Enabling Physical Analytics in Retail Stores Using Smart Glasses

Swati Rallapalli
UT Austin
swati@cs.utexas.edu

Aishwarya Ganesan
Microsoft Research India
t-aigane@microsoft.com

Krishna Kant Chintalapudi
Microsoft Research India
krchinta@microsoft.com

Venkata N. Padmanabhan
Microsoft Research India
padmanab@microsoft.com

Lili Qiu
UT Austin
lili@cs.utexas.edu

ABSTRACT

We consider the problem of tracking physical browsing by users in indoor spaces such as retail stores. Analogous to online browsing, where users choose to go to certain webpages, dwell on a subset of pages of interest to them, and click on links of interest while ignoring others, we can draw parallels in the physical setting, where a user might *walk* purposefully to a section of interest, *dwell* there for a while, *gaze* at specific items, and *reach out* for the ones that they wish to examine more closely.

As our first contribution, we design techniques to track each of these elements of physical browsing using a combination of a first-person vision enabled by smart glasses, and inertial sensing using both the glasses and a smartphone. We address key challenges, including energy efficiency by using the less expensive inertial sensors to trigger the more expensive vision processing.

Second, during gazing, we present a method for identifying the item(s) within view that the user is likely to focus on based on measuring the orientation of the user's head.

Finally, unlike in the online context, where every webpage is just a click away, proximity is important in the physical browsing setting. To enable the tracking of nearby items, even if outside the field of view, we use data gathered from smart-glasses-enabled users to infer the *product layout* using a novel technique called *AutoLayout*. Further, we show how such inferences made from a small population of smart-glasses-enabled users could aid in tracking the physical browsing by the many smartphone-only users.

Categories and Subject Descriptors

C.2.m [**Computer Systems Organization**]: COMPUTER - COMMUNICATION NETWORKS - Miscellaneous

General Terms

Algorithms, Design, Experimentation

Keywords

Physical Analytics, Wearables, Smart Glasses, Localization, WiFi

MobiCom'14, September 7–11, 2014, Maui, HI, USA.

http://dx.doi.org/10.1145/2639108.2639126.

1. INTRODUCTION

Web analytics is a multi-billion dollar business today, providing tools that measure and analyze users' online browsing behavior to help companies assess the effectiveness of their websites. A typical tool will track webpage-related events, such as page views, clicks, and the degree to which different landing pages are associated with online purchases.

Most purchases today, however, occur in the physical realm, *e.g.*, at retail stores and shopping malls. While many retailers track shoppers' purchases (*e.g.*, as part of customer loyalty programs) in order to perform analytics, these systems do not provide insight into shoppers' behavior *during* the shopping process, which we refer to as *physical browsing* in this paper. For example, which sections of the store did the shopper spend most of his or her time in? Which products did the shopper express interest in by gazing at them or reaching out for them? How many shoppers reached out for competing products A and B, say to compare these?

We believe that access to physical browsing information of shoppers in retail stores can not only provide crucial insights into shoppers' needs and interests but also reveal the effectiveness of the store layout itself. Furthermore, such *physical analytics* that track in-store shopping behavior could also be combined with online information such as web browsing history or a shopping list to generate automatic alerts, say when there is a match between the physical browsing context and the online information.

One approach would be for the retail stores themselves to use their in-store surveillance videos to mine this information. However, many retail stores today neither possess the expertise nor the resources to gather and analyze tons of surveillance data for this purpose. Furthermore, this approach would *not* allow aggregating data across a user's online and physical browsing or indeed even just physical browsing across different stores that a user may visit since an individual store is unlikely to be willing to share information with potential competitors. On the other hand, the advent of smart glasses such as Google Glass [23], equipped with a camera, inertial sensors and WiFi, enables user-driven data gathering, both (privately) tracking an individual user's visits across stores and crowd-sourcing store layout information based on the visits by a community of users. Furthermore, from the viewpoint of an individual user, there would be an incentive to participate since their physical browsing information could be used directly for their own benefit (*e.g.*, for receiving personalized alerts from a digital personal assistant) rather than being shared for the stores' business purposes. For it to scale to a wide footprint, such a user-driven approach should ideally *not* depend on any support from the dis-

parate stores that the user may visit. Therefore, in this paper we ask the question, *"How can we develop a system that leverages smart glasses to track physical browsing, without depending on the store itself for any a priori information (*e.g.*, store layout or product information) or infrastructure support, and without requiring any explicit input from the shopper?"*

Further, while smart glasses seem poised for rapid growth and could even supplant smartphones down the road, for the foreseeable future it is likely that most shoppers will only be carrying smartphones. Thus, an important question is *"How can we use the data from the few smart glasses enabled users to track physical browsing by the large number of shoppers only carrying smartphones?"*

The ThirdEye system: In this paper, we present a system called ThirdEye to address the above questions. ThirdEye only uses image, inertial sensor, and WiFi data crowd-sourced from shoppers wearing smart glasses to track the physical browsing of shoppers, without requiring any input from either the store owners or from the shoppers themselves. ThirdEye includes three main components: *shopping behavior classification*, *user attention identification*, and *product layout inferencing*. We discuss them briefly in turn.

Shopping behavior classification: By analyzing over 50 video streams from 7 shoppers wearing the Google Glass, shadowing them as they were shopping and interviewing them, we have identified four distinct behaviors that shoppers exhibit at retail stores: *purposeful walking*, *dwelling*, *gazing*, and *reaching out*.

A shopper who knows what they wish to purchase or is in a hurry would likely walk purposefully to the relevant section. They might then dwell in a specific area (*e.g.*, an aisle) for a length of time. Such dwelling is characterized by slow, unpurposeful movement, say with the shopper ambling around nearby locations. Thereafter, having further narrowed down the products of interest, the shopper might gaze for a little time, looking steadily at a product or a set of closely-placed products, perhaps to search for or make up their mind about the products. Finally, the shopper might reach out for a product of interest with his or her hands, either to examine it more closely or to purchase it.

ThirdEye uses video and inertial sensing from smart glasses to determine whether the shopper is walking, dwelling, gazing, or reaching out. A key challenge is in tracking all of these activities accurately and in an energy-efficient manner. Since continuous video is extremely expensive in terms of energy consumption, our general strategy is to rely on inertial sensors as much as possible and to trigger vision only when necessary.

Purposeful walking: To detect purposeful walking, we simply leverage prior work [40] based on inertial sensing alone.

Dwelling: We develop a novel dwell detection scheme that uses inertial sensors to distinguish between walking and dwelling, with an accuracy of over 95% (Section 3.4). The scheme is based on the observation that while dwelling, the net physical displacement would be small even though the user may be moving continuously.

Gazing: While gaze is characterized by almost zero head movement, as described in Section 3.3, our experiments indicate that detecting gaze solely based on inertial sensing, even from head-mounted devices like smart glasses, leads to a large number of false positives (as high as 33%), with dwelling often being confused for gazing. Thus, we use inertial sensing based detection to turn on the video and then propose a scheme where vision based techniques can be used to complement inertial sensing, to achieve both energy efficiency and a gaze detection accuracy of 88% at a very low false positive rate of 1.2%.

Reaching out: Our scheme for detecting reaching out relies on the observation that hands are almost always seen within the field of view of the shopper (and hence of the Glass) whenever the shopper reaches out. The reason is simply that shoppers are typically looking at an item while reaching out for it. So we first detect hands, and then also examine the neighborhood of the detected hand to minimize false positives (Section 3.6), thereby achieving an accuracy of 86% with tests run on several shoppers.

User Attention Identification: Once we establish that the shopper is gazing, the video feed can be used to identify the products within the shopper's field of view using existing techniques such as Google reverse image search. However, as we discuss in Section 4.2, our experiments reveal the presence of as many as 16 products in the shopper's field of view. An important task then is to accurately identify the shopper's item of focus among the many products within view, *i.e.*, which item the shopper is looking at.

A naive approach would be to use the distance between an item and the center of the field of view as an (inverse) measure of the likelihood of it being the object of the shopper's focus. However, as we discuss in Section 4.2, the item of interest may not lie at the center of the field of view, depending on the orientation of the shopper's head. By analyzing ground truth data from several real shoppers, we develop a model for the likelihood of an item being at the shopper's focus, depending both on its position within the field of view and the orientation of the shopper's head, obtained from the Glass.

Layout Inferencing: Thus far, we have focused on identifying items within the shopper's field of view. However, limiting ourselves to such items would not be sufficient from the viewpoint of our goal of tracking physical browsing. For instance, an application might wish to generate a proximity-based alert for an item that is quite close yet out of view (*e.g.*, it might be right behind the shopper). Also, there would be no vision-based sensing for shoppers who only have a smartphone and no Glass. Thus, inferring the layout of the products becomes an important goal for ThirdEye.

Since we do not rely on the store to provide us with a map, we design a novel scheme called *AutoLayout* that fuses video, WiFi, and inertial sensor data from the Glass devices worn by various shoppers, to simultaneously track them while also constructing and updating the *product layout* in a virtual coordinate space (Section 5). The advantage of this scheme is that we can localize not only Glass-enabled shoppers but also shoppers who carry only smartphones using WiFi and inertial sensor data, albeit with reduced accuracy.

Contributions: Our key contributions are:

- A first-of-its-kind system, ThirdEye, to track physical browsing in retail stores using data from smart glasses, and without relying on input from either the store or the shoppers.
- Techniques to identify shopper behavior such as dwelling and gazing using video and inertial sensors.
- A scheme to determine the shopper's focus while gazing.
- A scheme to simultaneously track shoppers and infer the product layout using video, WiFi and inertial sensing data from smart glass users, and using this to also track physical browsing of the smartphone-only users albeit at a reduced accuracy.
- An implementation of ThirdEye and its evaluation in two retail stores, with the participation of 7 shoppers.

2. RELATED WORK

Prior work of relevance to ThirdEye spans a wide range: localization, vision-based sensing, robotics, human activity sensing, and physical analytics in retail (including the work in start-ups).

Indoor Localization and Sensing: There has been extensive work in indoor localization and sensing [49, 50, 39, 24, 27, 5] that cover spaces both big (*e.g.*, buildings) [48, 42] and small (*e.g.*, retail

stores) [14, 12, 45], both using infrastructure (*e.g.*, WiFi, RFID) [10, 29, 37, 51] and not (*e.g.*, purely-phone based, crowd-sourcing) [9, 34, 16], sensing both the environment [41, 53] and the users [44, 13]. Despite much work on WiFi localization, existing work can achieve high accuracy only at the expense of a high calibration cost or requiring additional information or modifications to WiFi Access Points (APs) (*e.g.*, Horus [53] requires detailed profiling of the indoor space ahead of time, Zee [40] requires maps and ArrayTrack [51] requires modified APs). In comparison, our system achieves semantically meaningful information, such as item or category labels without requiring external input (*e.g.*, a map) or other human input (*e.g.*, manually-assigned labels from crowd-sourcing).

SurroundSense [9] proposes an interesting approach that fingerprints locations based on visual, acoustic, and motion signals. The goal is to determine which store the user is located in. In comparison, ThirdEye seeks to identify the user's context in a more fine-grained manner than just the identity of the store that they are in (*e.g.*, whether the user is gazing and if so at which item) and do so without requiring any fingerprinting in advance.

CrowdInside [8] presents a method for constructing indoor floor plans by leveraging the movement of users carrying smartphones. It uses dead-reckoning together with anchor points to prevent error accumulation. The anchor points are defined by the unique inertial sensor signatures corresponding to elevators, escalators, and stairs. However, such inertial-based anchor points might be few in number or even non-existent within a store. In contrast, by leveraging vision, ThirdEye is able to identify more accurate, item-level landmarks. Furthermore, by combining this with both inertial tracking and WiFi information, ThirdEye provides a unified framework that caters to both smart glass users and smartphone-only users.

Vision: There is a vast body of research on vision-based place recognition and pose estimation. For example, [7] construct a 3D model of a city based on a large number of photos collected online, which can then be used to estimate the pose (*i.e.*, position and orientation) of an observer with a given view. [33] determines the location where a photograph was taken by matching it with photographs of popular landmarks. [38] develops an approach that retrieves similar images from a visual database on a mobile device with a small memory footprint.

Vision-based approaches are generally expensive, especially if it involves building a 3D model, hence these have generally only been applied to well-known landmarks. The insides of stores are typically lacking in such distinctive landmarks and are often crowded with people, posing challenges. In ThirdEye, we make limited use of image information to recognize items rather than scenes, thereby sidestepping these challenges.

Robotics: There is a vast body of work on robot navigation, specifically on the classical problem of simultaneous localization and mapping (SLAM) [32]. While the robots typically have access to precise wheel odometry, ThirdEye uses only human walking and does not use specialized sensors, such as laser range-finders.

There have also been prior SLAM-like efforts in the context of human motion. FootSLAM [42] uses foot-mounted inertial sensors to track a user's walk, while FeetSLAM [43] extends this to combine the trajectories of multiple users, assuming each walk is sufficiently long. Neither uses WiFi or visual information. WiFiSLAM [19] only uses WiFi information, not inertial sensing. Only visual information is used in MonoSLAM [18]. Zee [40] combines smartphone-based inertial and WiFi sensing but requires an a priori map. To our knowledge, ThirdEye is the first system that combines inertial and WiFi sensing with visual information to construct a map from walks by multiple humans, even when the individual walks are short.

Human-activity sensing: There has been much work on fine-grained human-activity sensing using wearable devices such as pedometers, heart-rate monitors, microphones, etc. [15, 20, 17]. ThirdEye focuses on a small subset of activities of relevance in the physical browsing context and enables these using smart glasses. As more wearable devices become prevalent, we could leverage those too; regardless, ThirdEye's ability to combine inertial, WiFi, and visual information would help provide physical context to the sensed information (*e.g.*, associating a jump in the user's heart rate with them gazing at particular object on display).

Shopping behavior: [28] places sensors on people to track their trajectories, and uses a clustering scheme to predict users' future behaviors (*e.g.*, fast walking, idle walking, or stopping). [31] studies 701 hours of sensor data collected from 195 in-situ customers to understand customer behavior in shopping malls. [52] monitors customers' shopping time. In comparison to these specific studies, ThirdEye provides a broad and systematic framework for combining multiple sensing modalities to track physical browsing by users.

Retail analytics start-ups: Nearby Systems [36] requires retail stores to deploy a customized WiFi localization infrastructure to analyze in-store customers (dwell times, visit frequency, window conversions, etc.) Euclid Analytics [21] leverages existing in-store WiFi infrastructure to provide similar analytics to retail stores. Neither approach provides fine-grained, item-level information. Apple's iBeacon [26] transmits location-specific messages in a store to nearby smartphones via Bluetooth Low Energy (BLE). Mondelez [46] requires retail stores to deploy cameras in shelves that use facial-recognition to provide insights into demographics of customers that browse a given product. In comparison to these systems, which require extensive infrastructure roll-out or support and hence are meant to be deployed by store owners, ThirdEye does not require any special infrastructure and hence is amenable to crowdsourcing. Moreover, ThirdEye provides more fine-grained analysis including user behaviors.

3. BEHAVIOR CLASSIFICATION

As discussed in Section 1, there are four modes exhibited by a typical shopper: purposeful walking, dwelling, gazing, and reaching out. We now describe these in greater detail and present the detection techniques used by ThirdEye.

3.1 Understanding the behaviors

Upon entering a store, a shopper might walk purposefully towards a certain section of the store that they have in mind or in a certain direction. Upon reaching the section or aisle of interest, the shopper would typically slow down and dwell, i.e., amble around, say looking within an aisle for their product of interest. Occasionally, the shopper might stand at a spot and gaze at products on the shelf, visually scanning these to find the one of interest and perhaps making up their mind to reach out to it. Finally, the shoppers might reach out to the item of interest, either to pick it up or just to read the information on it.

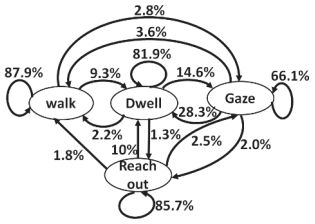


Figure 1: An example model for shopping behavior

The amount of time spent by the shoppers in each of the four modes would reflect their shopping intent. For example, shoppers who have time to kill or are unsure of what they intend to buy might spend a considerable length of time dwelling. Shoppers trying to

compare and select among different brands/options of a specific item might be gazing. Finally, the act of reaching out indicates the greatest intent where the shoppers either intend to purchase the item or read information written on the item such as calorie contents or net weight. These behaviors may be exhibited in various sections of the store and may even be repeated by a shopper as (s)he comes back for a second look.

	To					P(State)
	State	Walk	Dwell	Gaze	Reach Out	
From	Walk	87.9%	9.3%	2.8%	0%	17.3%
	Dwell	2.2%	81.9%	14.6%	1.3%	50.7%
	Gaze	3.6%	28.3%	66.1%	2.0%	23.8%
	Reach Out	1.8%	10%	2.5%	85.7%	8.2%

Table 1: State transition probabilities

Figure 1 models the various behaviors as a finite state machine quantifying the probabilities of transitioning from one state to another at a granularity of one second. Table 1 reports the state transition probabilities estimated by studying the behavior of 7 shoppers who shopped in two large stores in the United States, a grocery store and a department store, while wearing Google Glass. We shadowed the shoppers, taking notes as they shopped, interviewed them in the end, and finally studied the video obtained from their Google Glass. Thus, we manually classified the time spent by the shoppers in the store as walking, dwelling, gazing, or reaching out. We make the following observations:

- Shoppers tend to spend a majority of their time dwelling (50.7%) (as indicated by the P(State) column in Table 1), followed by gazing (23.7%) and walking (17.3%).
- Most frequent inter-state transitions are between dwell and gaze.
- For all four states, the high probability of remaining in the same state indicates that shoppers tend to continue in their current mode for several seconds before transitioning to another.

3.2 Behavior Classification Overview

ThirdEye automatically classifies shopper behavior into one of the four categories: walking, dwelling, gazing, and reaching out — using the inertial sensor data (*i.e.*, accelerometer and 3-axis compass) and the video feed from the camera of the Google Glass. ThirdEye also makes use of the inertial sensor data on shoppers' smartphones, where available, as discussed later in this section.

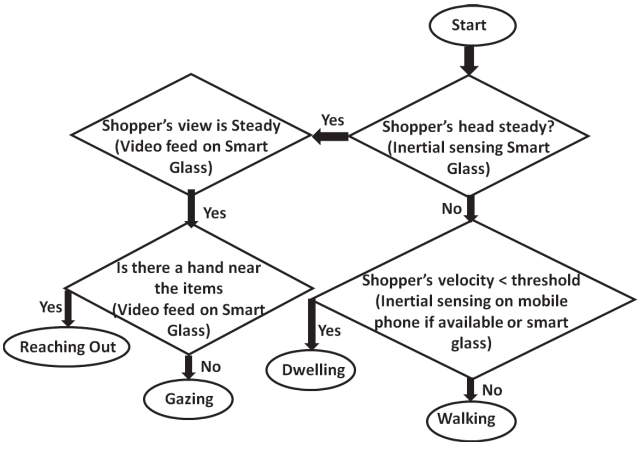


Figure 2: Overview of Behavior Classification Algorithm

Turning on the video on Google Glass causes a high power drain on the device (as discussed in detail in Section 6), while inertial sensing results in much less power drain. Consequently, in our behavior classification scheme, we try to minimize the duration for which the video is turned on. Figure 2 depicts the high-level flow of our behavior classification scheme. We ran our scheme once every second, thus shopper behavior was classified at the granularity of each second.

Gaze detection: The first step in our scheme is to detect whether or not the person is gazing. Since shoppers typically hold their head steady while gazing and are also not walking, this can be detected by measuring the standard deviation in inertial sensor measurements on the smart glass. However, as will be discussed in Section 3.3, just relying on inertial sensors leads to a large number of false positives. Consequently, we use a preliminary detection based on a low standard deviation in the inertial sensor measurements on the smart glass as a trigger to turn on the video and analyze the scene to confirm that the person is indeed gazing (Section 3.3).

Dwelling and Walking: If there is a high standard deviation in the inertial sensors, this could mean that the person is either dwelling or walking. We use the step counter in Zee [40] to detect steps from the accelerometer. As discussed in Section 3.4, the accuracy of the step counter is higher when the accelerometer readings from the shopper's smartphone are used compared to when measurements from the smart glasses are used. This is because of spurious step detections caused by head movement. The step count is combined with compass data from the smart glasses to estimate the average velocity of the shopper. If the magnitude of average velocity is less than a certain threshold, the behavior is classified as dwelling, otherwise it is classified as walking.

Reaching out: In order to detect reaching out, we rely on detecting the shopper's hand in the field of view. As described in Section 3.6, we train an existing classifier [2] to detect hands in the video feed.

3.3 Gaze Detection

Gaze is an important signal since, when a shopper gazes at one or more products, it reflects his or her interest in those products. We expect the person's head to be relatively steady while they are gazing. This can be detected using the inertial sensors on the smart glasses, in particular the accelerometer and the 3-axis compass. We use the rule $std(a_i) < th_a$ and $std(\theta_i) < th_\theta$ for $i \in \{x, y, z\}$, *i.e.*, the standard deviation in the acceleration values along all three axes must be lower than the threshold th_a, and the standard deviation along all three axes of compass must be lower than the threshold th_θ. Figure 3 depicts the dependence of false negatives (*i.e.*, missing gaze detection) and false positives (*i.e.*, mistaking other behavior as gazing) as a function of various combinations of threshold values. As seen from Figure 3, reducing th_a and th_{theta} decreases the false positives but increases the false negatives. While false negatives lead to missing actual gazing events, false positives will lead to turning on the video too frequently leading to higher power consumption. In our implementation, we chose thresholds at th_{theta} = 4.4 degrees and th_a = 0.7g m/s^2, to limit false negatives to 10% while minimizing false positives (at 33%).

Using optical flow to confirm gaze: However, 33% false positives is still too high. Since video has already been turned on now, one way to reduce this is to analyze the video feed to determine whether the person is indeed gazing. In order to achieve this, we rely on the property that when the shopper is gazing (s)he will intentionally make sure that there is very little change in what (s)he is seeing, from one moment to the next. Therefore, we used a vision based technique — *optical flow* [25] — applied to the video captured by the Glass. Optical flow measures the movement of pixels between consecutive images. We perform the optical flow computation on frames captured at 10 frames per second and calculate the average

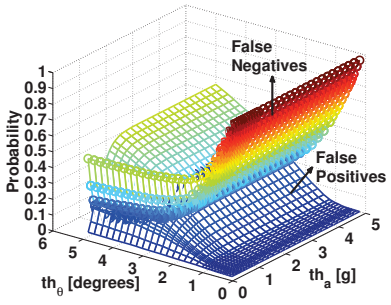


Figure 3: Choosing thresholds for gaze detection using inertial sensors

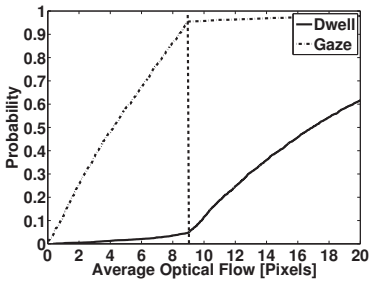


Figure 4: Choosing threshold for Optical flow based gaze detection

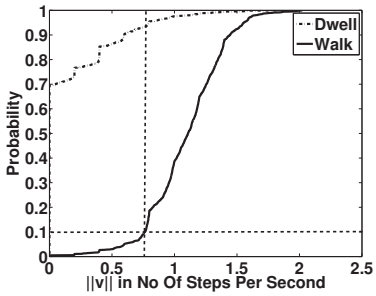


Figure 5: Choosing the threshold for Dwell Detection

magnitude of the optical flow vectors over a one-second window. We then apply a threshold to this average optical flow value to decide whether the shopper is, in fact, gazing. Figure 4 depicts the cumulative distribution function of the optical flow for dwell and gaze behaviors obtained from the labeled videos (from experiments where the camera was turned on all the time, to understand the difference between the optical flow values during dwell and gaze). As seen from Figure 4, a threshold of 9 pixels, achieves a high detection rate (97%) while keeping the false detections low (2%) if video is on all the time. We achieve a detection rate of 88% with 1.2% false detections for our energy efficient scheme where the video is turned on only after gaze is detected by the inertial sensors.

Finally, we shed some light on why gaze detection based just on the inertial sensors leads to a high false positive rate of 33%. We rule out the movement of a shopper's eyes while their head is held steady as a significant cause since the optical flow method, which also does no eye tracking, yields a low false positive rate of 2%. Based on a closer examination, we identify three reasons for the significant difference in the detection rate in the two cases: (1) the shopper stands at a spot and leans forward, *i.e.*, zooms in, while holding their head steady to examine a product closely, (2) the shopper is looking at an object that they are holding and turning around from side to side (*e.g.*, to examine the front and back of a product), and (3) the shopper is looking steadily at a distant scene, which is a more common reason. In each case, inertial sensors based detection concludes that the shopper is gazing while the optical flow based approach concludes the opposite. One could argue that at least cases (1) and (2) correspond to the shopper, in fact, showing attention in a manner that the notion of gazing is intended to capture. However, we have taken the conservative position of restricting gazing to only cases where the visual image seen by the user is steady.

3.4 Dwell Detection

Dwelling refers to the shopper lingering around the same location. While the shopper could be taking steps during dwelling, they would tend to be ambling around, but roughly remaining in the vicinity, *i.e.*, having a small net displacement. Consequently, there could be significant movement during dwelling, so dwell detection based on simple approaches such as measuring the accelerometer magnitude or even looking for a reduction in the step rate, may not work correctly. Therefore, to build an effective detector, we rely on the intuition that unlike purposeful walking, in dwelling, the net displacement of the shopper will be very small, even though he/she may be walking around. In other words, dwelling is detected by a sharp drop in the magnitude of the average velocity vector of the shopper.

Device	Step Count Error [%]
Phone	2.6%
Glass	10.2%

Table 2: Step detection accuracy in mobile phone vs smart glasses

As discussed in Section 1, we use the step counter in Zee [40] to detect walking. Since most shoppers are likely to be carrying a smartphone with them, even if they are wearing smart glasses, measurements from either the smartphone or the smart glasses (or both) can be used to detect and count steps reliably. In our experiments, we found detection to be superior when done in the mobile phone as compared to smart glasses, as indicated in Table 2. This is because a shopper's head movements during shopping are mistaken for steps. Consequently, in ThirdEye we use inertial sensors from the phone for step counting, whenever possible.

In order to detect dwelling, we continuously compute the net displacement over a window of the past τ seconds by using the compass in the Google Glass to provide the instantaneous direction of the shopper's motion. Suppose that the shopper takes K steps during a τ-second window. Further, suppose that at the i^{th} step, his/her orientation was θ_i. Then, we compute the magnitude of the net velocity vector as,

$$\|v\| = \sqrt{\left(\sum_{i=1}^{i=K} \cos\theta_i\right)^2 + \left(\sum_{i=1}^{i=K} \sin\theta_i\right)^2} \quad (1)$$

When $\|v\|$ drops below a certain threshold $\|v\|_{dwell}$ steps/sec, we conclude that the user is dwelling. In our implementation we set $\tau = 5$ seconds, under the assumption that any period shorter than 5 seconds cannot be considered as dwelling.

In order to choose the threshold $\|v\|_{dwell}$, we considered the distributions of velocities seen during walking and during dwelling, namely $\phi_{walk}(\|v\|)$ and $\phi_{dwell}(\|v\|)$ (depicted in Figure 5), as derived from the ground truth obtained from the videos collected from the shoppers. As seen from Figure 5, the threshold of 0.71 was chosen to allow 10% false errors (*i.e.*, walking is mistaken as dwelling) while providing a detection rate of about 95% (*i.e.*, dwelling is mistaken as walking 5% of the time).

3.5 Summary of Dwell, Walk, Gaze Detection

Overall, using inertial sensing based dwell detection and inertial sensing plus optical flow based gaze detection, ThirdEye is able to effectively classify the shopper's traversal into walking, dwelling, and gazing. Table 3 shows the confusion matrix for this classification based on 3 hours of store visit data from 7 shoppers. The large values along the diagonal (in bold) show that the classification

Predicted⇒ Actual⇓	Walk	Dwell	Gaze
Walk	**90.4%**	9.1%	0.5%
Dwell	3.4%	**95.8%**	0.8%
Gaze	1.4%	10.7%	**87.9%**

Table 3: The confusion matrix for classification of walk vs. dwell vs. gaze

is correct in the overwhelming majority of cases. It is also worth remembering that there is an element of subjectivity in the ground truth, and that might account for some of the misclassifications.

3.6 Reaching Out Detection

When the shopper eventually reaches out for a product, it indicates a very high degree of interest in that product. How can we detect such reaching out events? Unsurprisingly, we find that measurements from smart glasses or smartphone based inertial sensors are inadequate for distinguishing reaching out from gazing, since these sensors do not detect movement of the shopper's arm or hand.

Our detection scheme relies on the observation that in most cases of reaching out, the user's hand becomes visible in the image, whereas it tends not to be seen in the image when the shopper is walking, dwelling, or gazing. This is because users tend to look at whatever they are reaching out for and moreover the camera in the smart glasses typically sees what the user sees.

To detect hands, we trained a TextonBoost classifier [2] on 100 hand images from various shoppers. The classifier identifies the areas covered by a hand and marks them as depicted in Figure 6 and Figure 7. These figures also show two typical causes of error that arise from using the classifier, namely *hand fragmentation* and *spurious detections*. As seen in Figure 6, the presence of a watch on the hand fragments the hand into two separate parts. On the other hand, in Figure 7 the presence of items with similar texture and color as the hand results in spurious detections. To deal with fragmentation, we cluster together all fragments that either have overlapping bounding boxes or bounding boxes that are within 50 pixels of each other at their closest points. To deal with spurious detections, we rely on the observation that most spurious fragments tend to be much smaller in area compared to a real hand. Consequently we eliminated all fragments that occupied fewer than 1500 pixels (0.16% of the image). Overall, our reaching out detection scheme has a success rate of 86%, and a false detection rate of about 15%, based on testing over several videos from the training set. Further, we may miss detections when shoppers are reaching out for items in the lower shelves, as their hands may not appear within the field of view. Our reaching-out detection is currently offline since it involves more expensive vision processing and is an analytics step to understand the user's interests. Perhaps tapping sensor data from other devices such as smart watches would improve the reliability and computation cost of reaching out detection in such cases. Also, the use of more advanced pattern recognition techniques such as deep learning [30] could increase the success rate while reducing the false detections. We defer an exploration of these directions to future work.

4. ATTENTION IDENTIFICATION

Once it is established that the shopper is either gazing or reaching out, the next step is to identify the item(s) that the shopper is bestowing their attention on. Here, visual input plays a key role.

4.1 Reverse Image Search

ThirdEye captures images and invokes a cloud service to perform reverse image search, *i.e.*, identify the objects contained within the image. In our current implementation, we use Google's service [3], although ThirdEye is not tied to it and could also invoke alternative services, such as Bing Vision [1] and Nokia Point & Find [4].

Reverse image search involves object recognition, which in turn involves extracting image features (*e.g.*, using an algorithm such as SIFT [35]), comparing these with precomputed features from a large corpus of existing images to find a match, and retrieving meta-data associated with the matching image(s) in the corpus (*e.g.*, product name). The scale and complexity of this computation necessitates running it on the cloud (*e.g.*, the corpus of product images could run into the tens of millions). In turn, this calls for being selective in how frequently reverse image search requests are sent.

Therefore, only when gazing or reaching out are detected, does ThirdEye capture images and invoke the cloud service. During each period of gazing, ThirdEye captures all the images during the gazing period. Each image is then divided into 25 overlapping parts (in order to avoid images being broken at the borders) and each part is sent up to the cloud separately to minimize the chances of object recognition failing because of image-complexity issues. An example is depicted in Figure 8 with the overlapping parts that are sent for look up. In the example in Figure 8, 2 out of 5 brands were detected by Google reverse image search. Our evaluation based on manually studying the frames indicates that this scheme detects 53% of the products in the frame correctly. 7% are false positives, in other words, products are recognized incorrectly when they are absent. Finally, almost 40% of the products are never recognized. Further analysis indicated that almost all these 40% were not popular brand names but local brand names or items without labels like fruits and vegetables. Note that human computation (*e.g.*, through Amazon Mechanical Turk) could be used for brands that are not listed as a part of Google reverse image search and thus improve the database.

Figure 9 depicts the distribution of reverse image look up times over all our look ups. As seen from Figure 9, the look up times vary between 250 ms and 1 sec. with a mean of 650 ms.

4.2 Focus Within Field of View

In general, the number of items within the field of view of the Glass could be large. Figure 10 depicts the cumulative distribution function of the number of items recognized in any given frame. As seen from the figure, in some cases the number of items seen can be as large as 16. It is likely, however, that the shopper is only focused on one or perhaps a small number out of these items in his/her field of view. Our goal, therefore, is to identify the shopper's focus within the field of view.

It might be tempting to assume that the shopper's focus lies at the center of the field of view of their Glass. Figure 11 depicts the distribution of center of attention from the videos from several shoppers as they gazed. To learn the center of attention, we shadowed each shopper (*i.e.*, walked with them) as they shopped and also interviewed them afterwards while playing back the video recording from their Glass. As seen from the figure, the center of attention of the shopper is not necessarily at the center of the screen but is distributed over various pixels in the view. The dotted rectangle ABCD in Figure 11 depicts the area that includes 90% of the centers of attention as indicated by the shoppers. Based on Figure 11, the region of the video that might include the center of attention with 90% probability is only about 27% ($\frac{575\times426}{1280\times720}$) of the frame.

Further, as seen from the figure, there is a clear bias towards the left of the field of view. In fact, the mean x value in Figure 11 is $x_o = 511$. This is simply a consequence of the fact that the front-facing camera in Glass is mounted to one side (to the top-right of the right eye), so the center of the shopper's field of view lies to the left of that of the Glass.

Dependence of Attention on Head-Tilt: At a first glance, Figure 11 seems to indicate that the variation along y-axis of the center of attention is random and has a spread across the entire view (580 out of 720 pixels). However, deeper analysis revealed an interesting fact – *the center of attention in fact depends on the tilt of the head of the person.* Figure 12 depicts the dependence of the y-coordinate of the center of attention as a function of tilt of the head δ, as mea-

Figure 6: Two pieces of same hand

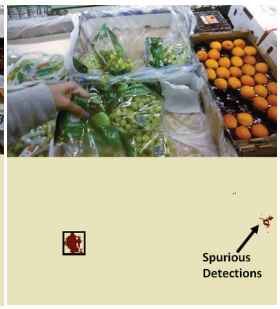


Figure 7: Spurious hand

Figure 8: Reverse image search example. Brands found: Lands o lakes butter, Earth Balance. Brands missed: Spreadable butter, Olivio, Smart Balance

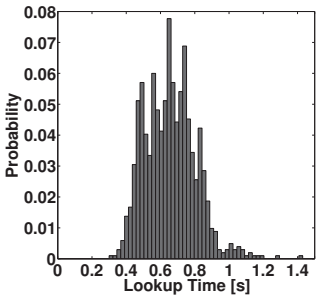


Figure 9: Distribution of reverse search times

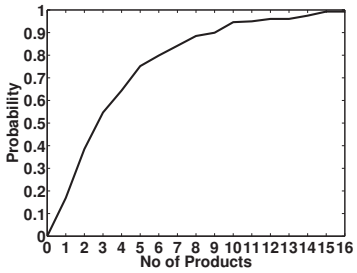


Figure 10: CDF of number of items seen in the field of view

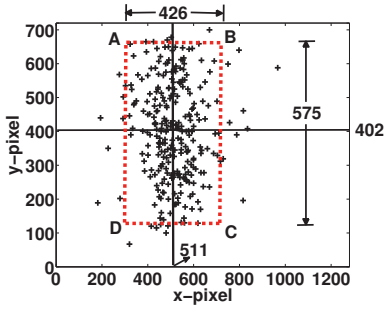


Figure 11: Distribution of center of attention

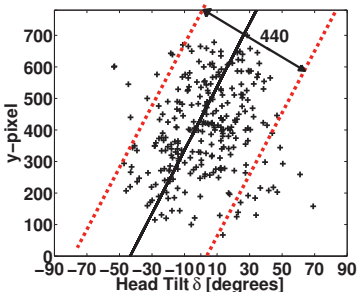


Figure 12: Dependence of center of attention on tilt

sured by the compass on the Google Glass. As seen from Figure 12, there is a mild linear dependence of the center of attention on the head tilt. We used RANSAC [22] (a popular scheme used in computer vision that is robust to outliers) to find the equation of the line to be $y = m\delta + c$, where $m = 10.05$ and $c = 436.38$ (depicted in Figure 12).

In hindsight, this makes sense, since when a shopper looks up or down towards a product, he/she accomplishes this by partly turning head up/down and partly by rolling their eye-balls. Consequently, the shopper's focus within their field of view would tend to shift in the same direction as their head is tilting. The region that includes the 90% of the centers of attention is also indicated by the dotted lines in Figure 12. As seen from the figure, the spread around the line now has reduced to 440 (60% of 720) instead of 575 (80% of 720) pixels (in Figure 11).

Probability Distributions for center of attention: Since it is in principle impossible to determine exactly which item the shopper is focused on, we resort to assigning a probability value $P(x, y|\delta)$ to an item in his field of view at the x and y pixel position, given the head-tilt, δ, as obtained from the compass:

$$P(x, y|\delta) = \Phi_X(x - x_o)\Phi_Y(y - m\delta - c) \quad (2)$$

The distributions $\Phi_X(z)$ and $\Phi_Y(z)$ are obtained using histograms from actual data. Figure 13 depicts the function $P(x, y|\delta)$ as a function of $x - x_o$, and $y - m\delta - c$.

Evaluation of Focus of Attention: We now ask the question how accurately we can identify the item that the person is looking at during gaze? We consider two schemes to measure the relative importance of various items in view. Our first scheme, deemed Naive in Table 4, naively uses distance from the center of the video (*i.e.*, pixel 640, 360) while our second scheme, deemed Max-Likelihood in Table 4, uses $P(x, y|\delta)$. In Table 4 we measure the fraction of times the top ranked item corresponded to the correct item the shopper was gazing at and also the fraction of times where the correct item was among the top three ranked item.

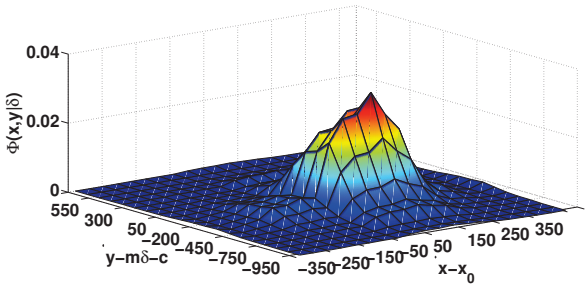


Figure 13: $\Phi(x, y|\delta)$

As seen from Table 4 the Max-likelihood approach is 76% accurate in terms of pinpointing the correct item being gazed at, while the Naive approach only succeeds 66% of the time. Further, about 90% of the time, the correct item is among the top three prospective candidate items.

Scheme	Top 1	Top 2	Top 3
Naive	66%	80%	86%
Max-Likelihood	76%	86%	90%

Table 4: Determination of focus of attention

Devices such as the Tobii glasses [6] perform eye tracking, which can enable more accurate user attention identification than we present in this paper, but these are sophisticated devices which include multiple eye cameras in addition to a front-facing scene camera.

5. AUTOLAYOUT DESIGN

As discussed in Section 1, another important requirement to enable physical browsing analytics is information regarding the layout of products within a store. However, given the immense num-

ber of stores in the world, including the many that a single shopper could visit over time, and the likely reluctance of store owners to share information with their competitors, a crowd-sourcing based solution that can organically build layout maps without any help from the stores is attractive. Further, since we do not wish to discomfort the shoppers during their shopping, we also do not wish to seek any inputs from the shoppers. Therefore, we propose *AutoLayout*, which automatically infers layout based on video feed, inertial sensing data, and WiFi measurements obtained from the shoppers equipped with smart glasses. If individual store owners make partial or complete layout information available to us (*e.g.*, distance between certain products), we can incorporate it into our AutoLayout framework to further enhance accuracy.

Inferring layout from crowd-sourced data is challenging:

- We have no control over shopper behavior, *i.e.*, no control over how they walk or where they dwell.
- There is no ground truth available from the shoppers since we do not seek any input from them.
- While there is much existing work on building 3-D layouts from image or video data, these assume availability of well-known landmarks, which are usually not available in stores. Moreover, they also require extensive imagery with overlapping coverage, which is problematic in our context because we have no control over where the shoppers go or which way they look and recording the video continuously incurs prohibitive energy cost.

The key idea in AutoLayout is that while different shoppers walk in different ways within the store, WiFi APs and positions of products in the store can serve as anchors to align these different walks. The anchors can be identified using both WiFi RSS and images from the videos. AutoLayout simultaneously estimates the product locations and tracks the shoppers in a virtual 2-D coordinate system. Further, AutoLayout constructs a WiFi propagation model (based on EZ [16]) so that it can track both smart glass users and smartphone users, albeit with reduced accuracy for smartphone users.

5.1 Data Collected by AutoLayout

AutoLayout continuously collects inertial sensor data, WiFi data and opportunistically collects video feeds (whenever the shopper is gazing at products).

Inertial Data: As shoppers enter a store, in the background AutoLayout constantly keeps collecting inertial sensing data (accelerometer, compass), and continuously counts steps and tracks the direction θ, as obtained from the Google Glass, as the shopper walks around the store. Thus, for the i^{th} step that the k^{th} shopper takes, a direction value θ_k^i is recorded.

Video Data: Suppose that after the m^{th} step, the k^{th} shopper stops to gaze at or reach out for a product within the store, then as discussed in Section 4, the camera on the Google Glass is turned on and various products within view are identified by processing the images. Let I denote the set of all products in the store. Each product I_j is also associated with a list L_j of $< m, k >$ pairs indicating that the k^{th} shopper was in the immediate vicinity of the product at this m^{th} step. Note that the same shopper may approach the same product repeatedly.

WiFi Data: The glasses perform periodic WiFi scans to record the WiFi signal strengths. Suppose that, the l^{th} WiFi access point AP_l was seen by the k^{th} shopper at the n^{th} step and the measured signal strength was $r^{l,k,n}$. For each WiFi access point, a list W_l is maintained that stores the values, $r^{l,k,n}$.

5.2 Joint Layout Construction and Tracking

The unknowns that AutoLayout needs to estimate are:

- x_k^i, 2-D location vector of the k^{th} shopper after his i^{th} step
- z^j, 2-D location vector of j^{th} product within the store, and,
- $< P_l, y^l, \gamma_l >$, P_l is the transmit power of the WiFi AP_l, y^l is its location, and γ_l is the path loss constant for the Log Distance Path Loss (LDPL) propagation model [16]. This is because, AutoLayout uses the LDPL model to predict the received signal strength at various locations from the APs.

The approach taken by AutoLayout is to minimize the objective function, which quantifies various errors, as shown below:

$$\begin{aligned} J &= J_X + J_Y + J_{AP} \\ J_X &= \frac{1}{\sigma_X^2} \sum_i \sum_k \|x_k^{i+1} - x_k^i - \hat{e}_k^i\|^2 \\ \hat{e}_k^i &= \begin{bmatrix} \cos\theta_k^i & \sin\theta_k^i \end{bmatrix}^T \\ J_Y &= \frac{1}{\sigma_Y^2} \sum_{j, I_j \in I} \sum_{<m,k> \in L_j} \|x_k^m - z^j\|^2 \\ J_{AP} &= \frac{1}{\sigma_{AP}^2} \sum_l \sum_{r^{l,k,n} \in W_l} \left\| r^{l,k,n} - rss(P_l, \gamma_l, y^l, x_k^n) \right\| \\ rss(P, \gamma, y, x) &= P - 10\gamma \log(\|y - x\|) \end{aligned} \tag{3}$$

In Eqn 3, the overall error objective function J comprises of three parts. First, J_X captures the constraint that shopper locations x_k^{i+1} and x_k^i coming from two consecutive steps of the same shopper must be separated by a single step and may be subject to σ_X (0.1 in our implementation) standard deviation of error. Second, J_Y captures the fact that all shopper locations from where a particular product was gazed at by the shopper must be very close to each other (we use a variance σ_Y, 1 step in our implementation, to account for the fact that the shoppers may not be exactly at the same place). This term essentially ties all the tracks from all the shopper together into the same coordinate system. Third, J_{AP} minimizes the difference between measured RSS and estimated RSS based on the parameters describing the LDPL model for all APs across all observations made by all shoppers. Since no global reference frame can be established, AutoLayout arbitrarily fixes any one position of a shopper as the origin. Our implementation simply chooses $x_0^0 = \mathbf{0}$.

5.3 The Optimization

Optimizing Eqn 3 is no easy task since the search involves a large number of unknowns and the objective is non-linear with multiple local minima. Consequently, we take a four-step approach to solve the optimization problem.

1. Obtain an initial estimate for shopper locations x_k^i and product locations z^j using the *BFSLoc* algorithm described shortly.
2. Optimize the partial objective function $J_{part} = J_X + J_Y$ to refine x_k^i and z^j using gradient descent method.
3. Obtain an initial guess for all the AP parameters $< P_l, y^l, \gamma_l >$ by minimizing J_{AP} using gradient descent method based on the existing values for x_k^i and z^j.
4. Starting with current values of the unknowns as the initial values, we iteratively optimize the overall objective function J as follows: first search x_k^i and z^j's that optimize J while fixing APs' parameters, and then search APs' parameters that optimize J while fixing x_k^i's and z^j's, and iterate. We iterate until the objective function improvement is within 10^{-5}.

In the interest of brevity, we do not describe gradient descent, since it is a well known optimization method. We describe BFSLoc, the first step in the AutoLayout procedure outlined above.

BFSLoc: The key idea in BFSLoc is to estimate locations (either product location or shopper location) using the shortest possible explanation. BFSLoc starts by constructing a graph where each shopper's location or product's location is associated with a node

as depicted in Figure 14. Edges exists between nodes that represent consecutive steps from the same shopper (*i.e.*, x_k^i and x_k^{i+1}). Edges also exist between nodes that represent a product location z^j and a shopper location x_k^m when the product has been seen at that shopper location ($< m, k > \in L_j$).

The BFSLoc algorithm involves performing a breath first traversal on this graph starting from x_0^0 until all the nodes are traversed. Upon visiting any node a from a parent/previous node b, the value of the unknown corresponding to the node is computed:

- if $a = x_k^{i+1}$ and $b = x_k^i$, compute $x_k^{i+1} = x_k^i + \hat{e}_k^i$.
- if $a = x_k^{i-1}$ and $b = x_k^i$, compute $x_k^{i-1} = x_k^i - \hat{e}_k^i$.
- if $a = x_k^i$ and $b = z^j$, compute $x_k^i = R(z^j, \rho)$, where $R(z, \rho)$ generates a random location in a disc of radius ρ with its center as the 2-D location z.
- if $a = z^j$ and $b = x_k^i$, compute $z^j = R(x_k^i, \rho)$

The intuition here is that, starting from x_0^0 (which by definition is a certain location in our virtual coordinate space), we wish to estimate the location of all entities, whether a shopper at a particular step in their walk or a particular product, by following the fewest "links in the chain", with a view to minimize the accumulated error. We emphasize that BFSLoc is only the first step in AutoLayout procedure, so our goal in this first step is to obtain a good initial guess, not the correct estimates.

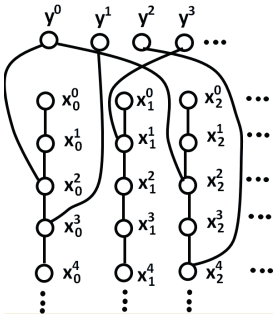
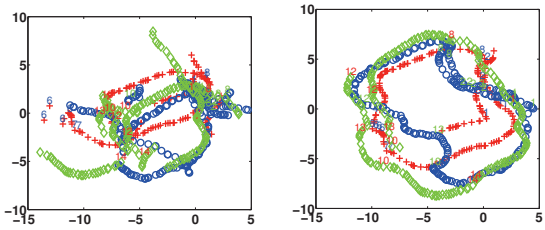


Figure 14: BFSLoc graph



(a) After BFSLoc (b) After Optimization

Figure 15: Function example of AutoLayout

5.4 An example of AutoLayout Functioning

In order to provide an intuition to the reader about the functioning of AutoLayout, we now show an example. In this example, three different shoppers were asked to take a rectangular route through the aisles in a retail store. The entire route was about 80m long. During the measurement, they occasionally walked, gazed and dwelled. Using the video, inertial and WiFi data from the three shoppers, we used AutoLayout to construct the product locations and the tracks of the shoppers. Figure 15 shows the layouts obtained after applying BFSLoc and after the entire optimization process. As seen from Figure 15 at the end of BFSLoc, the tracks of the three shoppers are very fragmented and do not overlap very well. This is due to the following reasons. First, there are errors in the compass measurements and in counting steps. Second, shopper stride lengths are different. However, after the optimization, AutoLayout not only determines the relative product locations but also identifies the entire path taken by the various shoppers. As a result in Figure 15 (b) the tracks after optimization look closer to the rectangular paths that the shoppers took. Moreover we also see AutoLayout inferred two large overlapping rectangles (where the two shoppers walked around the same set of aisles) and one smaller rectangle where the user turned around in the previous aisle.

5.5 Tracking Users in real Time

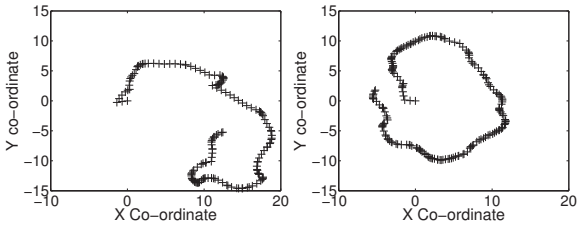
For tracking shoppers in real-time, we use the same scheme as above except that we only consider the shopper's locations as unknown. Thus, the moment the shopper sees a product while gazing, a WiFi scan is performed and his/her location is initialized. Thereafter, the subsequent locations are estimated by performing a gradient decent on J over only the shopper's locations. The optimization converges very rapidly in real-time since the initial location estimate is already quite close to the shopper's real location.

5.6 Dealing with shopper's head movements

An important measurement in AutoLayout is the shopper's direction of motion. Since smart glasses are located on the top of the head and face the direction of the user, they typically are aligned in the direction of motion of the shopper. However, shoppers often turn their heads to look around and sometimes they walk with their heads oriented obliquely while watching something interesting or talking. Thus, a key challenge in using smart glasses is detecting the fact that the shopper has turned his head and not updating the direction of his/her path.

One obvious approach is to see if head movements can be detected and distinguished from actual change in direction of shopper's path. However, after studying several shoppers, we realized that it is often not possible to distinguish this based on accelerometer, compass or gyroscope measurements. Consequently, we take a very different approach based on the fact that most Google Glass users also carry their mobile phone along with them (usually in hand or pocket). The key idea is that when the shopper turns his/her head, the Google Glass compass orientation changes, but there is no change in the orientation seen by the mobile phone. Below we describe our scheme to correct for head movements.

1. We estimate the median offset between Google Glass and phone. As most of the time people walk while looking straight ahead, this accurately provides an estimate of the mobile phone's orientation with respect to straight ahead as α.
2. Direction of user's heading is then, $\theta = \theta_{mob} + \alpha$ where θ_{mob} is orientation given by the mobile phone.

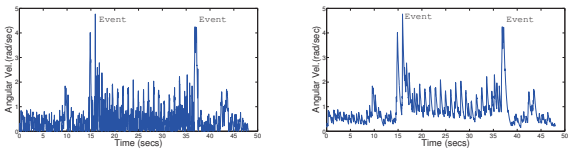


(a) Uncorrected (b) Corrected

Figure 16: Shopper tracks before and after compass correction

Figure 16 depicts the benefits of using the above scheme on a real shopper's track who started and came back to the entrance of the shop after taking about 100 steps (about 70m). This track was computed by simply adding $\hat{e}_k^i$ (the direction vectors). As seen from Figure 16, correction significantly improves the track quality.

Detecting when phone is taken out of pocket: One problem with relying on the shoppers' mobile phones for correcting the direction is that the shoppers might occasionally take their phone out to use



(a) Raw Signal (b) Onset Signal

Figure 17: Absolute value of angular velocity about Z axis

it. However such events can be detected very reliably since they involve sudden motion. Once having detected such an event, we simply re-estimate the median offset from the Google Glass.

We show the events in the Figure 17 (a). We detect the onset of such events based on the absolute value of the signal as shown in Figure 17 (b). We examine 20 traces from 2 users, with each user taking the smartphone out of the pocket and putting it in the pocket for 5 of 10 of their walks. In the other 5 walks, no such event occurs. We compute the CDF of the onset signal over all the walks and vary the detection threshold from top 0.2% to 5% of the signal values. We apply this threshold to quantify true detection accuracy and false alarms. We quantify fraction of the times the event was detected versus the fraction of total seconds that we wrongly detected as pick up events. As shown in Table 5, we can detect up to 85% of phone putting in/pulling out of pocket events without any false alarms and up to 95% of the events with only false alarms during 0.33% of the total time of the walk.

We also tested other cases like holding phone in hand and swaying hands while walking. We find that this action does not change the orientation significantly. For example, we found that the standard deviation of compass heading when user is walking with phone held still in hands is 5.98 over 6 walks by 2 users where each walk was a straight line of about 25 m. It was 8.4 when the users were swaying their hands.

True detection [%]	False Alarm [%]
85	0
90	0.11
95	0.33
100	19

Table 5: Detecting change in phone position

6. ENERGY MEASUREMENTS

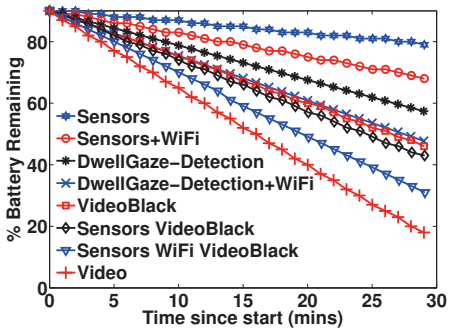


Figure 18: Power measurement on the Google Glass

In this section, we evaluate the power consumption of running the inertial sensors (accelerometer and compass), WiFi, Video capture and our dwell and gaze algorithms on the Google Glass. Since we can not remove the battery and bypass it using a power monitor, we use the android API to monitor battery level [11] and log the remaining battery level every minute. We plot the result for various combinations as shown in Figure 18. We plot the battery level starting from 90% for the next 30 minutes. The legend and the description below are arranged in an increasing rate of power consumption for ease of reading.

Sensors: Here accelerometer and compass were sampled at the fastest allowed rate, *i.e.*, 200 Hz samples were received.

Sensors+WiFi: Since AutoLayout uses WiFi (to enable ThirdEye for non-smart glass users), here we measure the impact of performing back-to-back WiFi scans while simultaneously sensing from accelerometer and compass at 200Hz.

DwellGaze: Here we evaluate the power consumption as the shoppers go shopping during their due course without scanning WiFi. Note that, here VideoBack is turned on only when the glasses detect that the person is gazing (this includes the effect of 33% false positives and 10% false negatives).

DwellGaze+WiFi: Here we compute the same as DwellGaze except with back-to-back WiFi scans for AutoLayout.

VideoBlack: VideoBlack [47] is an application that takes video without displaying on screen so it is energy efficient compared to Video. This is also the default method of taking a video in Google Glass. Video is simply stored, no other computation is performed during the measurement.

Sensors+VideoBlack: During gaze, once video is turned on, we measure the impact of storing video while simultaneously sensing accelerometer and compass at 200Hz.

Sensors+VideoBlack+WiFi: Here we measure the impact of running WiFi scans back-to-back simultaneously with sensing at 200Hz and storing video using VideoBlack.

Video: Here the video on the smart glasses is turned on and stored.

Table 6 depicts the predicted lifetime of various schemes based on our measurements. As seen from Table 6, VideoBlack provides almost 60% higher energy savings than using Video. Further, even VideoBlack is more than 3× expensive compared to using inertial sensors indicating the need for judiciously turning on Video. Continuous WiFi also has a significant impact and typically cuts lifetime significantly. Optimized schemes can be designed to use WiFi sparingly for AutoLayout, however we defer it to future. Thus, the lifetime of ThirdEye is approximately between 70 min (for WiFi turned on continuously) and 90 min (with WiFi scanning turned off).

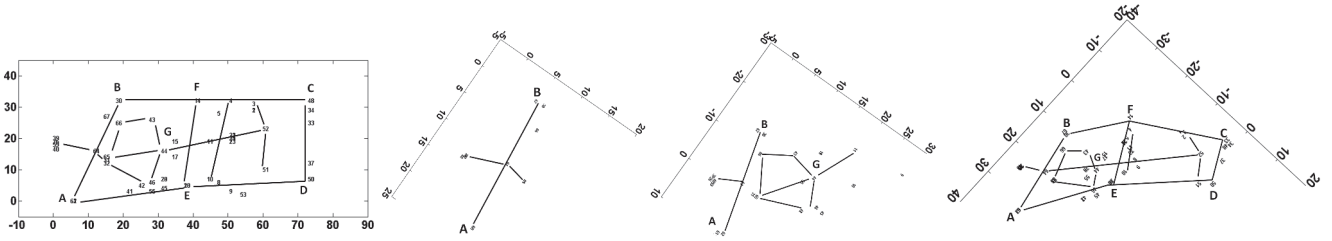
Scheme	Predicted Lifetime [m]
Sensors	265
Sensors+WiFi	134
DwellGaze	89
DwellGaze+WiFi	69
VideoBlack	75
Sensors+VideoBlack	70
Sensors+VideoBlack+WiFi	49
Video	47

Table 6: Predicted Lifetime

7. EXPERIMENTAL EVALUATION

In this section, we evaluate two aspects of ThirdEye: the performance of AutoLayout and the performance of a shopping list reminder application for shoppers without smart glasses.

Experimental Methodology: We tested AutoLayout on 7 shoppers comprising three males and four females of different heights who shopped at a large grocery store (over 2 Km^2 in area) in the US. First, we performed a survey of the store, and measured the relative locations of about 67 different products for obtaining the ground truth. Each shopper was provided with a Google Glass and a smart phone that they would place in their pocket. The smart phone and the Google Glass communicated using Bluetooth. We shadowed each shopper as they went about their due course

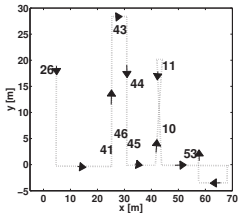


(a) Actual layout (scale in m) (b) Inferred layout after 2 shoppers (c) Inferred layout after 4 shoppers (d) Inferred layout after 7 shoppers

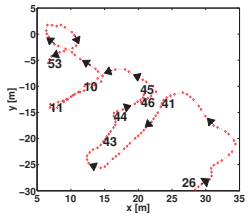
Figure 19: How ThirdEye inferred layout of the shop improves with more shoppers

K	$\eta_{true}(K)$	$\eta_{false}(K)$
1	60%	40%
3	62%	29%
5	67%	26%
10	80%	18%

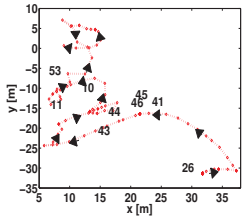
Table 7: Product layout proximity in AutoLayout



(a) Original Path



(b) Path inferred by glass users



(c) Path inferred by smartphone users

K	$\eta_{true}(K)$	$\eta_{false}(K)$
1	17%	85%
3	29%	64%
5	38%	59%
10	60%	40%

Table 8: Performance for Non-Smart Glass Users

Figure 20: How ThirdEye tracks shoppers' paths

of shopping without interfering. Each shopper turned on Google Glass and smart phone at different locations within the shop (hence the start locations were different), visited different parts of the shop while taking different paths. While shadowing we noted the path that each shopper took and the locations where they gazed, for obtaining the ground truth.

Product Layout Generated by ThirdEye: AutoLayout generates a product layout of the store in a virtual 2-D coordinate system by merging the measurements from various shoppers with possibly different stride lengths. Consequently, the virtual coordinate will typically be a scaled, rotated (and possibly reflected), and sometimes distorted version of the original layout. However, one important property that it preserves is proximity relationships *i.e.*, products that are close physically will also be close in the virtual map.

First, to provide intuition into how the original layout compares to that inferred by AutoLayout, in Figure 19 we depict the original and inferred layouts containing 67 different products within a large grocery store (a unique number is assigned to each brand in the Figure). To aid the readers in comparing the original and inferred layouts, we have connected lines between the locations of a few brands (labeled as A,B,C,D,E,F) that are located far away. Figure 19 depicts the evolution of the layouts inferred by ThirdEye as data from more and more shoppers is obtained. The evolution naturally depends on the exact areas where the shoppers visited. As seen from the Figure, after 7 shoppers visited the shop, the inferred layout approximately matches the original layout and proximity relationships are preserved in most cases (albeit rotation and scaling).

How well does AutoLayout Preserve Proximity Relationships? In order to answer this question, for each i^{th} product we create the set $\mathbf{\Psi}_i(K)$ containing its K closest neighboring products using the ground truth data and the $\mathbf{\Omega}_i(K)$ containing its K closest neighbors based on the layout inferred from AutoLayout. We then evaluate average true detections $\eta_{true}(K)$ and average false detections η_{false} as

$$\eta_{true}(K) = \frac{\sum |\mathbf{\Psi}_i(K) \cap \mathbf{\Omega}_i(K)|}{\sum |\mathbf{\Psi}_i(K)|} \quad (4)$$

$$\eta_{false}(K) = \frac{\sum |\mathbf{\Omega}_i(K) - (\mathbf{\Psi}_i(K) \cap \mathbf{\Omega}_i(K))|}{\sum |\mathbf{\Psi}_i(K)|} \quad (5)$$

Here $|\bullet|$ represents cardinality of the set.

As seen from Table 7, AutoLayout predicts the closest product correctly about 60% of the time and reports incorrectly about 40% of the time. Note that predicting the closest product is particularly challenging given the close proximity of the products within the layout. The correct prediction ratio increases as the value of K increases and is about 80% for the set of closest 10 products. This can be very useful for a reminder application that can remind the shopper of a product they intended to buy when they are purchasing another product nearby or help provide a timely discount offer and convince them into purchasing another product that is close by.

How well does AutoLayout Track Shoppers With and Without Glasses? ThirdEye tracks shoppers in a rotated, scaled and potentially reflected coordinate system. Since it also constructs the product layout in this coordinate system, it can place the shopper relative to the products in the shop. Figure 20 (a) depicts the true path (ground truth) and Figure 20 (b) depicts the path inferred by ThirdEye in its coordinate system for one of the shoppers. In this case, the coordinate system is rotated, scaled and reflected. The products that the shopper passes by are marked by unique numbers assigned to them. In the example, the shopper starts at product 26 and walks to product 53, passing through a total of 9 products locations. As seen in Figure 20 (b), ThirdEye successfully tracks the shopper in its own corresponding rotated, scaled and reflected coordinate system. Figure 20 (c) depicts the same path inferred by ThirdEye without using smart glass data, *i.e.*, using mobile-phone inertial and WiFi data. As shown in Figure 20 (c), the inferred path is similar to the actual path though at a reduced accuracy.

How well does the reminder application work for shoppers without Glasses? We consider an application, where the shopper has a shopping list and is reminded to purchase as soon as he comes near one of the products in his shopping list. In order to evaluate how well the inferred map helps this application, we tested AutoLayout on shoppers without using their video from the Glass. We used Au-

toLayout to locate the person at each location (s)he dwelled (using only WiFi and inertial data from his mobile phone) and then used the inferred map to determine all the top K nearest products and then compared with the ground truth layout map.

Table 8 depicts the true and false detection rates as we vary the number of items K. As seen from Table 8, errors are higher than the case of Glass users due to lower localization accuracy. Nevertheless with an increasing K, we are able to cover the top K nearest items more accurately, indicating not only that AutoLayout preserves proximity but even shoppers without smart glasses can benefit from the proximity information.

8. CONCLUSION

In this paper, we have presented ThirdEye, a system for tracking physical browsing by users in retail stores using data gleaned from vision, WiFi, and inertial sensors. Through our evaluation in two large retail stores in US, we show the effectiveness of ThirdEye, despite not requiring any input from the store or the users themselves. In future work, we plan to conduct a more in-depth physical analytics of shoppers based on ThirdEye's ability to track user browsing.

Acknowledgements

This work is supported in part by NSF Grants CCF-1117009 and CNS-1343383. We are grateful to our anonymous shepherd and reviewers for their valuable comments to help improve our work. We thank our friends and lab-mates for their shopping data.

9. REFERENCES

[1] Bing vision. http://en.wikipedia.org/wiki/Bing_Vision.
[2] Code for textonboost. http://jamie.shotton.org/work/code/TextonBoost.zip.
[3] Google search by image. http://www.google.com/insidesearch/features/images/searchbyimage.html.
[4] Nokia point and find. https://betalabs.nokia.com/trials/nokia-point-and-find.
[5] Time Domain. http://www.timedomain.com/.
[6] Tobii. http://www.tobii.com/.
[7] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. of ICCV*, 2009.
[8] M. Alzantot and M. Youssef. Crowdinside: automatic construction of indoor floorplans. In *Proc. of the 20th International Conference on Advances in Geographic Information Systems*, 2012.
[9] M. Azizyan, I. Constandache, and R. R. Choudhury. SurroundSense: Mobile Phone Localization via Ambience Fingerprinting. In *Proc. of Mobicom*, 2009.
[10] P. Bahl and V. N. Padmanabhan. RADAR: An Inbuilding RF-based User Location and Tracking System. In *Proc. of INFOCOM*, 2000.
[11] Battery level api. http://developer.android.com/training/monitoring-device-state/battery-monitoring.html.
[12] K. Bauer, D. Mccoy, B. Greenstein, D. Grunwald, and D. Sicker. Using Wireless Physical Layer Information to Construct Implicit Identifiers. In *Proc. of HotPETs*, 2008.
[13] S. Beauregard and H. Haas. Pedestrian Dead Reckoning: A Basis for Personal Positioning. In *Proc. of WPNC*, 2006.
[14] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless Device Identification with Radiometric Signatures. In *Proc. of MobiCom*, 2008.
[15] A. Campbell et al. The Rise of People-Centric Sensing. *IEEE Internet Computing*, Jul/Aug 2008.
[16] K. Chintalapudi, A. P. Iyer, and V. N. Padmanabhan. Indoor Localization Without the Pain. In *Proc. of MobiCom*, 2010.
[17] T. Choudhury, S. Consolvo, B. Harrison, and J. Hightower. The Mobile Sensing Platform: An Embedded Activity Recognition System. *IEEE Pervasive Computing*, Apr-Jun 2008.
[18] A. J. Davidson, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-Time Single Camera SLAM. *IEEE PAMI*, 29(6), Jun 2007.
[19] B. F. Dieter and F. N. Lawrence. Wifi-slam using gaussian process latent variable models. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.
[20] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell. The BikeNet Mobile Sensing System for Cyclist Experience Mapping. In *Proc. of SenSys*, 2007.
[21] Euclid Analytics. Answers and insights for qsr, coffee, specialty retail, and large format stores, October 2013. http://euclidanalytics.com/.
[22] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communication of the ACM*, 24(6):381–395, 1981.
[23] Google glasses. http://www.google.com/glass/start/.
[24] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki. Practical Robust Localization over Large-Scale 802.11 Wireless Networks. In *Proc. of MobiCom*, 2004.
[25] B. Horn and B. Schunck. Determining Optical Flow. *Artifical Intelligence*, 17, 1981.
[26] ibeacon. http://tech.fortune.cnn.com/2014/02/28/apples-ibeacon-signals-turning-point-for-mobile-engagement/.
[27] N. B. John, J. Heidemann, and D. Estrin. GPS-Less Low Cost Outdoor Localization For Very Small Devices. *IEEE Personal Communications Magazine*, 7:28–34, 2000.
[28] T. Kanda, D. F. Glas, M. Shiomi, H. Ishiguro, and N. Hagita. Who will be the customer?: A social robot that anticipates people's behavior from their trajectories. In *Proc. of UbiComp*, 2008.
[29] P. Krishnan, A. S. Krishnakumar, W. H. Ju, C. Mallows, and S. Ganu. A System for LEASE: Location Estimation Assisted by Stationery Emitters for Indoor RF Wireless Networks. In *Proc. of INFOCOM*, 2004.
[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. of NIPS*, 2012.
[31] S. Lee, C. Yoo, C. Min, and J. Song. Understanding customer malling behavior in an urban shopping mall using smartphones. In *Proc. of ACM Workshop on Mobile Systems for Computional Social Science (co-located with UbiComp)*, 2013.
[32] J. Leonard and H. F. Durrant-whyte. Simultanoues Map Building and Localization for an Autonomous Mobile Robot. In *IROS*, pages 1442–1447, 1991.
[33] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *Proc. of ECCV*, 2012.
[34] H. Lim, L. Kung, J. Hou, and H. Luo. Zero-Configuration, Robust Indoor Localization: Theory and Experimentation. In *Proc. of INFOCOM*, 2006.
[35] D. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proc. of ICCV*, 1999.
[36] Nearbuy Systems. In-store mobile commerce, October 2013. http://www.nearbuysystems.com/.
[37] L. Ni, Y. Liu, C. Yiu, and A. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Network*, 2004.
[38] J. Panda, M. S. Brown, and C. V. Jawahar. Offline mobile instance retrieval with a small memory footprint. In *Proc. of ICCV*, 2013.
[39] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of MobiCom*, 2000.
[40] A. Rai, K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *Proc. of MobiCom*, 2012.
[41] T. Rappaport. *Wireless Communications Principles and Practice*. Prentice Hall, 2001.
[42] P. Robertson, M. Angermann, and B. Krach. Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors. In *Proc. of UbiComp*, 2009.
[43] P. Robertson, M. G. Puyol, and M. Angermann. Collaborative Pedestrian Mapping of Buildings: Using Inertial Sensors and FootSLAM. In *Proc. of ION GNSS*, Sep 2011.
[44] S. Sen, R. R. Choudhury, and S. Nelakuditi. SpinLoc: Spin once to know your location. In *Proc. of HotMobile*, 2012.
[45] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka. Precise Indoor Localization using PHY Layer Information. In *Proc. of HotNets*, 2011.
[46] L. Stampler. That chocolate chip cookie in the checkout aisle is watching you. *Time NewsFeed*, October 2013. http://newsfeed.time.com/2013/10/16/that-chocolate-chip-cookie-in-the-checkout-aisle-is-watching-you/.
[47] Videoblack. http://glass-apps.org/videoblack-google-glass-app.
[48] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No Need to War-Drive: Unsupervised Indoor Localization. In *Proc. of MobiSys*, 2012.
[49] R. Want and et al. The Active Badge Location System. *ACM Transactions on Information Systems*, Jan 1992.
[50] A. Ward, A. Jones, and A. Hopper. A New Location Technique for the Active Office. *IEEE Per. Comm.*, 4(5):42–47, 1997.
[51] J. Xiong and K. Jamieson. ArrayTrack: A Fine-Grained Indoor Location System. In *Proc. of HotMobile*, 2012.
[52] C. You, C. Wei, Y. Chen, H. Chu, and M. Chen. Using mobile phones to monitor shopping time at physical stores. *IEEE Pervasive Computing*, 2011.
[53] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *Proc. of MobiSys*, 2005.