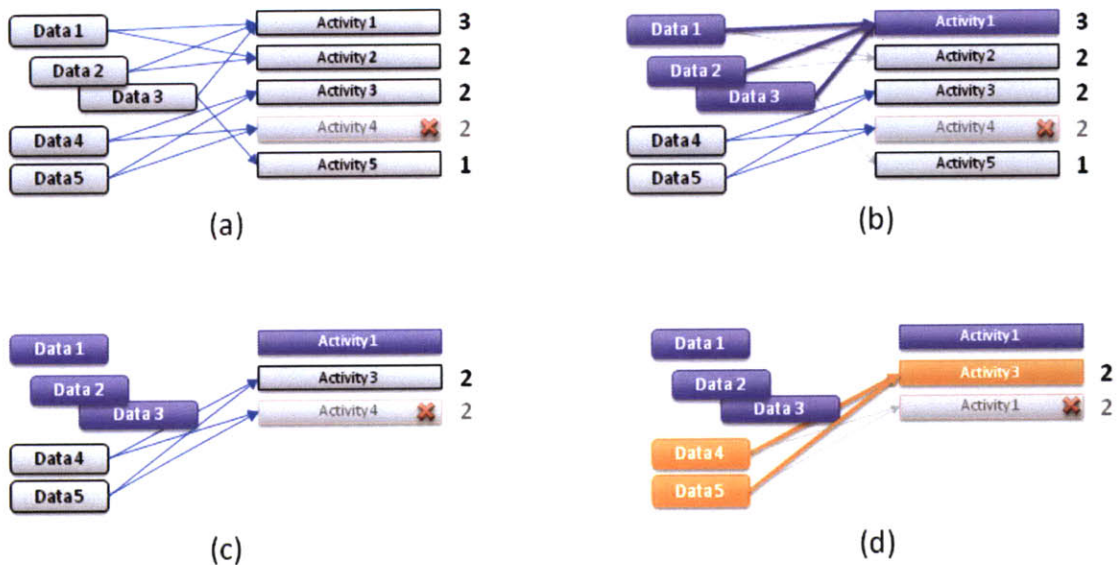
Abstract terminology, chained reasoning, or other complex interactions between restrictions are not appropriate.

Restriction type	Possible values
Room	Bathroom, bedroom, dining room, living room, kitchen, office, entrance/exit
Time	Early morning, morning, noon, afternoon, evening, night, wee hours
Posture	Standing, sitting, crouching, lying down

**Table 1: Possible restriction values. Raw sensor values are never displayed to the user. Instead, they are encapsulated in simple semantic labels. Instead of listing the time as “2:27 PM”, it is simply “afternoon”.**

Each restriction can contain one or more allowed values chosen from a set of abstract labels (see Table 1). During recognition, these labels are matched against the **current state** of the environment, and any candidate activity that violates one or more of its activity restrictions is discarded from the recognition process.

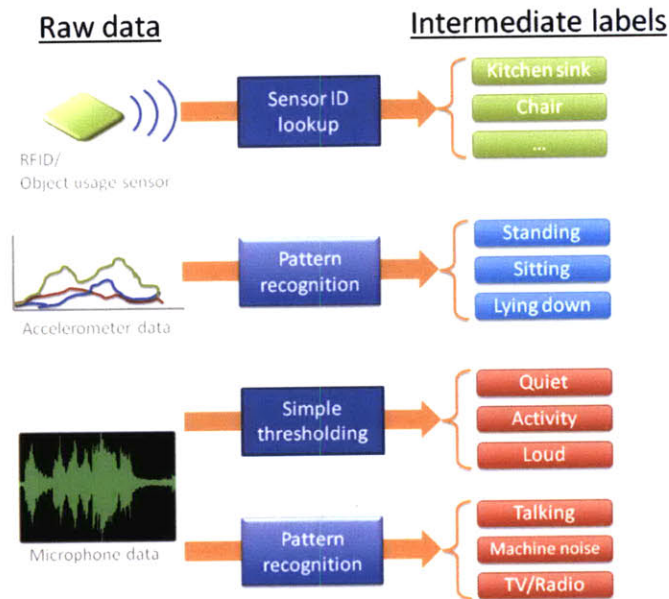
An example recognition attempt is presented in Fig. 2. First, recently-moved objects vote for their related activities and a ranked list of candidates is generated based on the number of votes that each activity received (a). Some candidates may be excluded from the process because one or more of their restrictions conflicts with the current environment (in the example, Activity 4 is in violation). A winner is chosen (b), and all data that voted for it are removed (c), preparing the system for another vote (d).



**Fig. 2: Multiple voting rounds.** (a) Data votes for all the activities they might indicate. Some activities are eliminated because they violate a common-sense rule (in this case, Activity 4). (b) The activity with the highest score is chosen as the winner (purple). (c) Data that voted for the winner are removed from the voting pool. (d) Any remaining data participate in another round of voting.

## Data abstraction

Both the recent object activations as well as the state labels making up the current environment are the result of a data abstraction process. Distinguish abstracts raw sensor data to the user level by condensing it into semantically rich labels. Unlike raw sensor data, these labels carry recognizable meaning for the end user (for example, a “loud” label for microphone input instead of a raw decibel reading). Depending on the type of sensor involved, this labeling process may involve simple thresholding or renaming techniques, or it may require its own specialized pattern recognition system. For example, data from object usage sensors is only marginally preprocessed by the system: the sensor ID is replaced with the name of the object that the sensor was attached to, and excess duplicate firings are pruned to only occur once every few seconds. Other, more complex, data inputs may require additional preprocessing. For example, posture information from on-body accelerometers might be transformed into posture labels by a carefully trained machine learning system. Microphone input might be transformed into simple “quiet” and “loud” labels, or a dynamic graphical model might be employed to recognize the difference between music, conversation, machinery noise, and general background noise (see Fig. 3).



**Fig. 3 – Example data sources, abstraction providers, and intermediate labels.** Some raw data are transformed using relatively simple processes, while other abstraction providers may utilize more sophisticated techniques to provide semantic labels.

This approach brings with it a number of advantages. First, because the data labels exist on a much higher semantic level, they are frequently sensor-agnostic. Object usage information might be provided by on-object sensors, electrical grid monitors, computer vision systems, or a combination of the three. Recognition will be unaffected, beyond the differences in quality of data from these sources. As such, systems that are “trained” at this

high level of abstraction are much more portable between homes. Second, it allows the user to detect the difference between recognition failures and sensing failures. Detecting malfunctions from raw sensor data is difficult unless the user is familiar with the normal operating ranges of such sensors, but incorrect semantic labels are easier to spot (“ah, it didn’t work because it thought that I was standing up when I was actually sitting down”).

The Distinguish system’s primary source of data is from object-usage sensors. During evaluation, the system was tested with data gathered from MITes object sensors [4], which use a combination of piezo sensors and accelerometers to detect motion. In this case, the abstraction process was spread between the firmware running on the sensors and the Distinguish recognition software. Firmware on the sensors first converted raw sensor output into distinct “firings” based on a single, experimentally-determined thresholding system. The Distinguish abstraction system then converted raw sensor IDs to human-readable descriptions of the objects and pruned out duplicate firings. The design of these thresholding systems can become arbitrarily complex, and is beyond the scope of this work. Mittal’s Ubicorder [34] represents promising work in providing system designers with the tools to dependably construct and optimize such abstraction systems.

### Design limitations

The involvement of the end-user in the recognition process places limits on the complexity and expressivity of the system. There are many features and additional pieces of information that one might imagine adding to the system in order to improve performance, but doing so is not always possible without destroying the end-user’s ability to understand what is going on. In order for the user to be truly in control of the recognition process, he or she must be able to:

- Easily “read” the current state of the system (what sensors have fired, what other data is being considered)
- Understand why the current activity labels have been chosen for this state
- Modify the results of this labeling to some other desired set of labels, or to understand why that is not possible

These goals present a number of restrictions on the design of the recognition algorithm:

1. **No raw sensor data.** Data taken directly from sensors is difficult to interpret unless the end-user has intimate knowledge of the technical design of each sensor type involved. This does not mean that users should be unaware of the technical limitations of sensing, but that these limitations should be presented in a way that is meaningful to them.
2. **Clearly-defined state.** The “current state” of the system should have clearly-defined boundaries, both in space and time. In other words, it should be clear to the user which data are being used to make the decision and which data are not. This precludes many “fuzzy” time window approaches where the system looks further back in time during some situations but not others. For example, the system cannot adjust its time window if it detects that a long-duration activity is taking place.

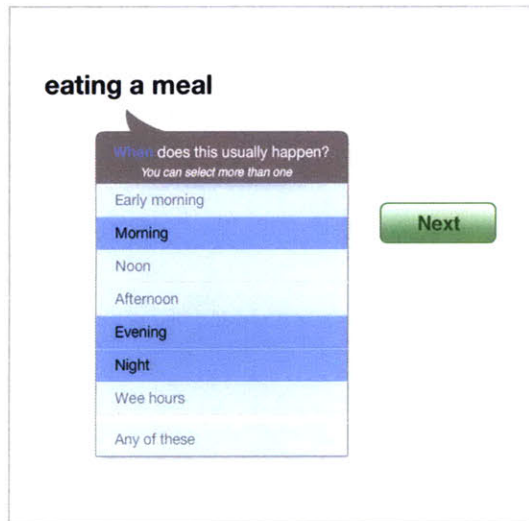
3. **Simple math & whole numbers.** In order for the user to truly understand the recognition process, he/she should be able to manually calculate the result him or herself without having to pull out a calculator. This limits most processes to arithmetic over whole numbers (no fractions, no decimals).
4. **No probabilistic reasoning.** In order for users to comfortably modify the behavior of the system, they must be able to predict what effect their actions will have. If randomness is involved in the decision-making process, this “predictability” is eroded, along with user confidence.
5. **Mobile.** The system must be available for modification as the user goes about his/her daily routine. When error occurs, the user should be able to modify the system immediately, with the memory of the actions they performed still fresh in their mind. For this study, the system was designed to run on a mobile phone. This significantly restricts the complexity of the UI available to the user.

These limitations do not come without costs. The loss of probabilistic reasoning removes the ability to capture frequency-of-use information and prevents users from setting preferences or weights for certain objects or activities. The use of the 5-minute window prohibits the system from recognizing long-term activities that do not generate data for long periods of time (for example, when watching a movie, the occupants of a space remain relatively still and may not activate any sensors for an hour or more). The mobile requirement also greatly reduces the complexity of computation available to the system: not only must it be computationally light enough to function on the limited processing capabilities of a mobile device, it must also be visually simple so that it can be displayed graphically on a small screen and touch screen interaction is sufficient. These costs are not inconsequential, but are compensated for by the involvement of the end-user. See the “Design discussion” section at the end of this chapter for a more thorough analysis.

### **Generating the common-sense databases**

The two common sense databases used by Distinguish were generated from the aggregate results of a web survey. No rules were handcrafted or tuned by the investigators, a common criticism of rule-based systems. Participants were asked two types of questions: activity restriction questions (Fig. 4) and object association questions (Fig. 5). Activity restriction questions were all of a simple “multiple choice” format with pre-defined options. These questions asked the participants to specify the allowed rooms, times of day, and postures for a particular activity. Object association questions were of the format “When do you use or touch an <object>?” Here, participants were provided with a pre-defined list of activities easily searched using a smart list, but were also allowed to write in custom activities if so desired.





**Fig. 4: Restriction question.** This particular question is asking about the times of day that this activity occurs.



**Fig. 5: Object usage question.** In this case, the participant has associated *chair* with *eating a meal*, *studying*, and *working*.

## Modification interface

Distinguish uses object associations and activity restrictions generated from aggregate user input (the aforementioned web survey). However, these definitions represent the most common way of doing things in the most common household layout. These definitions allow the system to make “acceptable” decisions out-of-the-box, but end-users may wish to adapt them to the specifics of their home and the way they do things.

Distinguish’s user interface allows its users to view recent (abstracted) sensor data, the activities that have been inferred from this data, and a graphical representation of the decision process that generated these labels. Users are able to modify the recognition process directly and view the results of their changes in real time. In order to fulfill the “mobile” user experience requirement, Distinguish’s user interface was designed to fit on a typical smartphone screen (480 x 320 pixels).

The interface’s default view displays only recent data (left column) and the labels generated from the most recent vote (right column) (see Fig. 6). New data “blocks” appear at the top of the screen and slowly fall down as time passes, similar to a game of Tetris. When blocks are older than 5 minutes, they pass off the bottom edge of the screen.

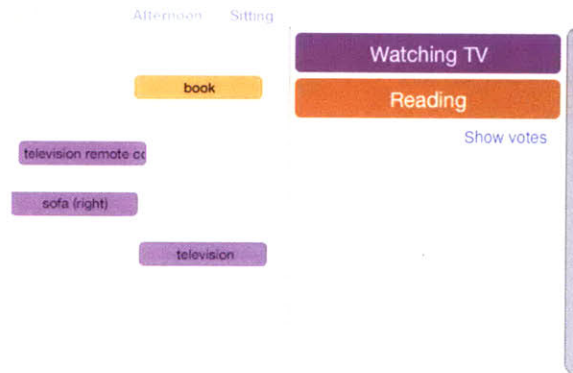


Fig. 6 – Recognition summary. The last five minutes of data appears on the left. Recent recognized activities appear on the right. As time passes, data blocks on the left drift downwards until they fall off the bottom of the screen.

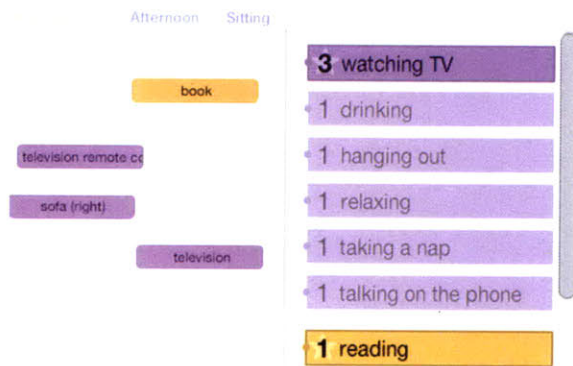


Fig. 7 – All possible activities. If the user requests that the system “show votes,” all possible activities are displayed on the right side of the screen. In this example, *watching TV* was chosen over other activities such as *drinking* and *relaxing* because it has a higher score. *Reading*, which involved an unrelated piece of data, was also chosen.

The user can request to view a representation of the most recent voting process by selecting “Show votes” (Fig. 7). This replaces the list of winning labels from each round with the complete list of possible candidates and their scores.

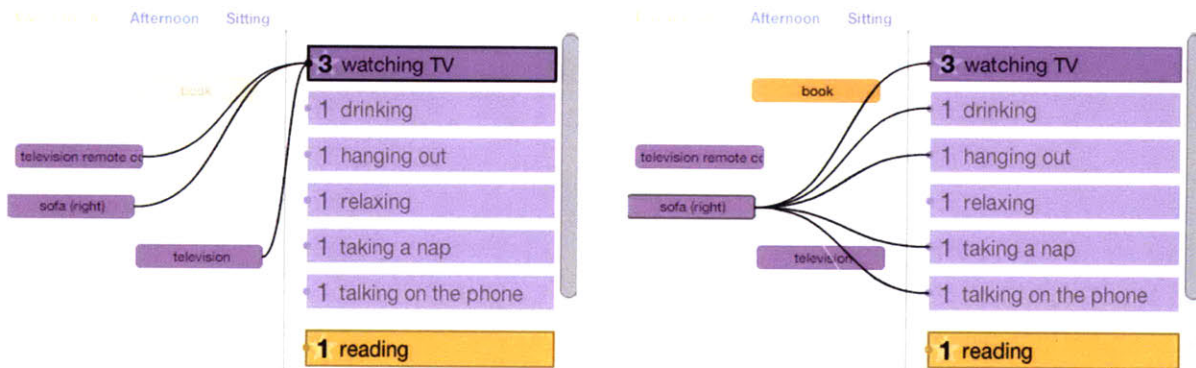


Fig. 8 – Object association lines. Users can choose to view all the objects currently voting for a particular activity (left), or all the activities that an object is voting for (right).

Users can modify the current recognition results in two ways: first, by modifying the votes involved, and second by modifying the restrictions applied to certain activities. Votes are represented as simple lines that connect data to activity candidates (Fig. 8). Users can

easily create lines by dragging between blocks, or remove them by tapping on existing lines. Candidate activities whose restrictions are violated are represented as being struck-through (Fig. 9). When a user taps on a candidate activity, the state of its three restrictions is displayed (Fig. 10). In the example given, the “room” restriction is in violation, while the “time” and “posture” restrictions are satisfied. Users may tap on these icons to modify the restrictions using a simple allowed/not allowed interface (Fig. 11).

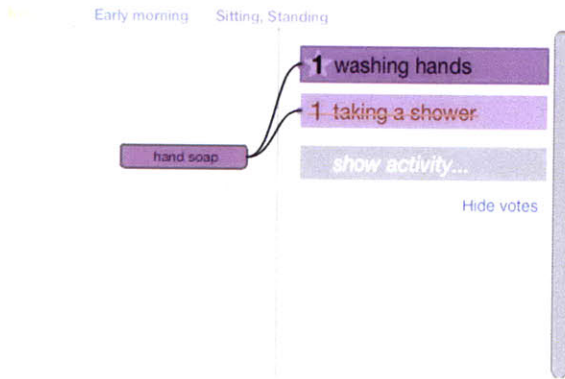


Fig. 9 – An invalid activity

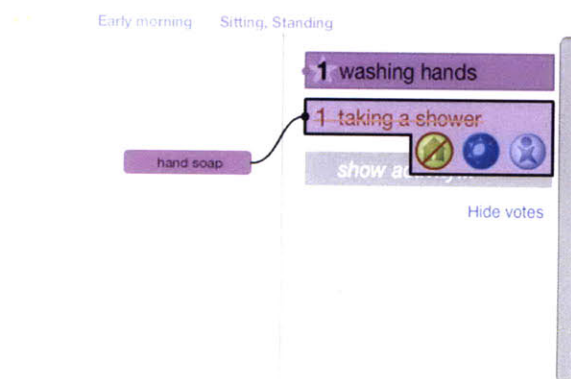


Fig. 10 – Current status of restrictions



Fig. 11 – Restriction modification interface. The current active values (here, the current room) are highlighted in yellow. Users can drag values from one circle to the other.

The Distinguish UI was developed in Flash and runs on all platforms that support the Adobe Flash player. It currently serves as a prototype for a future, native, phone application.

### Design discussion

Distinguish is essentially a rule-based system for performing recognition, similar in basic design to many common-sense reasoning systems. Rule-based approaches are well-known for struggling with a number of issues related to their procedural design: rules generated for such systems are frequently brittle and sensitive to small changes in the environment, at which point they will cease to function properly. In addition, the choice of knowledge representation used in the rules may be inappropriate for the situation, or may not be sufficiently expressive to perform robust recognition. Finally, in practice, the behavior of such systems can be almost as difficult to decipher as probabilistic machine learning

approaches, especially ones that involve the repeated application of rules in long inference chains in order to generate a final answer. Rules interact with each other in unexpected ways, and changes to one rule might cause the system's performance to suffer with seemingly unrelated activities. Finally, rule-based systems do not scale well to large numbers of rules: as rules are added, so do the possible (perhaps unintended) interactions between rules. In this manner, adding additional rules may decrease rather than increase performance.

The simplicity of Distinguish's "rules" minimizes many of these problems. Activity restrictions apply only to a single activity and do not interact with other restrictions, thus reducing the chance of unintended side effects when modifying rules. Inference chaining does not take place, which removes the possibility of long, convoluted chains of reasoning that are difficult for end-users to parse. However, because of these and other simplifications described in previous sections, the Distinguish system is much less powerful than complete logic engines or probabilistic machine learning algorithms.

The system makes up for this deficit by allowing end-users to generate their own rules about how they do things at home. In constructing their own rules, users perform the most complex reasoning tasks – ones that automated systems are most likely to fail at. In this manner, users directly encode their personal knowledge of their habits and tendencies (or, in the case of the web survey, the average habits of a larger population of individuals) into the knowledge representations of the system, without a need for the system to "guess" what those pieces of information are.

For example, one user might reason, "when I come home I always put my keys in the cup next to the door. If I link this cup with 'coming home,' the system should be able to recognize this activity correctly." This formulation results in a simple rule (*key cup* → *coming home*) to describe a complex behavior (*coming home*). A probabilistic system might automatically learn a similar association, or a common-sense engine might come to the same conclusion after applying a series of rules, but such effort is unnecessary if the user can simply provide this information up-front. This is not to say that the user's reasoning is necessarily sound. If, one night, the user neglects to place their keys in the cup when returning home then the system will fail. However, having constructed the rule in the first place<sup>3</sup>, the user will be able to understand the reason for failure, and may be able to use his or her knowledge of the reasons for the current behavior to construct a new, superior rule. On the other hand, the user may decide that this particular night is a rare exception and not adjust the activity model.

Distinguish's innovation does not come from the design of its (relatively simplistic) recognition algorithms, but from the direct and meaningful inclusion of the end-user in recognition process. Other systems employ much more powerful algorithms, but in doing so dramatically reduce the potential of the user meaningfully understanding the behavior of their home.

---

<sup>3</sup> Built-in rules may require some additional effort to understand, but because they are written using the same system, the user should be able to analyze and correct them in the same manner he or she would analyze and correct self-constructed rules.



# Evaluation and Results

Distinguish is a combination of multiple interrelated systems: a common-sense knowledge database, an activity recognition algorithm, and a portable control interface. These systems are dependent upon one another to achieve the overarching goals of *intelligibility*, *control*, and *iterability*.

## Common sense survey

The two common-sense databases (object associations and activity restrictions) were generated from data gathered from a web survey. This survey was administered through a website over a period of three weeks. The survey was originally distributed to friends and acquaintances of the research staff, but the website was open to the public, and participants were encouraged to invite their friends to participate.

The survey contained questions about 108 common house objects and 51 home activities, including activities involving leisure/entertainment, work, cleaning, cooking, resting, social interaction, and grooming/personal hygiene (see Appendix D for the complete list). There were a total of 261 distinct questions (108 object questions, and 3 variations of 51 activity questions regarding room, time, and location). Individual participants were asked 20 of the possible 261 questions, and once those were completed could elect to answer more in batches of 10. In total, 95 people participated in the survey, with an average response rate of 23 questions per user (standard deviation:  $\sigma = 16.73$ ). Eight participants answered 0 questions, while the most prolific participant answered 100. Questions were evenly distributed amongst users; as a result, every question was answered 9-10 times. Descriptive statistics of the survey data are available in Table 2.

Total participants	95
Questions <sup>4</sup> answered per participant	23.1 ( $\sigma = 16.73$ )
Total values <sup>5</sup> provided per participant	62.3 ( $\sigma = 49.4$ )
Values provided per question	2.48 ( $\sigma = 0.8$ )
Number of custom values <sup>6</sup> supplied	213 (3.5%)
Number of "no restriction" values supplied <sup>7</sup>	150 (2.4%)
Total object associations <sup>8</sup>	3482

<sup>4</sup> A "question" is an instance of one of the two possible questions asked in Fig. 3 and 4, e.g. "What activities is <object> associated with?" and "<What times/what rooms/what postures> are required for <activity> to occur?"

<sup>5</sup> A question may be answered with multiple values. For example, the answer "Soap is associated with *washing hands* and *taking a shower*" contains two values (*washing hands* and *taking a shower*).

<sup>6</sup> In addition to choosing from a predefined set of activities, participants could "write their own." This was only possible with object association questions.

<sup>7</sup> When answering an activity restriction question, participants had the option of specifying "any of these," indicating that there was no restriction on when/where/how the activity took place.

<sup>8</sup> Total number of values supplied to object association questions

Total activity restriction values <sup>9</sup>	2666
Total values supplied	6148

Table 2 – Survey summary

In general, participants tended to be fairly terse in their answers, provided less than 3 values<sup>10</sup> per question (and with a relatively tight  $\sigma$  of 0.8). Participants also tended to utilize the built-in values rather than creating their own (in fact, many of these custom values are repeats of built-in versions that the participants were apparently unable to find when browsing). In addition, few restriction questions were answered with the “any of these” response – participants almost always provided specific answers that excluded some values.

### Database construction

The common-sense databases used to perform recognition were constructed directly from data gathered from the surveys – no definitions were “tweaked” or custom-crafted by the researchers. Entries in the database consisted of simple lists of values: objects were associated with lists of possible activities, while activity restrictions were associated with lists of acceptable values (for example, the “room” restriction for *washing hands* might be: [kitchen, bathroom]).

A simple popularity metric was employed to eliminate excess noise from the survey responses. For every question in the survey, answers were ranked based on how many times they were supplied by survey participants (for example, a score of 9 would indicate that nine unique people supplied that answer for that particular question). A minimum popularity threshold was then computed for each question based on its most popular answer:

$$\omega = \alpha P$$

where  $P$  is the popularity of the most popular answer and  $\alpha$  is a factor from 0 to 1 used to control the amount of noise removed. All answers with popularity greater than  $\omega$  were accepted; all others were discarded. The selection of this number is left up to the architect of the system – a large value will reduce noise, but possibly cut off valuable information, while a small value may include relatively useless information. This value may be calibrated based on the source of data (trusted participants or general population) or on the desired complexity of definitions. In the following experiments, a value of  $\alpha=0.7$  was used to filter out most noisy responses while preserving an acceptable richness of definition. An example filtering process is presented below.

#### **Object: Pen & pencil holder, $\omega=4.9$**

Activity	Popularity	Included? ( $P > \omega$ )
Writing	6	Y
Working	5	Y
Studying	5	Y

<sup>9</sup> Total number of values supplied to activity restriction questions

<sup>10</sup> i.e. fewer than three activities per object and fewer than three allowed values per restriction.

General organizing	5	Y
Working on a hobby	3	
Talking on the phone	2	

Using  $\alpha=0.7$ , each object was associated with average of 2.8 activities ( $\sigma = 1.6$ ). Restrictions contained an average of 3.0 ( $\sigma = 1.9$ ) allowed values<sup>11</sup>. These synthesized rules were slightly lengthier than the average response to a question in the web survey (2.5 values per answered question).

### Base recognition results

The Distinguish algorithm is designed to be customized by the end-user in order to achieve optimal recognition performance. However, in the interests of generating a baseline performance metric, the accuracy of the system was evaluated on a naturalistic dataset taken from a real home. This evaluation used the “generic” definitions taken directly from the common-sense databases – no adjustments were made to tailor the definitions to the target home.

Three weeks of fully annotated data was obtained from a 2009 installation of the BoxLab in the home of a single male. The BoxLab is a mobile version of the PlaceLab [3] sensor network that can be installed in pre-existing homes [36]. Object usage data were gathered from MITes [21], small wireless object usage sensors that were attached to commonly-used objects. This particular installation contained 185 MITes sensors placed on objects throughout the home.

The three-week dataset was hand-annotated with activity labels by a third-party who did not live in the space. These annotations included the current activities taking place, social contexts, postures<sup>12</sup>, and the current location<sup>13</sup> of individuals in the apartment.

This dataset is a recording of the day-to-day life of a real occupant of the space. It and similar collections are challenging activity recognition datasets (see [11]) due to noise, sparse activities, or missing or sparse data – problems that occur in real sensor deployments in homes. There are lengthy spans of time where no sensors fire at all, even though multiple activities are taking place. Such data voids occur frequently and in a variety of situations. Just as frequent are situations where only one or two sensors fire. For example, many of the occupant’s visits to the kitchen only tripped the *faucet* sensor. Taken by itself, this single piece of data might indicate *washing hands, getting a drink, washing dishes, or general cleaning*. In another example, the firing of a *dresser drawer* object in the bedroom was the sole indication that the following activities were taking place: *waking up, getting dressed, grooming, and personal hygiene*. In such situations, there is simply not enough information to make confident decisions. This problem is not unique to the

<sup>11</sup> Room, time, and posture restrictions have a maximum number of allowed values of 7, 7, and 4, respectively.

<sup>12</sup> Normally the occupant of the BoxLab would wear wearable sensors that permit accurate inference of basic posture (standing, sitting, lying down). Unfortunately, this system was not fully functional at the time of recording of the dataset used in this work. As a result, in this work we simulate this data using the manually annotated posture data.

<sup>13</sup> The current location of the occupant was inferred from object-usage data. This is a relatively coarse metric; other systems [3] exist that provide much more precise location information.

automated system – a human would have similar difficulty making a decision in such situations, although he or she might be able to use their knowledge of the long-term behavior of the occupant of the space to make a more educated guess.

When Distinguish encounters situations where the recognition system is uncertain as to what is occurring, it does not provide any labels for the data. Situations of uncertainty occur when no data is present or multiple activities have the same winning score, resulting in a tie. Ties are most likely to occur in situations with only one or two pieces of data, as each data point will be associated with multiple activities, which will each receive a single vote. This behavior is intentional – random guessing is unlikely to arrive at the correct answer and is much more likely to aggravate the end-user with unsupportable output.

The system achieved a total accuracy of 20.4% with 51 target activities over the 20 days of test data. The majority of errors (58%) were false negatives, in which an activity was taking place but the system did not recognize it, either due to insufficient data, a voting tie, or an error in recognition. A random-guess algorithm would have achieved an accuracy of 1.9%,

<b>Correct classifications</b>	1082 (20.4%)
<b>False positive errors</b>	1253 (23.7%)
<b>False negative errors</b>	2951 (55.8%)
<b>Total errors</b>	4204 (79.6%)
<b>Total classifications</b>	5286 (100%)

Table 3 – Baseline performance

Pure, “generic” common-sense was used to generate this result, and so may not have accurately represented how the particular occupant of the home went about his daily routine. A breakdown of accuracy by activity is available in Table 4.

<b>Activity</b>	<b>Accuracy</b>	<b>Correct recognitions</b>	<b>Total recognitions</b>
Watching TV	100%	66	66
Using a computer	80.1%	744	923
Shaving	68%	13	19
Studying	63.4%	26	41
Putting clothes away	59.0%	23	39
Sleeping	44.5%	65	146
Preparing a meal	43.2%	32	74
Getting/preparing a drink	41.7%	5	12
Washing hands	27.8%	10	36
Putting the dishes away	23.6%	21	89
Getting ready for bed	16.1%	5	31
Doing the dishes	15.9%	21	132
Going to the bathroom	14.9%	20	134
Waking up	12.1%	17	140
Cleaning the bathroom	9%	4	42
Eating a meal	7.3%	5	68
Throwing something away	0%	0	29



Sweeping	0%	0	63
Taking a shower	0%	0	78
Cleaning teeth	0%	0	30
Ironing	0%	0	5
Doing laundry	0%	0	21
Listening to music	0%	0	3
Healthcare	0%	0	20
Taking a nap	0%	0	52
Eating a snack	0%	0	5
Reading	0%	0	5
General organizing	0%	0	5

Table 4 – False positive accuracy

A modified confusion matrix is available in Table 5. Because Distinguish uses a slightly modified activity taxonomy compared to the BoxLab annotations, the axes of the matrix are not symmetric. As a result, correct recognitions are displayed in the first column instead of the diagonal of matrix. In addition, a weighted counting metric was used to generate the confusion values. When the system makes a mistake, it is difficult to determine which of the recent activities should have been selected instead. Simply incrementing the confusion count of all recent activities would create the impression that many more errors were occurring than actually took place. To account for this ambiguity, each recent activity was incremented with an equal *fractional value*, the sum of which was one. See Appendix H for a version of the matrix containing specific numerical values as well as a more detailed description of the process used to create the matrix.

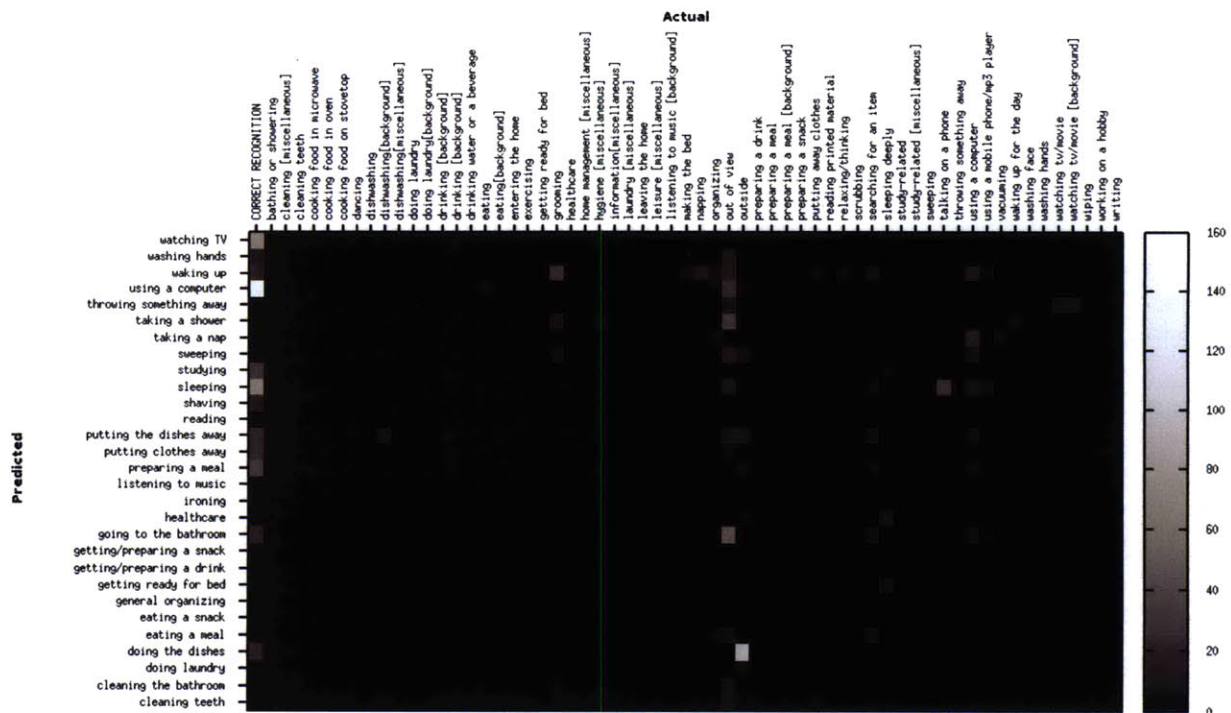


Table 5 – Confusion matrix. Correct recognitions are displayed in the leftmost column. Note the bright columns corresponding to “out of view”, “outside”, “searching for an item”, and “using a computer” activities. Some of these

activities, such as “searching for an item” were not part of the Distinguish activity taxonomy and were therefore impossible to recognize, while others were simply very common, such as “using a computer.”

These numbers represent a lower bound for the performance of the system as Distinguish is designed to be improved by end-users familiar with the space and the habits of its occupants before it achieves optimal performance. A number of objects in the occupant’s house were used for extremely specific purposes that were not captured in the information available in the default common-sense databases. For example, one particular bowl was used exclusively by the participant to store his keys when entering or leaving the apartment, but this object was automatically associated with eating and dishwashing activities due to its classification as a “bowl.” Such errors of generality are common problems that end-users can quickly rectify using the Distinguish interface.

### Usability study

A usability study of the Distinguish interface was carried out with the goal of answering the following questions:

1. Is the voting concept employed comprehensible to end-users?
2. Does the interface allow the end-user to correct the system?
3. Is it possible for the end-user to make changes that improve the system’s performance in the current moment without degrading performance in the past or future?
4. When the user cannot correct the system, is it clear to that person why correction is not possible (either due to poor data or a weakness of the algorithm)?

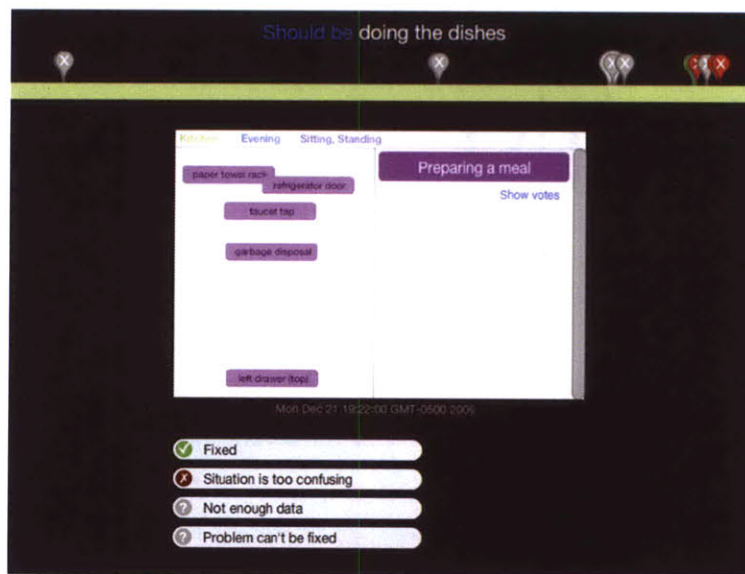
Distinguish is designed to run on a phone and be used *in-situ* by the occupant of an instrumented home. When errors occur, the occupant can correct the system with the memory of recent events still fresh in his/her mind. However, such *in-situ* user studies frequently reduce the number of practical participants to at most one or two. In order to evaluate system feasibility with a larger number of participants, the test was performed on prerecorded data and participants were asked to correct the system without any familiarity with the environment involved other than a video of the scene – an even more difficult task. Participants were evaluated on their ability to understand the voting system, correct recognitions errors, detect noisy or insufficient data, and identify situations where the correction was not possible (due to sensor noise or a weakness in the voting system).

Each evaluation lasted approximately an hour. At the beginning of the test, users were given a 15-minute tutorial on the interface that introduced them to the voting concept, adding and removing object associations (lines), and modifying activity restrictions. Participants were also warned about the possibility of noisy data due to sensor malfunction or environmental interactions (e.g. air from opening a door blowing the shower curtains). See Appendix E for the text of the tutorial.

After the tutorial, participants were asked to improve the system’s ability to recognize the activity *doing the dishes* over a 24-hour period. This particular activity was selected for a number of reasons. First, it occurred multiple times during a normal day (mean=3.2, sd=2.7), and so presented a variety of recognition challenges for the participants. Second, it

was similar to many other kitchen-related activities (such as *setting the table*, *preparing a meal*, and *putting dishes away*) and frequently co-occurred with them, requiring participants to think carefully about their changes lest they introduce error elsewhere. Third, it occurred with varying durations (median=5 s, avg=3.74 min  $\sigma$ =14.4 min, min=2 s, max=79.8 min). Finally, it was the most frequently misclassified activity that met these requirements, with 84.1% misclassification by the default system.

Participants were shown an idealized timeline representing a day of data where the dishwashing activity took place nine distinct times, as identified by the annotator (Fig. 12). These situations ranged from momentarily washing a drinking glass to washing many dishes and loading the dishwasher. Situations where the system made a mistake were marked with “X”s. False positive and false negative errors were colored differently from one another.



**Fig. 12 – User test interface. Remaining errors (top) are grey (false negative), red (false positive), or green (correct). The interface (center) displayed the data and recognition that took place when the currently selected error occurred. After attempting to correct an error, the participant may select one of four feedback options (bottom).**

The day of data selected contained 6 situations where “doing the dishes” was classified correctly, as well as 49 situations where it was incorrectly classified (12 false positives, and 37 false negatives). Activity labels were recomputed every 60 seconds, with the result that the same error might effectively appear multiple times over the course of a few minutes. Strings of these extremely similar errors that occurred within a minute of each other were represented with a single “X” icon. This resulted in 3 “correct” icons, 3 “false-positive” icons, and 13 “false-negative” icons, although during the test, a participant’s changes might have created new errors, and thus new icons that needed to be addressed.





**Fig. 13 – Video recording display.** Two video feeds were visible during the test: a broad view of the living room/kitchen area (left) and a close-up view of the kitchen area (right). Participants did not interact with the video feed interface; they were only able to watch video as it played.

When a participant clicked on an X icon, the interface would display the last 5 minutes of data gathered at that particular time as well as the resulting recognition. The participant could then attempt to modify the recognition result using the interface. Before attempting to fix a particular error, participants were shown a video recording of the activity taking place (Fig. 13). Participants were instructed to attempt to fix as many of the visible errors as possible, or indicate that fixing a particular error was not possible. After attempting to fix a particular error, participants were able to select one of the following summaries:

- **Fixed** – “I think that I’ve fixed the mistake that the system was making.”
- **Too confusing** – “The situation is too confusing for me to understand or fix.”
- **Not enough data** – “There isn’t enough data for the system to make a correct decision.”
- **Problem can’t be fixed** – “There’s enough data, but the problem can’t be fixed for some other reason, such as sensor noise or the weakness of the design of the algorithm.”

After every correction attempt, the system recalculated its classifications over the entire day. If new errors were generated, they appeared on the timeline and participants were asked to correct them as well. If the participant selected “fixed” then the error icon would only be removed if the recognition error was in fact corrected. If s/he selected any of the other three options, the error was hidden from the timeline.

After the end of the test, participants were asked a series of interview questions (show in Appendix F), and were allowed to provide any additional feedback.

## Study results

Ten participants were recruited to participate in the IRB-approved study. All participants were undergraduates at the Massachusetts Institute of Technology who responded to an e-mail advertisement, shown in Appendix G. Three of the ten participants had previous experience with programming languages. Participants were paid \$10 for participating in the study.

### Quantitative results

Summary statistics of the changes in recognition can be found in Table 4. After completing the test, participants were, on average, able to increase the number of “correct” recognitions from 6 to 17.1. False positives increased slightly, while false negatives decreased significantly. In total, recognition performance of the “doing the dishes” activity increased from 12.2% to 53.5%, more than quadrupling the system’s ability to correctly recognize this particular activity.

	Pre-test recognition results	Post-test recognition results	Pre-test error icons	Post-test error icons
<b>Correct</b>	6	17.1 ( $\sigma=6.0$ )	3	6.6 ( $\sigma=1.0$ )
<b>FP</b>	12	13.7 ( $\sigma=7.3$ )	3	1.2 ( $\sigma=1.9$ )
<b>FN</b>	37	23.5 ( $\sigma=6.1$ )	13	3.5 ( $\sigma=3.2$ )

Table 6 – User changes to recognition performance

The average participant ended the test with 6.6 “correct” icons (up from 3), 1.2 “false-positive” icons (down from 3), and 3.5 “false-negative” icons (down from 13).

In general, participants were most successful in increasing correct recognitions and removing false negatives. Some participants were quite effective at removing false positives as well, but, on average, the group created more than they eliminated. Most of these problems occurred to due boundary errors – participants would correct false negative errors, causing the system to more aggressively recognize the desired activity. This would then cause the system to classify the activity for slightly longer (~1-2 minutes) than was annotated, generating new false positive errors directly before or after the corrected time. Some participants were able to resolve these new errors; others chose to mark them as impossible to fix.

By the end of the test, participants had decreased the number of problems remaining on the screen from 16 to an average of 4.7. The most common reason for the remaining errors was that the participant ran out of time. No participants stopped early because they felt that they couldn’t fix or classify any more errors (in other words, no participants felt that the test was too confusing to complete).

Action	Average usage per participant	Std. dev. ( $\sigma$ )
Add line	7.6	6.0
Remove line	7.8	5.5
Modify restriction	9.3	7.8

**Table 7 – Average user interactions with interface**

Adding and removing lines were collectively more popular than modifying restrictions. Participants tended to favor modifying restrictions at the beginning of the test, before transitioning to line modification (see strategies section, below). However, in general there was great variation between users (note the relatively large  $\sigma$  values)

User response	Average ( $\mu$ )	Std. dev. ( $\sigma$ )
Fixed	10.1	7.6
Too confusing to fix	0.6	1.0
Not enough data	5.9	2.5
Problem not fixable	3.6	3.0

**Table 8 – Average user responses to errors**

Very few participants marked situations as “too confusing” (0.6 confusing situations per participant). Participants were initially reluctant to mark any other response but “fixed,” but after experimenting with the data and recognition system became more confident in their understanding of the source of error and began to select “not enough data” and “problem not fixable” as well.

**Participant behavior and feedback**

In general, all participants reported that the interface was intuitive and the voting system easy to understand. Most complaints were targeted at the data itself, either because there wasn’t enough of it in a particular situation, or because sensors were firing at inappropriate times. However, during testing a number of common themes emerged across participants that may provide guidance for future work in this area.

**Idealists vs. puzzle solvers**

Two distinct meta-strategies became quickly apparent during testing. Five of the participants guided their changes based on what might actually be possible or how they did things in their own home. Such participants sometimes would refuse to remove or add lines or restrictions even though they knew that doing so would solve their particular problem because they were reluctant to degrade the ideal representation of some activity. Such participants were quicker to recognize and flag insufficient or misleading data and move on to the next error. These participants will be referred to as the “idealists.”

The remaining five participants tended to view each particular error as a distinct puzzle that needed to be solved. Puzzle solvers were reluctant to supply any other answer than a “fixed” one, and they were much more willing to adapt the object associations and activity restrictions to fit the particular situation. As a result, puzzle solvers tended to create many more new errors than idealists, as well as becoming susceptible to error oscillation (see next section). At the same time, however, persistent puzzle solvers were capable of achieving the highest overall performance increases. These participants will be referred to as the “puzzle-solvers.”

**Oscillating errors**

There were a number of situations in the tests where the correction of two errors was mutually exclusive: fixing one problem would generate a new error, but fixing the new

error would cause the old one to return. This caused many participants to oscillate back and forth multiple times between fixing the same errors, sometimes making and undoing the same changes over and over again. Puzzle solvers were especially susceptible to error oscillation.<sup>14</sup>

However, after having spent some time fixing oscillating errors, participants began to realize what was happening and began making more strategic edits. Almost every participant was able to overcome problems with oscillating errors, either by discovering through trial and error the particular changes that allowed them to circumvent the problem or by marking the error as unfixable. Puzzle-solvers were surprisingly proficient at the former balancing act, although a number complained about the need to mentally predict the outcome of their changes in other areas. This act of “keep everything in their head” was perceived as too much mental effort. Because puzzle solvers did not use “idealistic” information to guide their editing decisions, they were forced to remember all of the edits they had previously made and why those edits had successfully fixed old problems. Because their previous edits were frequently the cause of new errors, they would be tempted to undo them when fixing those new errors, which would cause the old errors to return. Puzzle solvers dealt with these problems by either realizing that they were in an oscillating situation after encountering the same error multiple times and moving on or by finding a third solution that avoiding the oscillation. One participant made the comment that the situation was similar to some logic puzzles with interrelated clues of the form “Jane is taller than Joe, but Joe is twice as short as Alex, who is the same height as Mary.”

Because idealists based their edits on the ideal knowledge representation, they were better equipped to recognize situations where making a change would cause the system to perform poorly elsewhere. Idealists tended to simply mark such situations as “not enough data” or “problem can’t be fixed” and move on.

### **Strategies and brevity of action**

When attempting to fix a particular error, some participants attempted to understand all of the votes and restrictions taking place before making a change. Others quickly identified what *should* be recognized and made changes targeted directly at causing that activity to be chosen, ignoring the rest of the votes. Some participants had a specific order in which they solved errors (first eliminate bad restrictions on the target activity, then add or remove

---

<sup>14</sup> For example, one participant bounced between two linked errors more than 5 times. S/he would solve the original false-negative error by associating a *drawer* object with *doing the dishes* in order to break a tie. Because this *drawer* object was associated with no other activities, *doing the dishes* was now correctly recognized. However, this caused a new false positive error elsewhere. In this case, meal preparation was occurring, but due to the presence of the *drawer* object and relative scarcity of other data, *doing the dishes* was chosen instead. To remedy the situation, the participant would add an association between *meal preparation* and *doing the dishes*. This would solve the problem but cause the original error to return because there was now a tie between *doing the dishes* and *meal preparation*. The participant then oscillated between adding and removing the same association of *drawer* with *doing the dishes* as he attempted to repeatedly correct the same two errors. After the fifth attempt, the participant realized that the two were connected, and moved on.

lines, then add restrictions on unwanted activities), while others generated ad-hoc approaches to each situation.

Common among almost all participants, however, was a focus on brevity of action. Participants were rarely interested in editing the situation so that all of the votes and restrictions represented the “ideal” knowledge representation of what might be taking place. Instead, participants made the bare minimum of edits necessary to generate the desired classification. Idealists would rarely make changes that *degraded* the “ideal” model of an activity, but neither would they go out of their way to *improve* it<sup>15</sup>. Objects that were missing associations were rarely added to, even if they were clearly an important part of what was taking place. Such action was only taken if their vote was necessary in order to achieve correct recognition.

All but one of the participants began the test by attempting to modify restrictions rather than object associations. However, due to the situation selected, restrictions were generally found to be less helpful than modifying association lines, and participants were forced to change their approach. It is possible that this affinity for restrictions is motivated by the same inclination toward brevity. Given a particular situation where one activity is being recognized instead of the correct one, it is frequently easy to think of a reason why the recognized activity is *not* taking place, rather than why all of the data supports a different activity instead. This reduces the decision process to a single, negative answer, as opposed to a complex answer that might involve many pieces of object data.

### **Relationship to real data**

Surprisingly, participants were, in general, uninterested in the video recording of the activity taking place in the real space, even though they were informed that sensor noise and undependability would require them to evaluate what was actually taking place before making a decision. Left to their own devices, most participants (even the idealists) tended to think of the situations in abstract terms, rather than originating from a real situation. For example, multiple participants asked if they could manually “add” data to the situation in order to make it easier to recognize. When asked how they made the decisions that they did, most participants responded by describing how they personally might perform the action, or how a generic person might do so, rather than the person visible in the video.

However, when *forced* to watch the video beforehand, participants were able to recognize situations where sensors were firing incorrectly or not at all. They asked questions such as “what is this sensor actually attached to?”, “what objects are inside this drawer?”, and “when the dishwasher is turned on, does that shake the counter?”. This suggests that this issue may become less pertinent to actual occupants of a smart environment who are personally motivated to improve their sensor data and recognition performance.

---

<sup>15</sup> For example, in one situation a participant was attempting to force the system to recognize *eating a meal* instead of *doing the dishes*. To do so, the participant linked one of the kitchen chairs to *eating a meal*, which increased its score and caused it to be chosen. However, the participant did not link any of the other two kitchen chairs (that had also been recently moved), although they were most likely involved in the same manner. Only one new line was necessary in order to achieve correct recognition, and so only one was added.



In pursuing this course of data-ignorance, some participants essentially increased the difficulty of the task. While the fact that they were still able to complete this more difficult task is promising, further work should explore ways of testing such system without forcing participants into performing actions “the hard way” (see Discussion section).

### **Data overload**

Simple situations involving only a few pieces of data (and perhaps 1 or two activities) offered few problems for participants, although they would occasionally be confused by the lack of enough representative data. For example, short examples of the “doing the dishes” activity might only involve a single cup (with no sensor on it) and the faucet. Early in the test, participants would become confused that they only had the *faucet* data point to work with. However, six of the ten of participants complained that the interface quickly became too complex when large amounts of data were measured during the 5-minute window. In this case, many votes and many rounds of voting would take place, resulting in a situation that might take more than a minute to fully understand. There is no evidence in the study results to indicate that such situations were actually harder to correct, but these six participants reported that they reacted negatively to the increased time and mental effort required to debug such complex scenes. One participant reported that she tried to fix such errors and move on as quickly as possible, simply to minimize the amount of time she had to spend thinking about such problems. In order to identify the desired corrective action, users must first sift through a much larger set of votes and restrictions, many of which may end up involving completely unrelated activities. In addition, the small size of the screen increases the difficulty of navigating and manipulating extremely large sets of data.

### **Extending the system**

At the end of the test, participants were asked if there were other kinds of information they wanted to tell the system in order to improve its performance beyond simple object associations and where/when/how restrictions. Many possible restriction types were considered when designing the system, such as temporal relationships (X activity must occur before Y activity) or exclusionary information (X activity cannot co-occur with Y activity). However, there was no consensus among participants regarding additional information. The majority of participants (6) expressed complete contentment with the current state of the system, instead choosing to blame noise in the data for the system’s failures. Suggestions included various weighting approaches to object association, such as ranking, multiplicative factor weighting, and being able to specify whether an object was “necessary” or only “sometimes involved” in a particular activity.

However, eight of the ten participants expressed dissatisfaction with the current restriction model due to its relative lack of utility during the test. The “time” restriction was rarely used, since, in the words of one participant, “most activities can theoretically take place at any time of day.” The posture restriction was also a frequent source of frustration for end-users due to its relationship to the 5-minute window. Postures were not linked to a specific time of occurrence, but were simply marked as occurring or not occurring in the last five minutes. Thus, if the subject was standing and then transitioned to sitting, it was difficult to exclude all non-sitting activities because standing was still marked as occurring in the last five minutes.

## Data magnitudes

In general, participants struggled with correcting recognition in the following situations:

- **No data** – there were a number of situations during the test when no sensors fired at all, even though activity was taking place. This was initially jarring for most participants, as they were not yet accustomed to dealing with both problems of recognition as well as sensor misbehavior. Multiple participants asked if they could “add” data to the left side of the screen, while others blamed themselves for somehow erasing the information they needed. However, after encountering multiple situations with no data, participants learned to recognize them and move on.
- **Insufficient data** – Many situations during the test involved only one or two pieces of data. Frequently, these data were not sufficient to distinguish the desired activity from other, equally likely candidates. For example, in one situation only the *faucet* and *upper cabinet* sensors fired, which might be indicative of a number of kitchen-related activities. Some participants attempted to fix these situations by simply skewing the votes dramatically in favor of their desired activity, but this strategy frequently created new errors elsewhere on the timeline. Correcting those errors in a similar manner would usually result in error oscillation. Similar to problems of no data, participants learned to identify these situations and simply move on instead of trying to force the system to make a correct decision with inadequate data. Some participants were more successful at this than others – puzzle-solvers tended to fixate on the problem rather than the reality of the situation, and so would more frequently decline to mark such situations as “not enough data.”
- **Too much data** – As previously mentioned, some situations involved so much data that the resulting voting rounds were fairly complex and took a not insignificant amount of time to parse and understand. While there is no evidence that these situations were actually more difficult to correct, multiple participants reported feeling overwhelmed during such situations.

As the above list might indicate, the user experience is closely dependent upon the quality of data gathered by the in-home sensors. Too little information and end-users become frustrated with their inability to correct the system, while too much information causes increased mental strain and unpleasant feelings of the situation being “out of control.”

When asked what the most confusing aspect of the system was, 7 participants responded with a data-related answer. Only three of the ten participants mentioned aspects of the UI or the voting system. In these cases, two participants highlighted situations where the voting round system would cause one activity to “steal” votes from another activity, while the final participant responded that the most confusing aspect of the experience was the manner in which restrictions were verified<sup>16</sup>.

---

<sup>16</sup> In order for a restriction to validate, only one of the required values must be present. For example, if a restriction requires “sitting” and both “sitting” and “standing” have taken place in the last five minutes, the restriction is validated. This distinction was mentioned during the tutorial, but is not readily apparent in the UI.

# Discussion

Distinguish's baseline recognition performance is low compared to state-of-the-art systems (most of them dependent upon some form of machine learning), which frequently score between 80% and 90%, although rarely with naturalistic data or a large set of activities. However, this baseline performance represents a system operating solely from data gathered from a web survey. It has no knowledge about the specifics of the environment into which it has been installed, or of the quirks of behavior that its occupants exhibit. The Distinguish system is designed to only function well after it has been adapted to its particular space by the occupants of that space. As such, this performance metric presents a lower bound on the possible recognition performance of the system.

The fragmented nature of naturalistic human behavior continues to be a major obstacle to dependable activity recognition. Occupants of a home rarely perform activities one at a time – instead they interleave them, perform some activities simultaneously, or inject micro activities into their main activity, such as *organizing* for just 2 seconds. The occupant of the target home exhibited all of these behaviors, and as a result the ground-truth annotations contain many disjoint, fragmented, or micro activities. For example, the occupant might *wash the dishes* for only a second while rinsing a mug, or *organize* his desk while simultaneously *talk on the phone* and *use the computer*. In addition there are frequent situations where human occupants appear to be in transitional mental states and thus are not performing any one particular activity, although they are certainly being active. Occupants change their minds mid-way through performing an activity, or traveling to perform an activity, or upon arriving at an area in order to perform an activity. A significant portion of the occupant of the target home's time was spent in such "waffling" behavior that was both difficult to annotate as well as recognize.

Unfortunately, human memory of such events does not always capture this fragmentary nature of our own behavior. In hindsight, it may be easier to classify an extended collection of ambiguous behaviors under an umbrella term, but this is difficult to accomplish with real-time recognition. For example, in one situation, the occupant of the target home began by putting on a jacket, but then started to tidy his workplace. Midway through this activity he stopped and washed a couple of dishes, only to return to his computer to check a website and continue to tidy the work area. He then wandered around the house for a couple of minutes with no apparent purpose before returning to the computer once more and then exiting the apartment. Viewed as a whole, this extended activity, which took place across almost all rooms of the apartment, might be labeled as *leaving home*, but the diversity of behavior within this term makes correct recognition difficult. In addition, if the occupant had changed his mind and instead returned to working at the computer, we might instead label the entire process as *cleaning up*, even though very little data has actually changed.

In addition, naturalistic data gathered from an entire home is fundamentally problematic: sensors are undependable and noisy, often firing too often or due to unrelated actions, or not firing at all. For example, in the source home, the “shower curtain” sensor frequently fires whenever someone enters the bathroom, regardless of the activity they plan on performing. As a result, the system frequently interpreted almost all bathroom activity as involving “taking a shower” in one manner or another. The cause of this malfunction is not the recognition system, but rather the placement of the sensor on a loose section of shower curtain that is frequently disturbed by simply opening or closing the bathroom door.

A machine-learning system might be able to adapt to learn this particular association, but without an understanding of what’s going on, the end-user might well disturb this new, careful balance by changing the way the shower curtain is stored, or adjusting the room so that air flow no longer disturbs it. The system will then break once again, to the bewilderment of the occupant. Distinguish, on the other hand, allows end-users to understand the nature of the information being gathered from nearby sensors. Thus, the end-user would be able to discover the misbehaving sensor and either move it in order to reduce noise or adjust its associated activities to reflect this presence of noise.

During user testing, participants demonstrated their ability to correctly distinguish between situations caused by sensor error vs. recognition error<sup>17</sup>. In fact, this was a significant source of user frustration, with one user reporting that she felt that she was being unfairly held back in her attempts to fix the system by specific poor sensor readings<sup>18</sup>. A question remains as to whether users, once they have identified that sensor error is taking place, will be able to correct the error by repositioning or replacing a sensor. This question is beyond the scope of this work, but anecdotal evidence during testing suggests that users understood the basic principles causing noisy or missing sensor data<sup>19</sup>. Missing data requires adding additional sensors. Improving data from object usage sensors requires either (1) adding additional sensors or (2) moving objects or sensors so that they do not react to unrelated movement. Multiple participants expressed a desire to add additional sensors to objects that they saw the occupant of the home interacting with, but no participants mentioned a desire to relocate sensors in order to avoid noise. Further study is required in order to ascertain whether end-users are fully capable of modifying their personal sensor layout.

Problems specific to the layout of the particular home used were another source of error in the baseline system. Some errors related to the failure of a generic label to properly

---

<sup>17</sup> However, they only did so when forced to view the video recording of the situation before attempting to use the interface. Participants who ignored the video data were much less likely to mark “correction not possible,” or if they did, to recognize that there was a problem with the sensors’ data-gathering abilities.

<sup>18</sup> While the presence of such errors is undesirable, the fact that participants were able to assign blame correctly is desirable.

<sup>19</sup> During the test, one common source of noise was the dishwasher, which shook the surrounding counter-top and caused other, unrelated object sensors to fire. All participants correctly identified these situations as “bad data,” and three of the ten suggested out loud that the cause was due to the vibrating dishwasher. Other situations occurred during testing where participants correctly guessed or asked whether certain objects were missing sensors. This suggests that these participants would be able to identify situations where more sensors would aid in recognition, and attach them to the appropriate objects.

represent the specialized use certain objects attained in the target home. Some pillows were only used for napping, while others were used for sleeping. Some drawers contained knives and cookware while others contained cleaning supplies. In addition, some objects were completely misclassified, such as the key cup/bowl described in the previous section. However, none of these distinctions were present in the “generic” common-sense databases used. The layout of the participant’s home was also nonstandard, and so required some adjustment of the activity restrictions in order to function correctly. For example, the “coming home” and “leaving home” activities are normally restricted to only take place in the “entrance/exit” section of the home, but this particular home lacked any such area – the occupant entered and exited directly through the living room.

During the user test, it was found that end-users were able to improve the system’s ability to correctly recognize specific activities without generating new errors, although some participants required an extended period of trial-and-error in order to achieve this result. In addition, all participants were able to identify areas with insufficient or noisy data, although some participants were quicker to come to these conclusions than others. This preliminary finding is a positive one with regards to the practicality of end-user modification of activity recognition. Participants’ comments regarding the ease of understanding the system coupled with their ability to greatly improve the recognition performance of the system suggest that their involvement can be constructive and rewarding to the design of an activity recognition system.

### **Speed of correction**

Speed of correction is a critical requirement for *in-situ* interfaces such as the one employed by Distinguish. When systems are first installed in the home, errors will occur frequently and end-users will be required to make many corrections during the first few days. Thus, a single correction effort must be as fast and easy to accomplish as possible in order to avoid placing undue burden on the occupants of the space. During user tests, the actual act of correcting or identifying a single error required less than 2 minutes after participants began interacting with the interface. While almost every participant used the full amount of time available, this was due to the number of errors that they were asked to correct, the creation of new errors, and the time required to view the video recordings of errors taking place, which might be up to five minutes in duration.

While single-error correction is demonstrably quick and simple to accomplish, *in-situ* occupants of a home may fall victim to the same oscillating errors experienced by some participants, especially if they employ a puzzle-solver strategy. This is less likely given their personal connection with the space as well as the personal behaviors generating the data, but is it possible for oscillating to occur with even the most “idealist” strategies. At the moment, the phone interface provides no support to allowing end-users to predict the outcome of their changes. Situations may occur where the system oscillates between two similar errors, but spaced out over a series of days or months. While testing has demonstrated that user memory is sufficient to detect and correct oscillations while exposed to errors one after another, it is unlikely that the average end-user will be able to remember their changes from a previous day or previous week. If long-term oscillation becomes problematic in the wild, the interface should be augmented to provide feedback to

end-users regarding the effects their immediate changes may have on future situations. For example, the timeline interface used during user testing might be adapted for phone-sized viewing.

## Recommendations

Despite some limitations, the user test highlighted strengths and weaknesses in the current design. Given this feedback, the following guidance for future work is provided:

- **Support brevity of expression** – the user’s time is valuable and should be respected by the system. When correcting the system, almost every participant made the fewest possible changes in order to elicit the desired result. At the same time, it is impractical to expect end-users to spend 30 minutes correcting the system every time an error occurs. At the moment, a user may need to make multiple related changes to the voting system or restriction lists in order to correct a single error. An ideal approach would allow the system to benefit from a single change by the end-user, even if such changes might have to occur more often. For example, if the system confuses *leaving home* with *coming home*, the user might indicate, “I have to be outside before I can come home.” Or, when the system confuses *doing the dishes* with *preparing a meal*, the user might indicate, “I *always* to use this particular sponge when doing the dishes.” Such rules are powerful but brief, and would allow the end-user to get back to what he or she was doing as quickly as possible. However, the design of a rule system that is expressive enough to enable such succinct corrections is non-trivial. Concepts from the previous examples such as the idea of temporal ordering or “always” are not currently supported in the Distinguish system. However, adding such complexity might result in a system that is too unwieldy for end-users to comfortably understand. Future work might explore methods for reducing the total number of rules involved in any one recognition attempt. If only a small number of rules are involved at any one time, the complexity of these rules might be safely increased. Such a system might be a hybrid of machine-learning and rule-based recognition: a machine-learning “core” performs recognition, the results of which are filtered by a set of rules. This set of rules would initially be empty, but end-users could add rules to correct the system in times of error.
- **Focus the area of assistance** – a number of participants complained that they spent much of their time trying to understand the entire voting layout for a small number of complex errors. Such errors usually involved many pieces of data and multiple rounds of voting. Frequently, the edits they finally chose to make only related to a small subset of these complex scenes, rendering most of the upfront analysis pointless. For example, during these situations other, unrelated activities were frequently co-occurring, such as *adjusting the lights*, *using a computer*, and *throwing something away*. Data and votes from these activities did not affect the activities of interest, but were still displayed in the interface. Systems that request aid from the user should clearly define the boundaries of what does and does not require their attention. This reduces the mental strain placed on the participant and makes it less likely that they will accidentally attempt to “fix” the wrong thing. For example, future versions of the Distinguish interface might allow the user to “zoom

in” on just the data that is currently involved in a few, selected activities. The user could then choose to “zoom back out” to associate currently unrelated data, if necessary.

- **Improve edit feedback** – one advantage of Distinguish’s voting algorithm approach is that the system can retrain and recalculate recognition results quickly, even on mobile devices, allowing users to view in real time the results of their actions. However, the current interface is not completely adequate in helping users to avoid the oscillating problem caused by related, frequently opposing errors. Users in the study demonstrated that they could, in fact, overcome these oscillating problems through perseverance and a reliance on their own memory, but such a system is not a long-term solution. For example, one user repeatedly oscillated between three errors involving *doing the dishes*, *preparing a meal*, and *eating a meal*. Quick fixes to either activity would cause errors in the other activities. The user eventually solved the problem by memorizing the associations for all objects involved with these activities, making it possible to resolve all three errors simultaneously. However, such mental feats should not be necessary. When a user changes the system, the result to both the current recognition as well as previous recognitions should be instantly clear. As mentioned previously, the integration of a timeline interface similar to the one used during user testing may help alleviate this problem. Alternatively, the system might simply inform the user of the degree to which their changes will affect other classifications (“these changes would have caused the system to make 25 different classifications in the past week”).
- **Fluid time scale** – users were generally unhappy with the fixed, 5-minute window and the 1-to-1 mapping of time to the y axis. One participant described the window as both “too short and too long at the same time.” Participants also had trouble calibrating their sense of time to the relative spacing of data objects on the vertical timeline. Data at the bottom of the screen was about 5 minutes old, while data at the top of the screen was brand new, but many participants treated all of the data as occurring simultaneously, or were confused when some data “felt” much farther away conceptually than it was represented on the timeline. A knowledge and representation of time still seems to be a critical feature of robust activity recognition, a strict, linear representation of this information to the user may not provide them with an intuitive understanding of what is recent and what is not. One solution may be to abandon a fixed timeline altogether and have object data slowly fade away as it becomes older. The status of “old” vs. “recent” data is still clear, but users are no longer encouraged to mentally cluster data based on spatial location. In a linear timeline, data that is close temporally may be totally unrelated, and vice-versa. However, the spatial proximity of co-occurring data in the interface encourages users to assume a relationship that may not exist. For example, the occupant of a home may adjust the thermostat and then immediately make a phone call. These two pieces of data would appear next to one another in the interface but are unlikely to be related from a recognition standpoint.
- **More varied restrictions** – while users were able to perform well in correcting the system, many of them did so without relying upon activity restrictions. Though initially attracted to them, users reported that both the time and posture restrictions were rarely helpful in modifying the *outcome* of the recognition.

However, users did not provide much information regarding what restrictions they *would* find useful. Further study is required. The following are possible restrictions that were considered for Distinguish but were discarded due to concerns of complexity, and are provided as possible future inclusions:

- **Duration** – Allow users to mark activities as having a minimum or maximum duration. Activities would not continue to be classified beyond their maximum duration. The current voting system does not support a way for the classifier to track “minimum” durations (a memory of when that activity was first classified but rejected would be necessary). A modification to both the recognizer and the interface would be necessary.
- **Extent of use** – Allow certain object associations to be marked with “heavy use” requirements. For example, *brushing teeth* involves heavy use of a toothbrush. If the system only detects singular or intermittent use, this may indicate another activity such as *organizing* or *cleaning the bathroom*.
- **Required objects** – Some activities require certain objects in order to take place. For example, *using the computer* requires a *computer*. It may involve other objects, such as *computer desk*, and *office chair*, but if the system does not see *computer* it should never recognize *using the computer*. Care should be taken; such a system is sensitive to incomplete sensor data.
- **Temporal order/prerequisites** – There are many activities that are inversions of one another, such as *coming home/leaving home*, or *getting dressed/undressing*. Such activities frequently involve similar if not identical object usage patterns, but can be distinguished by temporal ordering. For example, you can only *come home* if you were previously *outside*. Similarly, one must first *get dressed* before they can *undress*. In addition, some activities, while not inverses of one another, are ordered temporally. Usually, *doing the dishes* can only occur after *preparing a meal* has taken place. The ability to order activities or states of the environment temporally is an attractive type of restriction, but is difficult to represent graphically and is prone to noise problems (if *getting dressed* is not correctly identified, then *undressing* will never be recognized, even if there is good data to support it).
- **Mutually-exclusive activities** – Some activities cannot co-occur, frequently due to shared resources. It is unlikely that a participant would be *eating a meal* and *preparing a meal* at the same time, although these activities involve similar objects and can frequently occur in the same space. Similarly, it is difficult to both *read a book* and *talk on the phone* simultaneously. Allowing end-users to construct “X and Y cannot co-occur” statements would allow them to quickly resolve such confusions. Issues of temporal specificity quickly arise with such approaches, however. How close together can X and Y occur before they are considered to be co-occurring? Care should be taken to avoid situations where two exclusive activities vie for supremacy over a period of time, causing the system to alternately choose on and then the other as incoming data varies:

In considering these recommendations, future architects must carefully consider the dangers of adding additional complexity to the system. Many of these new restriction types



create serious challenges for the design of the user interface and/or recognition algorithm. Combined together, multiple additions may result in an unwieldy UI or a recognition system that is too complex for end-users to comfortably debug. As end-users become able to more precisely describe the minutiae of their activities, the burden of accounting for all possible corner cases and exceptions grows. The current design of Distinguish prefers an approach that is occasionally incorrect, but for obvious reasons, over one that is more frequently correct, but for non-obvious reasons.

### **Limitations of the study**

The user study represents a preliminary evaluation of the proposed approach to recognition, and as such was successful in achieving its stated goals. Users demonstrated that they understood the voting metaphor, that they were able to correct the system while reducing overall error, and that they were able to distinguish between situations of sensor error vs. recognition error. However, the experimental limitations, namely the need to simulate what users might do by watching another person, leaves many open questions regarding the overall experience of the occupants of a home equipped with an interface similar to that used by Distinguish. When end-users identify situations of sensor error, will they be able to improve the system by correctly adding new sensors or relocating existing sensors? Is an average correction time of approximately 2 minutes acceptable, or will users neglect to correct the system if it requires many interactions per day? How common will the “puzzle-solver” and “idealist” approaches be, and, according, how frequently will end-users have to deal with oscillating errors? Based on the manner in which test participants approached the data, these questions cannot be easily answered by participants who do not spend significant amounts of time interacting with the system in their own home. Many of the frustrations and payoffs from a context-aware home are only made clear after many hours of interaction. This preliminary study has explored the feasibility of the voting-round approach as well as the accessibility of the concept for end-users. Further study is required in order to verify that the occupants of the space are capable of improving their own recognition results (and that they enjoy doing so).

## Conclusion

Distinguish sacrifices the mathematical optimality of many supervised learning approaches for increased transparency for end-users. As a result, Distinguish is not dependent on large corpora of annotated training data – the system can function on data gathered from a relatively small web survey. Because of Distinguish’s data abstraction, data is portable between homes, removing the onus on homeowners to provide large corpora of custom, annotated training data. Distinguish also avoids problems of dimensionality, and allows for easy extensibility. Adding sensors or activities can be accomplished easily, and requires no retraining of the system (although the user may be required to add information to the common-sense database, this can be done in a matter of minutes). Finally, and perhaps most importantly, Distinguish provides an interface to the end-user that allows them to understand the behavior of their sensors and their activity recognition system.

Distinguish’s recognition performance is still poor compared to its competitors. However, Distinguish differs from other approaches in that it attempts to recognize a much more complete set of activities over a relatively long period of time. Recent work includes systems that attempt to classify six [18], eight [8], eleven [20], eleven [12], eleven [9], fifteen [13], twenty-four [23], and thirty-one [21] possible activities. By contrast, Distinguish classified data from a set of 51 possible activities (see Appendix B) and supports simultaneous and interleaved activities (which greatly magnifies the complexity of the task). The choice of activities was not arbitrary, but was based on the set of activities available in the annotated data. These annotations were developed independent of the system as part of the BoxLab project [36] and seek to represent a complete picture of common household activities.

Other work has already demonstrated the efficacy of traditional approaches on simplified examples of the home activity recognition problem. These simplifications frequently involve acted instead of naturalistic data, or an assumption that only a single activity can be taking place at any one time, or that only a single occupant is present in the home. By contrast, the Distinguish system explores what might be possible given the practical realities of current home sensing. Sensor data is undependable and noisy. Occupants perform a wide range of possible activities. These activities are broken up, interleaved, and frequently simultaneous. Activity varies from a 2-second duration “organizing” activity to multiple-hour computer use activities and 8-hour sleeping patterns. People rarely behave in ideal manners – they forget what they are doing mid-activity, or become distracted and switch seamlessly to another activity.

Distinguish’s main contribution is the inclusion of the end-user. Given all of the complications outlined in the previous paragraph, user testing demonstrated that end-users were able to understand the recognition system, detect when the system was unable to make a correct recognition due to insufficient data or another of the problems outlined above, and were able to iteratively improve the system over the course of the test. As such,

this work represents preliminary work in exploring the feasibility and benefits of including the end-user in recognition process. It has provided promising results with regards to end-users' abilities to understand the relationships between data and recognition. However, the quality of end-user participation and experience while living in an instrumented home over an extended period of time has yet to be evaluated, and makes up the bulk of remaining future work.

A user-centered approach promises to avoid many of the problems associated with pure machine-learning approaches. However, such an approach is also desirable from the standpoint of user experience. As technology and the end-user become more closely intertwined, it is important that the end-user remain in control of the behavior of systems that have the power to greatly affect their happiness or day-to-day behaviors [15]. The concept of the context-aware home is one of the most promising and most delicate examples of this situation. Such systems are designed to increase the capabilities of the end-user, but systems that are too complex for the user to understand or modify may very well end up removing the very sense of power and knowledge that they were designed to grant. On the other hand, a transparent system that supports the concepts of *usability*, *control*, and *iterability* returns that power of understanding to the user, enabling their context-aware systems to improve and expand his or her life.

# Bibliography

- [1] M. Romero, J. Summet, J. Stasko, and G. Abowd, "Viz-A-Vis: Toward Visualizing Video through Computer Vision," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, pp. 1261-1268, 2008.
- [2] M. C. Mozer, "Lessons from an adaptive house," in *Smart environments: Technologies, protocols, and applications*, D. C. R. Das, Ed., Hoboken, NJ: J. Wiley & Sons, 2005, pp. 273-294.
- [3] *Ekahau Wi-Fi Based Tracking Systems, RTLS, and WLAN Site Survey*. Available: <http://www.ekahau.com/> (Accessed: Aug. 11, 2010).
- [4] S. S. Intille, K. Larson, J. S. Beaudin, J. Nawyn, E. M. Tapia, and P. Kaushik, "A living laboratory for the design and evaluation of ubiquitous computing technologies," in *CHI '05 extended abstracts on Human factors in computing systems*, Portland, OR, USA, 2005, pp. 1941-1944.
- [5] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd, "At the Flick of a Switch: Detecting and Classifying Unique Electrical Events on the Residential Power Line," in *Proceedings of the 9th international conference on Ubiquitous computing*, 2007, pp. 271-288.
- [6] S. Katz, T. D. Downs, H. R. Cash, and R. C. Grotz, "Progress in development of the index of ADL," *Gerontologist*, vol. 10, pp. 20-30, 1970.
- [7] S. S. Intille and P. Kaushik, "Deploying context-aware health technology at home: Human-centric challenges," in *Human-Centric Interfaces for Ambient Intelligence*, H. Aghajan, et al., Eds.: Elsevier, 2009.
- [8] T. v. Kasteren, A. Noulas, G. Englebienne, and B. Krose, "Accurate activity recognition in a home setting," in *Proceedings of the 10th international conference on Ubiquitous computing*, Seoul, Korea, 2008, pp. 1-9.
- [9] P. Rashidi and D. J. Cook, "Activity Recognition Based on Home to Home Transfer Learning," in *Workshops at the 24th AAAI Conference on Artificial Intelligence*, 2010.
- [10] V. W. Zheng, D. H. Hu, and Q. Yang, "Cross-domain activity recognition," in *Proceedings of the 11th international conference on Ubiquitous computing*, Orlando, Florida, USA, 2009, pp. 61-70.
- [11] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille, "A long-term evaluation of sensing modalities for activity recognition," in *Proceedings of the 9th international conference on Ubiquitous computing*, Innsbruck, Austria, 2007, pp. 483-500.
- [12] D. J. Patterson, D. Fox, H. Kautz, and M. Philipose, "Fine-grained activity recognition by aggregating abstract object usage," in *Proceedings of the 9th IEEE International symposium on Wearable computers, 2005*, 2005, pp. 44-51.
- [13] D. Surie, F. Lagriffoul, T. Pederson, and D. Sjölie, "Activity recognition based on intra and extra manipulation of everyday objects," in *Proceedings of the 4th international conference on Ubiquitous computing systems*, Tokyo, Japan, 2007, pp. 196-210.
- [14] S. Davidoff, M. K. Lee, C. Yiu, J. Zimmerman, and A. K. Dey, "Principles of smart home control," in *Proceedings of the 8th international conference on Ubiquitous computing*, 2006, pp. 19-34.

- [15] L. Barkhuus and A. K. Dey, "Is context-aware computing taking control away from the user? Three levels of interactivity examined," in *Proceedings of the 5th international conference on Ubiquitous computing*, 2003, pp. 149-156.
- [16] V. Bellotti and K. Edwards, "Intelligibility and accountability: Human considerations in context-aware systems," *Human-Computer Interaction*, vol. 16, pp. 193-212, 2001.
- [17] A. K. Dey and A. Newberger, "Support for context-aware intelligibility and control," in *Proceedings of the 27th international conference on Human factors in computing systems*, Boston, MA, USA, 2009, pp. 859-868.
- [18] Z. Wang and B. Li, "Human activity encoding and recognition using low-level visual features," in *Proceedings of the 21st international joint conference on Artificial intelligence*, Pasadena, California, USA, 2009, pp. 1876-1882.
- [19] J. Fogarty, C. Au, and S. E. Hudson, "Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition," in *Proceedings of the 19th annual ACM symposium on User interface software and technology*, Montreux, Switzerland, 2006, pp. 91-100.
- [20] J. Modayil, T. Bai, and H. Kautz, "Improving the recognition of interleaved activities," in *Proceedings of the 10th international conference on Ubiquitous computing*, Seoul, Korea, 2008, pp. 40-43.
- [21] D. H. Hu, S. J. Pan, V. W. Zheng, N. N. Liu, and Q. Yang, "Real world activity recognition with multiple goals," in *Proceedings of the 10th international conference on Ubiquitous computing*, Seoul, Korea, 2008, pp. 30-39.
- [22] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised activity recognition using automatically mined common sense," in *Proceedings of the 20th national conference on Artificial intelligence*, Pittsburgh, Pennsylvania, 2005, pp. 21-27.
- [23] N. Landwehr, B. Gutmann, I. Thon, L. D. Raedt, and M. Philipose, "Relational Transformation-based Tagging for Activity Recognition," *Fundamenta Informaticae*, vol. 89, pp. 111-129, 2009.
- [24] F. Kawsar, T. Nakajima, and K. Fujinami, "Deploy spontaneously: supporting end-users in building and enhancing a smart home," in *Proceedings of the 10th international conference on Ubiquitous computing*, Seoul, Korea, 2008, pp. 282-291.
- [25] S. Amershi, J. Fogarty, A. Kapoor, and D. Tan, "Overview based example selection in end user interactive concept learning," in *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, Victoria, BC, Canada, 2009, pp. 247-256.
- [26] J. H. Jahnke, M. d'Entremont, and J. Stier, "Facilitating the programming of the smart home," *Wireless Communications, IEEE*, vol. 9, pp. 70-76, 2002.
- [27] K. Truong, E. Huang, and G. Abowd, "CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home," in *Proceedings of the 6th international conference on Ubiquitous computing*, 2004, pp. 143-160.
- [28] J. Humble, A. Crabtree, T. Hemmings, K.-P. Åkesson, B. Koleva, T. Rodden, and P. Hansson, "'Playing with the Bits' User-Configuration of Ubiquitous Domestic Environments," in *Proceedings of the 5th international conference on Ubiquitous computing*, 2003, pp. 256-263.
- [29] *X10 ActiveHome Pro Home Automation Products*. Available: <http://www.x10.com/> (Accessed: Aug. 2, 2010).

- [30] *MisterHouse*. Available: <http://misterhouse.sourceforge.net/> (Accessed: Aug. 2, 2010).
- [31] A. Woodruff, S. Augustin, and B. Foucault, "Sabbath day home automation: "it's like mixing technology and religion"," in *Proceedings of the 25th SIGCHI conference on Human factors in computing systems*, San Jose, California, USA, 2007, pp. 527-536.
- [32] A. K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu, "a CAPpella: programming by demonstration of context-aware applications," in *Proceedings of the 22nd SIGCHI conference on Human factors in computing systems*, Vienna, Austria, 2004, pp. 33-40.
- [33] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer, "Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition," in *Proceedings of the 25th SIGCHI conference on Human factors in computing systems*, San Jose, California, USA, 2007, pp. 145-154.
- [34] M. Mittal, "Ubicorder: A Mobile Interface to Sensor Networks," SM, Program in Media Arts and Sciences, Massachusetts Institute of Technology, Cambridge, 2009.
- [35] Y. Li and J. A. Landay, "Activity-based prototyping of ubicomp applications for long-lived, everyday human activities," in *Proceeding of the 26th annual SIGCHI conference on Human factors in computing systems*, Florence, Italy, 2008, pp. 1303-1312.
- [36] *BoxLab Home Sensing Project*. Available: <http://boxlab.wikispaces.com/> (Accessed: Aug. 3, 2010).



# Appendix

## Appendix A - User interface prototypes

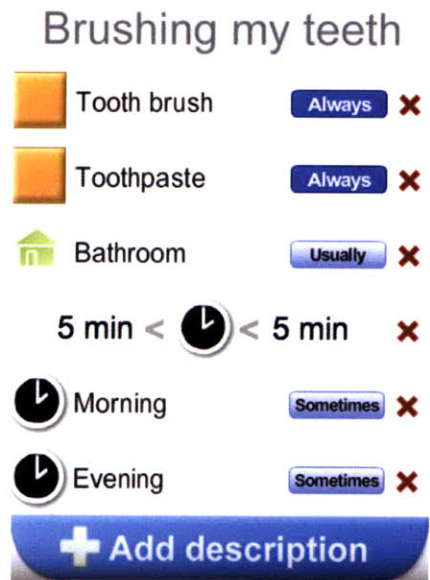
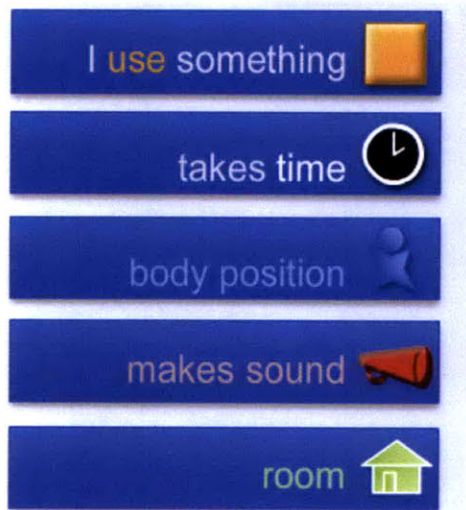
The design of the interface underwent a number of evolutions during the course of development as the knowledge representation changed from free-form text to a traditional model system to the final hybrid voting system.

### Preliminary survey - free-form text entry

The screenshot shows a user interface for a 'making breakfast' task. The title 'making breakfast' is displayed in green at the top center. In the top right corner, the progress indicator '0/15' is shown. On the left side, there is a blue 'Back' button with a left-pointing arrow. The main area contains a list of text input fields, each with a blue underline: 'refrigerator', 'frying pan', 'cup', 'orange juice', and 'fresh bre'. Below these fields are two more empty input lines. At the bottom left, a green instruction reads '[SHIFT] + [ENTER] when done, or'. To the right of this instruction is a blue 'Save' button. In the bottom right corner, the MIT logo is visible.

### Pick-and-choose survey with specific types of data

Note the presence of “always” and “sometimes” designations.



Refined pick-and-choose survey with a focus on objects

**Arriving home**

Takes at least:

2 min

Takes at most:

5 min

**+** Improve definition

Previous activity

Next activity

## Arriving home

Takes at least:

Takes at most:

I use something

room

body position

time of day

makes sound

Cancel

## Arriving home

Takes at least:

Takes at most:

Select from below or narrow it down:

Q |

example: clothing chair, or cheese

Address book

Aerosols

Aftershave

Air conditioner

Air freshener

Air pump

Air purifier

Done

## Arriving home

Takes at least:

Takes at most:

You can select more than one

Early morning

Morning

Noon

Afternoon

Evening

Night

Overnight

Cancel

## Making breakfast

Takes at least:

Takes at most:

10 min

15 min

### + Improve definition

Objects	Frequency	Action
Bowl	almost always	x
Cereal	almost always	x
Eggs	sometimes	x
Frying pan	sometimes	x
Posture		
Standing		x
Time		
Early morning		x
Morning		x

Final survey

### eating a meal

When does this usually happen?  
*You can select more than one*

Early morning
<b>Morning</b>
Noon
Afternoon
<b>Evening</b>
<b>Night</b>
Wee hours
Any of these

Next

**chair** → eating a meal  
→ studying  
→ working

When do you use or touch a **chair**?

Q wa

Hygiene
Washing face
Washing hands
Liesure
Watching TV
Sleep related
Waking up
Add your own...

Next

## **Appendix B - All activities used in survey (total: 51)**

Note: Activities are broken up into categories. If an activity has one or more synonyms, they follow in a comma-separated list.

For each activity present, three questions were asked: what rooms it can take place in, what time of days it can occur during, and what postures are involved.

food prep:

- preparing a meal, cooking
- getting/preparing a snack
- getting/preparing a drink
- setting the table

cleaning:

- doing the dishes, dishwashing
- putting the dishes away

- sweeping
- mopping
- vacuuming
- throwing something away
- taking out the trash
- clearing the table
- cleaning the bathroom

organizing:

- making the bed
- general organizing
- putting groceries away
- adjusting the light or temp

laundry:

- doing laundry
- ironing
- folding clothes
- putting clothes away

hygiene:

- washing hands
- washing face
- cleaning teeth
- flossing
- taking a shower
- going to the bathroom, toileting
- healthcare

entering/leaving:



coming home  
leaving home

preparation:

getting dressed, putting on clothes, clothing

grooming:

shaving  
hair care

leisure:

watching TV  
reading  
dancing  
listening to music  
working on a hobby  
relaxing

exercise:

exercising

eating:

eating a snack  
drinking, taking a drink  
eating a meal

sleep related:

sleeping  
taking a nap  
getting ready for bed, going to bed  
waking up, getting up

social:

talking on the phone  
hanging out

work:

working, doing work  
studying  
writing  
using a computer

### **Appendix C - All object questions used in survey (total = 108)**

For each object, participants were asked "When do you use or touch a(n) <object>?"

eyeglasses  
clothes iron

stove  
contact lens solution  
clothes dryer  
pepper shaker  
oven  
ottoman  
chair  
shopping cart  
dish detergent  
blender  
large bowl  
broom  
cookie jar  
tape  
plate  
ice dispenser  
food processor  
spice rack  
mop  
fan  
cushion  
desk  
toilet plunger  
toolbox  
window shade  
arm chair  
coffee maker  
shop vacuum  
toiletry bag  
lightswitch  
window blinds  
toilet brush  
mouthwash  
trash can  
clothes hanger  
pantry  
microwave  
desktop computer  
vitamins  
closet  
alarm clock  
ice  
detergent  
mug  
dish drying rack  
dust pan  
shoe rack

remote control  
faucet  
side table  
office supplies container  
dishsoap  
laptop computer  
watering can  
pillow  
thermostat  
armchair  
laundry hamper  
table  
recycling bin  
filing rack  
can opener  
answering machine  
front door  
beard trimmer  
dishwasher  
wardrobe  
fabric softener  
television  
storage box  
nightstand  
shower curtain  
toilet  
electric shaver  
telephone  
drawer  
paper towel  
towel rack  
key cup  
washing machine  
toaster  
bowl  
freezer  
couch  
cd rack  
lamp  
hat rack  
paperclip holder  
dresser  
printer  
vacuum cleaner  
hair dryer  
air pump  
notebook

toilet paper  
audio speaker  
refrigerator  
water dispenser  
pen & pencil holder  
stereo  
salt shaker  
tea maker  
hand soap  
phone charger  
garbage disposal  
knife block

## Appendix D - Total annotated activities from the source home

Note: some activities have slightly different names than those used by Distinguish.

406 out of view  
399 searching for an item  
323 using a computer  
242 grooming  
206 organizing  
122 using a mobile phone/pda/mp3 player (other than talking)  
95 eating  
80 throwing something away  
52 home management [miscellaneous]  
47 putting away clothes  
47 preparing a snack  
36 dishwashing  
35 outside  
34 leaving the home  
33 entering the home  
30 drinking water or a beverage  
30 making the bed  
28 hygiene [miscellaneous]  
27 cleaning [miscellaneous]  
25 relaxing/thinking  
25 preparing a meal  
24 reading printed material  
20 leisure [miscellaneous]  
20 eating[background]  
19 sleeping deeply  
18 cleaning teeth  
16 wiping  
16 study-related [miscellaneous]  
15 washing hands  
14 waking up for the day  
14 preparing a drink  
14 talking on a phone  
14 napping  
12 study-related  
11 getting ready for bed  
10 doing laundry  
10 preparing a meal [background]  
9 washing face  
8 watching tv/movie/video content  
7 dining area  
7 dancing  
7 cooking or warming food in microwave

5 watching tv/movie/video content[background]  
5 healthcare  
4 exercising  
4 sweeping  
4 cooking or warming food on stovetop  
3 dishwashing[background]  
3 doing laundry[background]  
3 drinking[background]  
3 writing  
3 vacuuming  
2 drinking [background]  
2 laundry [miscellaneous]  
2 dishwashing [background]  
2 cooking or warming food in oven  
1 information[miscellaneous]  
1 dishwashing[miscellaneous]  
1 working on a hobby  
1 listening to music or radio[background]  
1 scrubbing  
1 bathing or showering



## Appendix E – User study tutorial script

First of all, thank you for participating. Today you'll be asked to evaluate a user interface.

Our lab works with “smart” environments, which are places like homes or offices that have been outfitted with a variety of small sensors. Using the data from these sensors, we can do a lot of useful things – the home can adjust the lighting and heating when you move around, it can suggest recipes if it detects that cooking is taking place, it can automatically send phone calls to voice mail if it detects that you're watching a movie, it can remind you to exercise if you haven't done so in a while, etc. etc.

In order to accomplish all these things, the system needs to have an idea of what's going on at any one moment. To do this, it reads information from the sensors and then uses that to make a guess as to what's happening. This is called *activity recognition*.

We've built a system that tries to accomplish this, but right now it makes a lot of mistakes and needs some help from humans in order to fix them. The interface that you're going to evaluate is designed to do just this – allow inhabitants of a smart home to fix the activity recognition system when it makes a mistake. For example, if the system accidentally thinks that you're watching TV when you're actually studying, you should be able to use this interface to fix things so it doesn't make that mistake in the future.

All told, this will take about an hour. First I'll walk you through a tutorial on how to use the interface, then the test will start. You'll be shown a series of situations where the system made a mistake and guessed the wrong activity. We'll evaluate the interface based on how easy or hard it is for you to correct these mistakes.

It's important to remember that this is a test of the interface, not of you. If something is confusing or you don't know what to do, then the interface, or our design of the interface, is at fault, not you. So, don't get discouraged if you get stuck, just describe out loud what's giving you problems.

A quick note about our sensors: most of them are object usage sensors that detect when certain objects are moved. However, they can't tell the difference between movements, so they can't tell the difference between brushing your teeth vs. washing your toothbrush. They also can't tell the difference between sitting down in a chair or just bumping it as you walk by – just that the chair got moved. If you shake the table, all of the objects on the table will report being moved. Usually this isn't much of a problem, but sometimes it will cause the sensors to report movement in unexpected ways.

All right, let's start the tutorial.

On the screen is the interface. Objects that have been moved recently appear on the left side of the screen. As time passes, they slowly fall down like Tetris pieces until they slide off the bottom of the screen. If the system has a guess about what's taking place, it will appear on the right side of the screen.

So, for example, if someone moved a plate recently, the interface would look like this:

<PLATE>

The system keeps a list of what objects are used for what activities. For example, a plate might be involved with “eating a meal” or “doing the dishes.” At the moment, “plate” is colored grey, which means that the system doesn’t have any information about what “plate” objects are used for. Since that’s our only piece of data, the system doesn’t have any guesses.

Let’s associate some activities with “plate” so the system knows what might be going on.

First, click “Add activity” on the right side of the screen. This pops up a list of all of the possible activities the system supports. Scroll down and expand the “eating” category. Now click on “eating a meal.” “Eating a meal” appears on the right, but it’s also grayed-out. Why? Because there’s no evidence to indicate that it’s taking place.

In order to associate “plate” with “eating a meal,” draw a line from one to the other.

<WAIT>

Now that the system has some information about “plate.” Specifically, it thinks that when it sees plate, it should guess that “eating a meal” is taking place. Notice how both the color changes from grey and “eating a meal” gets a star.

You can remove lines just as easily. Just click on them – do this now for the one you just created.

<CLICK>

Things go back to being grey – the system doesn’t have any information about “plate” to work with. Add the line back in.

Now, “eating” isn’t the only activity that involves plates. Can you think of any other ones?

<LIST>

Great. We should probably tell the system about these activities too. Click on “add activity,” and add “doing the dishes.” (you can browse from the list below or just type in the name in the search field above).

Great. Now draw another line from “plate” to “doing the dishes”.

Now there’s a tie between “doing the dishes” and “eating a meal” (they both have one “vote”), so the system can’t decide what’s taking place. Notice how the color changes to brown (the color of ties) and the star turns into a question mark.

All right, let’s add one more activity: “setting the table.”

Now we have a three-way tie! The system really can't decide what's going on. However, if we humans tried to guess what was happening right now with just this one piece of information, we'd have the same problem. It could really be any of these things. We just don't have enough information.

What if, after moving the plate, someone also moved a sponge? It might look something like this:

<SPONGE>

Notice that sponge is gray, meaning that the system doesn't know anything about what sponges are used for. Now, what are some activities that involve a sponge?

<DOING THE DISHES>

Great. Draw a line from sponge to "doing the dishes."

We have a winner again! Doing the dishes now has two votes instead of everyone else's one, so the system selects it as the winner. You can see what objects are voting for doing the dishes by clicking on it. (do that now)

<CLICK>

But if you click on "eating a meal" ...

<CLICK>

Only one supporter.

You can also see what activities an object votes for by clicking on that object. So, first click on "plate"

<CLICK>

(here are the things it voted for earlier)

And then on "sponge"

<CLICK>

Sponge only votes for one activity. We could add more if we wanted to.

All right, that's the basic approach the system uses to try to recognize what's taking place. Recent data votes for possible activities, and the activity with the highest score is chosen.

Usually, you don't have to manually add in all these associations between objects and activities. The system has most of them built-in, but its lists are very imperfect. Some objects are missing some activities, some have too many. Some have the completely wrong set of activities associated with them. Essentially what you'll be doing for the rest of the test is modifying these associations so they give the system a better idea of what's going on.

There's one last bit of information that the system uses to improve its guessing. Imagine that someone has been moving a bar of soap recently:

<HAND SOAP>

This could indicate a number of different activities. Click on soap to see which ones it's associated with. Without any more information, we can't pick one over the other. However, what if we knew that this activity was taking place in the kitchen? Can we eliminate any of these possibilities?

<YES, TAKING A SHOWER>

Great. In addition to knowing what objects have been moved recently, the system also knows the WHERE, WHEN, and HOW of what's taking place.

<screen with WWH>

At the top of the screen is a display of the current room, the time of day, and what postures have taken place in the last five minutes (standing, sitting, lying down, etc.)

Notice that, now that the system has this extra information, it can cross off some options. Click on "taking a shower" to see why it's been removed from the list.

<CLICK>

These icons represent restrictions on room, time of day, and posture, respectively. Notice that the "room" icon is red, meaning that "taking a shower" can't take place in the current room (kitchen).

You can modify these restrictions just like you can modify the objects. Click on the room icon to modify what rooms "taking a shower" can take place in.

<CLICK>

Rooms where "taking a shower" can take place are in the "YES" circle. So, at the moment, "taking a shower" can only take place in the bathroom (surprise). Notice that the *current* room - "kitchen" is in the "NO" circle, so the restriction is being violated. You need at least one glowing value in the "YES" circle in order to pass the test.

For practice's sake, move kitchen into the YES circle

<MOVE>

Notice that the circle lights up to represent that the restriction's been passed now. Close the window to go back to the main screen.

<CLOSE>

Now the system is perfectly happy to guess that taking a shower takes place in the kitchen, and we're back to having a tie. This doesn't make a lot of sense, so go back and force "taking a shower" to only take place in the bathroom.

<UNDO>

All right, that's the basic principal behind the system. Data votes for activities, impossible activities are removed, and then a winner is chosen. If system makes a mistake, you have two methods to fix it: you can change that votes, or you can change the restrictions (or both).

Now, before we start the test we're going to do two short examples for practice.

<INTRODUCE TWO EXAMPLES. CHALLENGE USER TO CORRECT THEM. PROVIDE HELP AS NECESSARY>

## **Appendix F – Study debriefing questions**

1. In your own words, please explain how the activity detection system works.
2. Did you have a specific strategy when attempting to fix errors? (describe)
3. Do you think that the voting and restriction system was too simple or too complex? (why)
4. When fixing errors, on average did you feel like you understood or didn't understand why those errors were taking place? (why)
5. Did you ever encounter a situation where you felt that it was obvious that a particular activity was taking place but were unable to get the system to make the right decision? (describe)
6. Did you encounter a situation where a change you made caused an error somewhere else in the timeline? Were you able to fix that error as well?
7. What could be done to make the activity correction easier to do or to understand?
8. What was the most difficult situation you experienced while trying to correct the recognition?
9. What was the most confusing or frustrating aspect of the interface?
10. Any other comments?

## **Appendix G – Study advertisement**

MIT researchers are designing ways of making technology in the home easy to use and customize. We need participants to test out a new interface we have designed that gives people greater control over the way the technology in their home behaves.

Participation will last approximately 1 hour. In return you will receive \$10.

Interested? Learn more by contacting Ned Burns at <number> or [pixel@mit.edu](mailto:pixel@mit.edu).

## Appendix H – Confusion Matrix

The following table represents a modified confusion matrix of the system’s recognition performance. A traditional confusion matrix displays how frequently the classifier confuses one label for another. For example, a single cell of a matrix might represent how often the classifier specified “doing the dishes” when “eating a meal” was actually taking place. However, traditional confusion matrices are hard to generate for this particular application for a number of reasons, including the overlapping nature of activities, as well as the 5-minute window employed by the recognition system. When the system makes a classification error, it is difficult to calculate which activity it *should* have specified if there are multiple activities taking place simultaneously, or if a number of short activities have taken place one after another so that they all appear in the last five minutes of data. One option is to simply count *every* recent activity as confused. However, this makes it difficult to compare against correct recognition rates. A single correct recognition will only increase its cell value by 1, but a single incorrect recognition might increase many cell values by one, giving the impression that many more errors are occurring than was the case. Instead, the following confusion matrix is “weighted” – whenever a recognition error takes place, the confusion cells of all recent activities are incremented by  $1/N$  where  $N$  is the total number of recent activities. Thus, a single recognition error will still only increase the total number of confusions by one. This allows the reader to compare correct recognitions vs. confusion counts.

In order to fit on the page, the matrix has been transposed from its traditional orientation. Labels generated by the system are listed across the top row of the matrix. A column represents all of the confusions for a particular system-generated label. For example, the “cleaning teeth” *column* lists the number of times the system produced the “cleaning teeth” label when some other activity or activities were actually taking place. In addition, note that the “predicted” and “actual” labels differ slightly. “Actual” labels come from BoxLab annotations. Some of these labels were too verbose or too general in scope for end-user interaction, and so were renamed or broken down into a set of more specific activities<sup>20</sup>. However, because the resulting matrix does not have matching sides, correct recognition counts do not appear along the diagonal. These have been reproduced on the first row of the matrix.

---

<sup>20</sup> In addition, some BoxLab activities are broken into multiple versions, indicated with additional [miscellaneous] and [background] tags. During evaluation, these activities were treated as a single activity for the purposes of detecting errors.

# Recognition Confusion Matrix

<b>Actual</b>	<b>Predicted</b>	
	cleaning teeth	cleaning the bathroo
correct	0	4
bathing or showering	0	1.3
cleaning [miscellaneous]	0	0
cleaning teeth	0	0
cooking or warming food in microwave	0	0.1
cooking or warming food in oven	0	0
cooking or warming food on stovetop	0	0
dancing	0	0.1
dishwashing	0	0.3
dishwashing[background]	0	0
dishwashing[miscellaneous]	0	0
doing laundry	0	0
doing laundry[background]	0	0
drinking [background]	0	0
drinking [background]	0	0
drinking water or a beverage	0	0
eating	0	1.7
eating[background]	0	0
entering the home	0	0.8
exercising	0	0
getting ready for bed	0.3	0
grooming	4.2	4.9
healthcare	0	0
home management [miscellaneous]	0	0.4
hygiene [miscellaneous]	1.4	3.9
information[miscellaneous]	0	0
laundry [miscellaneous]	0	0
leaving the home	0	1
leisure [miscellaneous]	0	0
listening to music or radio[background]	0	0
making the bed	1.9	0.9
napping	0.1	1
organizing	0.7	1.1
out of view	9.3	9.8
outside	0	0.8
preparing a drink	0	0
preparing a meal	0	0.1
preparing a meal [background]	0	0
preparing a snack	0	0.8
putting away clothes	4.4	0
reading printed material	0	1.3
relaxing/thinking	0.3	0
scrubbing	0	0
searching for an item	4	2.7
sleeping deeply	0	0.4
study-related	0.1	0
study-related [miscellaneous]	0	0
sweeping	0	0
talking on a phone	0.5	0
throwing something away	0	0.2
using a computer	1.5	2.7
using a mobile phone/pda/mp3 player (other than talking)	1	0
vacuuming	0	0
waking up for the day	0	1.5
washing face	0.5	0
washing hands	0	0
watching tv/movie/video content	0	0
watching tv/movie/video content[background]	0	0
wiping	0	0
working on a hobby	0	0
writing	0	0



doing laundry	doing the dishes	eating a meal	eating a snack	general organizing	getting ready for bed
0	21	5	0	0	5
0	0	0	0	0	0
0.5	0	2	0	0	0
0.1	1.1	0.4	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	1.7	0.7	0	0
0	2.3	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0.1	0	0	0	0.2
0	0.1	0	0	0	0
0	0.3	0	0	0	0.3
0	0.4	0	0	0	0
0	0	0	0	0	0
0	0	1.7	0	0	0.1
0	0	0	0	0	0
0	0	1	0	0	0
4.1	0.5	4	0	0.4	0
0	0	0	0	0	0
0	0	0.1	0	0	0
0.4	0	1.1	0	0	0
0	0	0.9	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	2.5	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1.7	0.4	5.9	1	0	2
3.5	1.5	8.9	0	3	3.8
5	100	4.2	0	0	0.1
0	0.3	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0.1	0	0.3	0	0
0	0	1.5	0	0.8	0
0	0	2.9	0	0	0
0.4	0	0	0	0	0
0	0	0	0	0	0
3.9	0.7	7.4	1.9	0.4	1.9
0	0	0	0	0	8
0	0	0.2	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0.8	0	0	1.7
0	0	1.2	0	0	0
1.1	3.5	4.4	0.9	0.4	4.1
0	0	2.3	0	0	3.7
0	0	2.1	0	0	0
0	0	0	0	0	0
0.2	0	1	0	0	0
0	0	0	0.1	0	0
0	0	0.9	0	0	0
0	0	0.9	0	0	0
0	0	0.9	0.1	0	0
0	0	2	0	0	0
0	0	0	0	0	0

	getting/preparing a c	getting/preparing a s	going to the bathroo	healthcare	ironing	listening to music
	5	5	20	0	0	0
	0	0	0	0	0	0
	0.6	0	0.8	0	0	0
	0	0	2.3	0	0	0
	0.1	0.6	0	0	0	0
	0.1	0	0	0	0	0
	0	0	0.5	0	0	0
	0	0	0	0	0	0
	0.9	0.6	2.8	0	0	0
	0	1.3	0.8	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0.6
	0	0	0.1	0	0	0
	0	0	0.1	0	0	0
	1	0	1.3	0	0	0
	3.1	0.2	2.9	0	0	0
	0	0	0	0	0	0
	0	0	1.4	0	0	0
	0	0	0	0	0	0
	0	0	0.8	0	0	0
	0.5	0	4.6	2.5	0	0
	0	0	0	0	0	0
	0	0	0.7	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0.1	0	0	0
	0	0	3.1	0	0	0
	0	0	0	0	0	0
	0	0	0.7	1.1	0	0
	0	0	0.3	0	0	0
	3.3	0	4.2	0	1.8	0.6
	2	0	48	0	0	1
	0	0	3.9	5	0	0
	0	0	0.6	0	0	0
	0.7	0.6	0.7	0	0	0
	0.1	0	0.5	0	0	0
	0.9	0	0.5	0	0	0
	0	0	2.7	0.5	3.2	0
	0	0	1.5	0	0	0
	0.4	0	0.1	0	0	0
	0	0	0	0	0	0
	3.6	0.9	8.7	0.7	0	0.6
	0.4	0	0.7	10	0	0
	0	0	0	0	0	0
	0	0.5	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0.9	0.3	0.8	0.1	0	0.3
	2.9	2.1	7.4	0	0	0
	0	0	4.9	0	0	0
	0	0	3.6	0	0	0
	0.5	0	0	0	0	0
	0	0	0.7	0	0	0
	0.1	0	0.6	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0.8	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0

	preparing a meal	putting clothes away	putting the dishes away	reading	shaving	sleeping	
	32	23	21	0	13	65	
	0	0	0	0	0	0	
	0.9	0	0.6	0.9	0	0	
	0	0	0	0	0	0.1	
	0	0	0	0	0	0	
	0	0	1	0	0	0	
	0	0	0	0	0	0	
	1.4	0	0.8	0	0	0	
	0	0.9	11.1	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	1	0	0	0	0	0.9	
	3.5	0	3	0	0	0	
	0	0	0.8	0	0	0	
	1	0	0	0	0	0.4	
	0	0	0	0	0	0	
	0	0	0	0	0	2.8	
	1	0.4	3.3	0	0	4	
	0	0	0	0	0	0.7	
	0.7	0.9	0	0	0	1	
	0.1	0	0.7	0	0	0.7	
	0	0	0	0	0	0	
	0.8	0	0	0	0	0	
	1.3	0	0.3	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	0	0	0.5	0	0	0.2	
	0	0	0.4	0	0	0	
	2.6	0.7	0	0	0	1.6	
	2.4	6	11.6	0	2.8	13.8	
	6.6	2.5	12.3	0	0	2.4	
	0	0	0	0	0	0	
	0	0	2.1	0	0	0	
	0	0	0.8	0	0	0	
	2.6	0	0.1	0	0	0	
	0	0	0	0	0	0.5	
	0	0	0	0	0.5	0	
	0	0	0	0	0	0	
	5.6	3.2	7.2	0	0.3	5.6	
	0	0	0.6	0	0	0	
	0	0	1.4	0	0	0.8	
	0.1	0	0	0	0	0	
	0.8	0	0	0	0	0	
	0	0	0	0	0	27.7	
	2.4	0	1.1	0.9	0	0.6	
	5.7	0.8	6.8	1.9	1.5	11.2	
	0.1	0.6	0.4	0	1	6.1	
	0.6	0	0	1.4	0	0	
	0	0	0.7	0	0	0	
	0	0	0	0	0	0.1	
	0.6	0	0.7	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	
	0.1	0	0	0	0	0	
	0	0	0	0	0	0	
	0	0	0	0	0	0	

studying	sweeping	taking a nap	taking a shower	throwing something	using a computer	
26	0	0	0	0	0	744
0	1.1	0	0	0	0	0
0	0.7	0.9	0	1.7	0	0.1
0	0.2	0.4	0	0	0	1.3
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	1.7	0	1.1
0	0	0	0	4.2	0	0.7
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0.3	0
0.8	0	0	0	0.5	0.3	5.1
0	0	0	0	0	0	0
0	2.1	1.3	0	0	0	1.3
0	0	0	0	0	0	0
0	0	0	0	0	0.7	1.2
1.6	6.8	2.3	12.1	0	0	3.4
0	0	0	0	0	0	0
0.1	0.1	0.3	0	0	0	0.3
0	2.5	0.4	5.3	0	0	1.6
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0.1	0	0	0	1
0	1.1	0.9	0	0	0	0
0	0	0	0	0	0	0
0	0.8	0.8	0.8	0.8	0	0
0	1.2	0	0	0	0	0
1	2.3	4.9	1.1	0.7	0	4.7
1.3	13.7	1.3	33.7	8.1	0	123.2
0	7.1	1.4	0	0	0	4.4
0	0	0	0	0	0	0.8
0.8	0	0	0	0	0	1.5
0	0	0	0	0	0	0
0	0	0	0	0	0	1.1
0	3.7	0	0.3	0	0	0.9
0	1.8	0	0	0	0	0.9
0	0.3	4	0	0	0	0
0	0	0	0	0	0	0
3.4	4.5	4.2	3.8	0.6	0	4.5
5.2	0	0	2.3	0.6	0	0.2
0	0	0	0	0	0	0.5
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0.2	0	1.6	0.5	0	0	2.5
0	7.9	20.4	3.3	3	0	0
0.3	1.4	0.5	1.3	0	0	2.1
0	1.4	6.3	0	0	0	13.3
0.3	1.5	0	5.6	0	0	0.3
0	0.5	0	0	0	0	0.7
0	0	0.2	0	0	0	0.2
0	0	0	0	0	7.3	0
0	0	0	0	0	7.3	0
0	0.4	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

waking up	washing hands	watching TV	
17	10		66
0.6	0		0
0.5	0		0
0	3.8		0
0	0		0
0	0		0
0	0		0
0.2	0		0
0.1	0		0
1.8	0		0
0	0		0
0	0		0
0	0		0
0	0		0
0	0		0
0.1	0		0
0	0		0
0	0		0
0	0		0
0	0.1		0
27.5	1.5		0
0	0		0
0.9	0		0
2.9	0		0
0	0		0
0	0		0
0.3	0		0
0	0		0
0	0		0
7.1	0		0
14.3	0		0
6.1	0.7		0
16.1	14.2		0
0.3	0		0
0	0		0
0	0		0
0	0		0
5.4	0.2		0
0	0		0
5.5	0		0
0	0		0
10.2	1		0
0	0		0
0	0		0
0	0		0
0	0		0
1	0.9		0
17.1	1.1		0
4.8	0.6		0
0	0		0
0	0		0
0	1.7		0
0	0		0
0	0		0
0	0		0
0	0		0
0	0		0
0	0		0