4-2015

QueueVadis: Queuing analytics using smartphones

Tadashi OKOSHIi
Keio University

Lu YU
Institute for Infocomm Research

Chetna VIG
University of California, Davis

Youngki LEE
Singapore Management University, YOUNGKILEE@smu.edu.sg

Rajesh Krishna BALAN
Singapore Management University, rajesh@smu.edu.sg

See next page for additional authors

Citation

OKOSHIi, Tadashi; YU, Lu; VIG, Chetna; LEE, Youngki; BALAN, Rajesh Krishna; and MISRA, Archan. QueueVadis: Queuing analytics using smartphones. (2015). *IPSN '15: Proceedings of the 14th International Conference on Information Processing in Sensor Networks, Seattle, 13-16 April 2015*. 214-225. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/2679

Author
Tadashi OKOSHIi, Lu YU, Chetna VIG, Youngki LEE, Rajesh Krishna BALAN, and Archan MISRA

QueueVadis*: Queuing Analytics using Smartphones

Tadashi Okoshi[†], Yu Lu[‡], Chetna Vig[¶], Youngki Lee[§], Rajesh Krishna Balan[§] and Archan Misra[§]
[†]Graduate School of Media and Governance, Keio University
[‡]Institute for Infocomm Research, A*STAR
[¶]Department of Electrical and Computer Engineering, University of California, Davis
[§]School of Information Systems, Singapore Management University

ABSTRACT

We present *QueueVadis*, a system that addresses the problem of estimating, in real-time, the properties of queues at commonplace urban locations, such as coffee shops, taxi stands and movie theaters. Abjuring the use of any queuing-specific infrastructure sensors, *QueueVadis* uses participatory mobile sensing to detect both (i) the individual-level queuing episodes for any arbitrarily-shaped queue (by a characteristic locomotive signature of short bursts of "shuffling forward" between periods of "standing") and (ii) the aggregate-level queue properties (such as expected wait or service times) via appropriate statistical aggregation of multi-person data. Moreover, for venues where multiple queues are too close to be separated via location estimates, *QueueVadis* also uses a novel disambiguation technique to separate users into multiple distinct queues. User studies, performed with 138 cumulative total users observed at 23 different *real-world* queues across Singapore and Japan, show that *QueueVadis* is able to (a) identify *all* individual queuing episodes, (b) predict service and wait times fairly accurately (with median estimation errors in the 10%-20% range), independent of the queue's shape, (c) separate users in multiple proximate queues with close to 80% accuracy and (d) provide reasonable estimates when the participation rate (the fraction of *QueueVadis*-equipped people in the queue) is modest.

1. INTRODUCTION

Queuing is one of the more mundane, but frustrating, rites of daily life in bustling urban centers of Asia (such as Singapore, Tokyo or Shanghai): we often encounter *significant* queuing delays *multiple times* a day, while having meals in a variety of food and beverage (*F&B*) establishments, withdrawing money (at ATMs), availing of public transport (at bus stops & taxi stands), purchasing groceries (at store checkout counters) and watching movies (at movie theaters). Clearly, accurate, near real-time estimates of such individual and aggregated queuing delays, at the tens of thousands of retail establishments, movie theaters and taxi/bus stands dotting a city, can enable useful new urban applications such as: (a) *Where To Go?*, where an office worker searches for the food stall (among multiple nearby food courts) with the shortest wait time or (b) *Waiting Worth It*, where an F&B retailer in a mall pushes a specific discount to a queuing customer who has been queuing longer than 10 minutes. This paper describes *QueueVadis*, a participatory sensing-based system that first uses an individual's smartphone sensors to identify *individual queuing episodes*. It then uses the properties of multiple such queuing episodes (from different individuals) to (a) separate individuals across *multiple queues* (currently based on the trajectory trace of entire queuing episodes) and (b) derive *robust* estimates of queue properties (specifically the *service time* and the overall *waiting time*). This participatory paradigm offers an alternative to infrastructure-based solutions, such as use of video camera-based analytics [17, 12] or store-specific Wi-Fi APs [15]). In particular, our experience suggests that deploying any infrastructure (even if fairly inexpensive), ubiquitously across hundreds of retail establishments or taxi stands, is extremely hard, especially for the specific "narrow" problem of queuing-analytics. In contrast, *QueueVadis* can be embedded in venue-specific mobile Apps (that often exist for malls, campuses etc.) to opportunistically collect individual-level sensor data–we show that *QueueVadis* provides useful estimates even with participation rates as low as 10%.

Our key challenges, approaches and contributions are as follows.

A Movement-based Queuing Classifier: To enable identification of queuing behavior without any infrastructural aids, we develop a two-tier activity classification model for smartphones that exploits the *repetitive* micro activity sequence of queuing, consisting principally of "standing', interspersed with short bursts of "stepping forward". We implement this basic model efficiently on commodity smartphones; moreover, to monitor individual queuing behavior with very low energy overhead, we use coarse-grained location triggers to activate such classifiers.

Adaptation to Varying Queue Service Times: Based on empirical queue observations, collected at over 39 real-world locations across two Asian cities (Singapore and Tokyo), we found that queues exhibit significant variability, in terms of service rate and overall queuing delay. Moreover, even within a single queue, there were both short (few minutes) and longer time-scale (hour of the day) variation in these metrics. To detect queuing accurately and robustly across such variations, the *QueueVadis* client employs multiple concurrent tier-2 classifiers (operating at different timescales).

Robustness to Human Behavioral Artifacts: Our empirical measurements also showed two important usage-based artifacts: *queue shape / orientation* and *premature leaving behavior*. We found that users line up at different queues within a single crowded venue, such as a food court, in various organic shapes: sometimes in a straight-line, but often in a randomly curvy and even 'snaking' (with 180deg turns) fashion. Moreover, in queues such as at F&B stores and taxi-stands, users can occasionally leave a queue prematurely (often out of impatience). We show that our *QueueVadis* client's *orientation-independent* technique for classifying behavior achieves robust queuing detection across a variety of real-world

* A play on "QuoVadis", Latin for "Where are you going?"

IPSN '15 April 14 - 16 2015, Seattle, WA, USA

ACM 978-1-4503-3475-4/15/04 ...$15.00.
http://dx.doi.org/10.1145/2737095.2737120.

queue shapes and orientations. Moreover, by utilizing appropriate statistical filtering of outliers, the *QueueVadis* server can identify and eliminate spurious estimates from individuals who depart prematurely.

Handling Low Participatory Rates: A participatory system such as *QueueVadis* must deal with the *bootstrapping* problem, providing reasonably useful service and wait time estimates even when the fraction of individuals with "participatory probes" is relatively low. To achieve this, *QueueVadis* utilizes a unique approach of obtaining multiple service time estimates from the time intervals between successive "shuffle forward" activities of each participating individual–the multiplicity of such observations helps *QueueVadis* be very resilient to wide variations in the participation rate (the standard deviation of service time estimates show only a 12.7% increase, when the fraction of queuing individuals with *QueueVadis* drops from 100% to 10%).

Multiple Queues in Close Proximity: Across public venues in Asia, multiple queues occur in very close proximity (e.g, F&B outlets in a food court), often separated by less than 1–2 meters. Identifying multiple distinct queues, and separating individuals into these queues, is a non-trivial challenge indoors, as practical RF-based techniques (e.g., the Wi-Fi AP based approach in [15]) currently have location errors of ±4–5 meters [3] or higher. To address this challenge, *QueueVadis* utilizes a novel classification technique to identify the number of distinct queues, and then associates each queuing individual to his/her corresponding queue. The classification combines (i) temporal correlation features to identify phase-shifted movement behavior among individuals in the same queue, with (ii) a coarse-grained trajectory matching technique (utilizing the magnetic compass) to separate out queuing at multiple closely located queues. Experiments on real-world venues shows a disambiguation accuracy of over 70% (for the "Where to Go?" scenario, where one can use the movement behavior over the entire queuing episode retrospectively), even at low (20%) participation rates.

Based on extensive testing of *QueueVadis* across 23 different real-life queues in Tokyo and Singapore, we show that *QueueVadis* is able to infer the duration of individual queuing episodes with a median error of less than 10% (an estimation error of 1-1.25 minutes, given a median wait time of around 10 minutes during peak hours at F&B outlets) and provide estimates of the service and wait times with a median error of 10-15% (a median error of approx. 9 secs in detecting the onset of queuing at typical F&B outlets).

2. MOTIVATING SCENARIOS

Queuing is very clearly a significant problem in various places–note that restaurant wait times in Singapore have been known to reach 40 minutes or longer [1], while a Tokyo Disneyland attraction made headlines in 2012 with a 500 minute wait time [2]! We envision two key use cases that can leverage upon *QueueVadis'* automated detection of queuing behavior:

1 **Where To Go?**: In this scenario, an office worker looking to get a quick lunch is trying to decide which of several queues (at multiple nearby food courts) has the shortest waiting time. To satisfy this scenario, *QueueVadis* must estimate the wait times of all of the queues in the area.

2 **Waiting Worth It**: In this scenario, a coffee shop in a mall would like to identify the specific individuals who have been waiting for longer than 10 minutes, and push them an additional discount, so as to minimize customer dissatisfaction.

These two scenarios capture the fundamental objectives that drive the design of *QueueVadis*. They differ in their accuracy and recency requirements: (a) *Where To Go?* will be effective with relatively coarser ballpark estimates of queue wait times (users are unlikely to require precise wait-times); (b) *Waiting Worth It*, on the other hand, requires more accurate and relatively real-time estimates derived from users *currently* in the queue. Moreover, *Waiting Worth It* also requires the *identity* of the specific queuing individual–this is extremely hard to do reliably, in crowded urban spaces, using infrastructure-based approaches such as video analytics.

3. UNDERSTANDING REAL QUEUES

Before building *QueueVadis*, we analyze the properties of human-generated queues, based on extensive observations made in Singapore and Tokyo. We begin with a concise description of the fundamental queue properties that we desire to study, followed by an analysis of queuing at real-world locations, in terms of the variations observed in those properties and in the orientation/trajectory of individuals who physically queue at those locations.

3.1 Mathematical Queue Properties

From queuing theory, we identify the following key parameters of any single queue:

1 **Service time ($\frac{1}{\mu}$):** The service time is a random variable that denotes the amount of time taken by the currently-served customer to complete her transaction.

2 **Wait time (T_w):** This random variable denotes the total time taken by a newly arriving customer to complete the desired transaction — note that it includes the queuing time (T_q) and the service time. In other words, for any individual customer i, $T_w(i) = T_q(i) + \frac{1}{\mu}(i)$.

In addition to these parameters, the dynamics of the queue are governed by the arrival rate (number of arrivals per second) of customers (denoted as λ). In practice, we cannot observe (at least in a participatory manner), the true arrival rate λ, or the number of people (N_q) currently in the queue. We now study the variation in these properties, as observed empirically at a variety of real-world locations.

3.2 Variations in Real-World Queues

We videotaped the evolution of queues (as well as the movement patterns of each individual person) at 39 venues across Singapore and Tokyo (on multiple days, spanning several months), including 5 airport check-in counters, 5 airport boarding areas, 10 F&B locations, 10 movie ticketing venues, 7 taxi stations, and 2 different amusement park ride areas. Table 1 shows the different types of queues we explicitly observed, along with the characteristic functions performed at the respective service counters, as well as the computed values for: *a)* the service time statistics (avg, min, max, stdev) for each type of queue, and *b)* the duration of the contiguous *stepping* and *standing* activities (avg, min, max, stdev) during queuing activity. (At each queue venue, the *stepping* and *standing* activity durations of at least 50 samples by 10 different people were measured–e.g., numbers in "Airport check-in" are based on 250 samples from 5 venues.)

3.2.1 Service Times for Different Queuing Types

As may be expected, there is a clearly a wide variation in the average values of service times ($\frac{1}{\mu}$) across different categories/types of queues (e.g., movie theater queues move fast, while airport check-in takes over 1.5 minutes). Even in the same category, two different queues could have significantly different mean values–e.g., in the case of the two amusement park queues, one has users stepping forward (implicitly indicating that a visitor entered the attraction) once every 8 secs, whereas the other queue shows similar stepping forward movement every 4 secs. Also, for certain specific queues, there is a significant variation in the service times (when observed

Location	Queue Type (number of venues)	Characteristics	Service Interval (s)				People's Movements in the Queue (s) Stepping				Standing			
			Min.	Ave.	Max.	Stdev.	Min.	Ave.	Max.	Stdev.	Min.	Ave.	Max.	Stdev.
Singapore	Airport check-in (5)	V, N, P, Sel	10.7	102.1	421.7	136.9	1.7	5.7	16.0	2.8	2.0	55.3	203.0	42.7
Singapore	Airport boarding (5)	V, Sec	4.3	7.8	15.6	2.9	0.5	2.9	9.1	1.6	0.7	5.3	14.5	3.0
Singapore	Food and beverage (5)	P, Sel	10.3	32.9	77.0	17.8	0.6	2.2	4.4	0.9	1.5	20.0	54.6	12.7
Singapore	Movie ticketing (5)	P, Sel	5.0	7.9	11.5	2.3	1.3	3.4	10.2	1.7	1.2	21.5	42.0	13.6
Singapore	Taxi station (2)	Sel	9.6	50.8	85.4	27.8	1.2	2.2	3.6	0.7	19.5	46.8	88.6	20.3
Singapore	Amusement park ride 1	V	N/A	N/A	N/A	N/A	1.0	8.0	30.6	6.0	1.0	7.9	37.5	6.2
Singapore	Amusement park ride 2	V	N/A	N/A	N/A	N/A	1.0	4.4	21.7	4.0	1.1	53.4	145.0	46.1
Tokyo	Food and beverage (5)	P, Sel	19.0	51.9	128.2	28.4	1.7	3.1	5.1	0.9	1.7	37.1	132.0	30.1
Tokyo	Movie ticketing (5)	P, Sel	6.5	23.1	39.9	15.3	1.2	2.4	4.2	1.8	1.1	30.5	67.6	23.3
Tokyo	Taxi station (5)	Sel	8.6	15.9	47.2	8.5	0.6	2.6	5.1	1.1	1.4	13.3	66.8	11.7

For the *Characteristics* column, V = Validation (checking of tickets etc.), N = Negotiations (baggage allowance, flight routing etc.), P = Purchase (exchange of money), *Sec* = Security (security check of documents, items, etc), and *Sel* = Selection (picking specific seats, food products, etc.). *N/A = Not Available due to difficulty in observation*

Table 1: Taxonomy of Queues

over a multi-day time period)— a classic example is the airport check-in queue, where the standard deviation in the service time (136.9 secs) exceeds the average value (102.1 secs).

3.2.2 Temporal Variation in Service Time

We also collected additional longitudinal data at multiple times (both off-peak and peak periods) on different days, over a 1 week span, at the beverage stall in the food court of a Singapore-based university. Figure 1 plots the service time and its standard deviation, experienced by 15 consecutive people queuing, at three different days (Monday, Wednesday & Friday) during the morning session, whereas Table 2 plots the mean values of service time at different times on a normal working day. Clearly, even for a specific queue, the service time is still highly unpredictable even over shorter timescales. For example, in Figure 1, we observe that the service time on Monday varied by a factor of 9 (20sec-180sec), and on Friday by a factor of 7 (20sec-140sec). Moreover, the service time distribution (and mean) exhibits clear time-of-the-day effects as well: from Table 2, we observe that service during lunch is much faster and less variable (mean of 27.6 sec and std. deviation of 9.2 sec), compared to service during breakfast or dinner times.

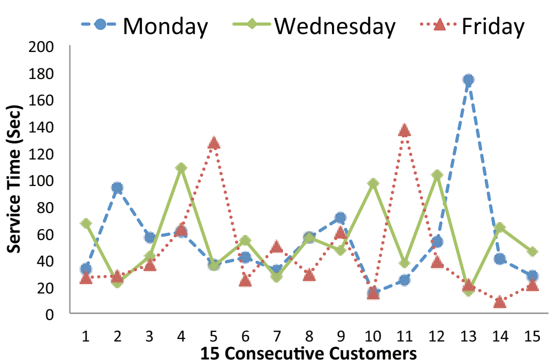


Figure 1: Service Time Variation (F&B@Uni 9:30-10am)

Time of Day	Service Time (secs) Min	Max	Mean	Std. Dev.
Breakfast (9:30-10am)	15	174	52.5	36.7
Lunch (12:30-1pm)	15	40	27.6	9.2
Dinner (7-7:30pm)	8	97	46.4	25.6

Table 2: Time-of-Day Variations in Service Time

3.2.3 Diversity of Queue Orientations

To understand the behavior of multiple proximate queues, we empirically recorded, over a period of 1 hour during the busy lunch period, the evolution of *all* the queues that formed in the food court of the university in Singapore. Figure 2 shows a schematic of the layout of the food court (with the different food stalls arranged along the periphery and the rectangular island in the center), along with the orientation/shape of the queues that formed for the different stalls (each stall is roughly 2.5-3 meters wide). The figures shows that, in dense environments, where the tight layout of tables and the sheer volume of crowds create some natural barriers to queuing, the queues evolve in an *organic* fashion, rather in orderly straight-lines. In certain cases (e.g., at the 'Malay' food stall), the queue exhibited a horseshoe pattern, doubling back on itself. We also see that nearby queues exhibit discernibly different queuing trajectories, an observation that we shall further explore (in Section 8) for disambiguating among multiple queues.

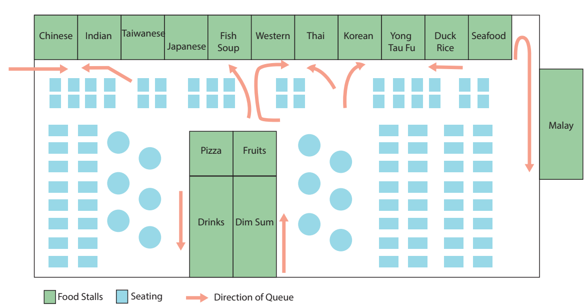


Figure 2: Multiple University Food Court Queues (12:15-1pm)

3.2.4 Queue Joining/Leaving Behavior

We also carefully analyzed the video traces for situations where an individual either *cut-in* to the queue (joining at an intermediate point), or left *preemptively* (before reaching the end of the queue). Overall, from the approx. 210 individuals (with an overall observation duration of over 80 minutes) that we visually analyzed, we found such incidents to be rare–only 1.46% of the users (i.e., 3 users) left the queue preemptively, while a similar number (3 users) cut-in to the queue (only at the airport). This suggests that out of order arrival or departures are uncommon in most "normal" queues. However, in several cases, we observed the phenomenon of "group-driven lingering", where individuals would wait at the service counter for other members of their group to finish their transactions, before leaving the queuing area together.

4. QUEUEVADIS

This section introduces the *QueueVadis* system architecture, for detecting individual queuing episodes and analysis of aggregate queue properties. We first outline our design goals and then describe the overall *QueueVadis* architecture.

4.1 Design Goals & Assumption

Based on the realization that real-world queues exhibit a lot of diversity, across multiple attributes, we focused on the following goals for *QueueVadis*:

- *Supporting In-Service, Physical Queues of Arbitrary Shape: QueueVadis* seeks to monitor and estimate properties of *in-service, physical* queues of arbitrary shape, that capture the majority of queues commonly observed at restaurants, food courts, movie theaters, supermarkets, taxi stands and similar venues. We explicitly do not focus on *virtual queues* (e.g,. deli-style queues where individuals take a number and then wait to be called), both because such queues have no distinguishing physical movement characteristics, and because the queue size is directly available from the corresponding infrastructural component (e.g., the ticket dispensing machine).
- *Adaptation to Varying Service Times:* As real world queues show significant variance in service times, *QueueVadis'* classification logic should be able to accommodate large (at least two orders of magnitude) variations in $\frac{1}{\mu}$.
- *Disambiguation for Multiple Proximate Queues:* The *QueueVadis* server should be able to identify if two customers are queuing in the same or different queues, which occur in close proximity (making them difficult to separate out on the basis of practical localization technologies).
- *Robustness to Variable Participation Rate: QueueVadis'* ability to provide estimates for queuing and service times should degrade gracefully, as the set of observed samples (the proportion of queuing customers who use *QueueVadis*) becomes smaller. Moreover, the ability to perform queue disambiguation should be robust to changes in the participation rate.
- *Minimize Detection Latency:* To support scenarios such as *Waiting Worth it*, *QueueVadis* must be able to quickly detect the *onset* and the *end* of a queuing episode, while avoiding spurious oscillations (between "queuing" and "non-queuing") and false positives.
- *Resource Efficiency:* The *QueueVadis* client should minimize the energy overheads associated with the sampling and processing of sensor (accelerometer and magnetic compass) data, by either modifying the processing pipeline and/or limiting the duration during which such queuing-related sensing is activated.
- *Use No Additional Infrastructure:* As stated in the introduction, one of the key goals of *QueueVadis* is to explore the possibility of smartphone-based queue detection. Venue owners (malls, airports, etc.) can use *QueueVadis* with their existing applications (which they are already developing) to obtain queue detection capabilities *without additional infrastructural investments* (and associated feasibility, tendering, installation, and support costs). However, *QueueVadis* is complementary to, and thus backward and forward compatible with, infrastructure-based queue detection solutions.

QueueVadis' design assumes (but does not mandate) the existence of an external service that can track a user's location at coarse-grained granularity (e.g., with $\pm 8-10$ meter accuracy). This has been empirically demonstrated to be possible in many public spaces. Such location monitoring serves as a useful trigger for the *QueueVadis* client– *QueueVadis* is triggered only when the user is in the vicinity of locations where queuing is plausible (e..g, near or at the food court, near the taxi stand), and is kept dormant when the user is at other implausible locations (e.g., working inside her office, or at the gym).

4.2 Architecture of QueueVadis

Figure 3 shows the overall *logic-flow* in *QueueVadis*: the sensing on each *QueueVadis* client is triggered when the user is at certain relevant locations (using an external location service that we do not explore further). The output from each *QueueVadis* client is then aggregated at the *QueueVadis* server, which first *disambiguates* users into multiple separate queues (if this is necessary– i.e., when two nearby queues cannot be distinguished on the basis of location alone), and then computes the *aggregate* property of each queue separately. Figure 4 shows the details of the client and server components, as explained below.

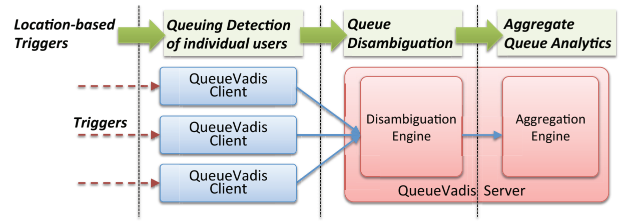


Figure 3: Overall Logic Flow of *QueueVadis*

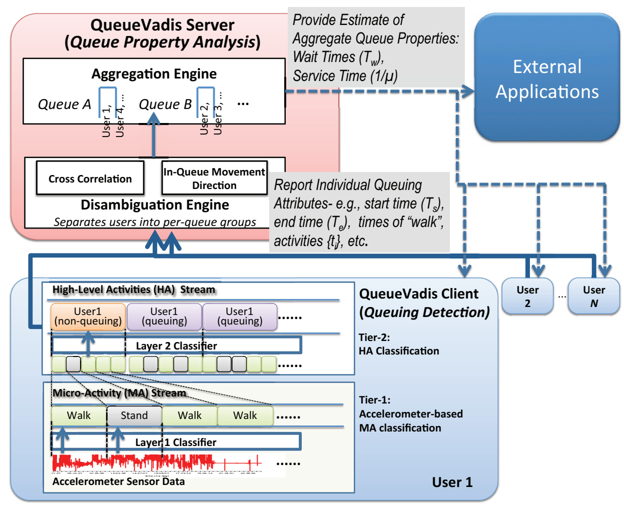


Figure 4: Architecture of *QueueVadis* Server and Client

Client-side: The *detection of an individual queuing episode* is performed by the *QueueVadis* client, an application running on an individual's mobile device. To support in-service, physical, queues of *arbitrary* shapes, we utilize the one common property of all physical queues–namely, that a user performs a repetitive sequence of micro (or postural) activities that principally involve "standing" for a while, interspersed with short bursts of "stepping (shuffling) forward". The *QueueVadis* client detects queuing activity by using sensing data from the accelerometer and then performing real-time activity recognition (similar to approaches such as [10, 6]). To accommodate the real-world variability in service times at different venues, *QueueVadis* utilizes multiple concurrent classifiers, each tuned to a different service time regime.

Server-side: The server receives the various queuing-related attributes, including the *start time* (t_s) when the individual started a queuing episode, the *end time* (t_e) when the episode was detected to have ended, and the time instants (t_s^i) when the individual shuffled forward in the queue, from multiple *QueueVadis* clients. In situations where the coarse location does not readily map to a single queue but can be associated with multiple queues (e.g., adjacent stalls in a mall's food court), the server first uses its *Disambiguation Engine* to separate out and group clients into multiple distinct queues. The *Aggregation Engine* then operates on a per-queue basis. It intelligently combines the attributes from clients that are part of the same queue to estimate various *aggregate* properties about the queue, including the queue's *service time* ($\frac{1}{\mu}$) and *wait time* (T_w). Finally, these statistics are then disseminated back to the clients or to external applications (e.g., one that displays queuing delays at different retail establishments).

In the next two sections, we study the low-level choices for the queuing activity classification logic and then the overall design and implementation of the *QueueVadis* client. Subsequently, in Section 7, we will describe and empirically evaluate the server-side analytics to compute the queue properties (service and wait times), given reports from *QueueVadis* clients in that queue, while Section 8 will describe how the server's Disambiguation Engine separates people into different queues, if needed.

5. TWO-TIER QUEUING DETECTION

In this section, we investigate the design of the classifier used on *QueueVadis* clients to detect queuing events. Following prior work in hierarchical activity recognition [7, 16], our hierarchical classifier depicted in Figure 4 views queuing as a *semantic* or High-level Activity (**HA**), that is inherently composed of a variable sequence of low-level or *Micro-Activities* (**MAs**), derived from features of the phone-embedded accelerometer sensor. Let N_m be the number of distinct MAs (e.g., N_m could be 4 and equal to the set {*walking, stationary, stepping, others*}), and N_h be the number of possible HAs (for example, N_h could be 4 and given by the set {*queuing, dining, browsing, unknown*}). The classification process then consists of the following two steps:

1. **Lower Layer MA classification:** In this layer, the raw 3-axis accelerometer stream (sampled at f Hz) is first partitioned into a series of non-overlapping *frames* of relatively small duration (e.g., 2-5 seconds), denoted by T_f, and features computed over the $f * T_f$ samples in each frame are used to classify each frame into one of the N_m labels.
2. **Higher Layer HA classification:** In this layer, the stream of MA labels is first partitioned into a set of non-overlapping *windows*, each with W_Q consecutive MA labels. A set of *high-level features*, computed over these W_Q elements, is then used to classify the entire window into one of N_h *HA* labels.

To utilize this framework, we need to select appropriate values for the following parameters:

- T_f: The frame duration; intuitively, an overly long T_f could lead each MA to be mis-classified if the user actually performed multiple *MA*s within it, whereas an unduly low T_f could be vulnerable to transient noise (e.g., a slight jerk while walking).
- W_Q: Intuitively, W_Q should be just long enough to capture the characteristic pattern of remaining *stationary*, followed by a short period of *stepping forward*. If W_Q is too long, then this characteristic movement may fail to be distinguished from other non-queuing behavior (e.g, walking to the coffee shop before queuing, and then sitting down after queuing); if W_Q is too short, then the characteristic feature may not manifest itself at all (e.g., if a person moves forward only once every 5 minutes in a queue, then W_Q=1 minute may simply show a sequence of "standing" activities.
- N_m/N_h and the set of *MA*/*HA* labels: While a smaller value of N_m is likely to improve the lower-layer classification accuracy by reducing the number of distinct activity labels, it may also lead to poorer higher-layer classification results caused by reduced discriminative capabilities between HAs. In this paper, our focus is **solely on queuing-related analytics**; accordingly, we use $N_h = 2$, given by *{queuing, others}*, where "others" captures all non-queuing activities.

5.1 Micro-Benchmarking Study

To select suitable values for T_f, W_Q and N_m, we conducted a micro-benchmark study of using a smartphone accelerometer sensor to infer queuing behavior.

5.1.1 Data Traces & Feature Selection

To perform this study, we recruited 10 participants, who each carried a smartphone (Samsung Galaxy S III) and performed a variety of MAs and HAs, as described next.

MA Data: Each participant performed a set of 9 pre-established MAs (each for 120 seconds): namely, (1) walking, (2) stepping, (3) jumping, (4) jogging, (5) riding bicycle, (6) standing, (7) sitting, (8) climbing up/down the steps, and (9) going up/down with elevator. The accelerometer data being recorded (at a frequency $f =$ 50Hz) by our custom Android application; to filter out initial measurement noise, we kept only 90 seconds in the middle of each measurement.

HA Data: To collect HA observational data, we asked each participant to perform a natural "queuing activity" at 4 venues in Singapore (2 F&B stalls in a food court, and 2 ticketing counters at two different movie theaters), while carrying two time-synchronized phones–one with our data collection application and the other with a timestamping App that the participant used to record the (start, end) times for each queuing episode. At each venue, we collected 15 queuing episodes, corresponding to 5 samples (one per participant) for each of 3 on-body phone positions ("back pocket", "front pocket", and "in palm"). Each participant also collected the raw accelerometer data, for approximately 4 hours a day for 4 days as they went about their daily lives at work (explicitly excluding queuing), thus providing us additional data corresponding to the HA=*others*.

Feature Vector for MAs & HAs : Table 3 overviews the features used in each tier. For the tier-1 classification of MAs, we use a set of 22 commonly used time-domain and frequency features, widely used for accelerometer-based activity detection. For the tier-2 HA classification, we extended the *duration/frequency*-based classification approach from [16], where the feature vector is N_m-dimensional, and the i^{th} element consists of the number (count) of the i^{th} MA label observed in any given HA window (of size W_Q). In addition, we also added a few extra features corresponding to the (*min.*, *max.*, *mean*, and *variance*) of the durations for each of these vector elements.

Tier-1	Time Domain	Mean $(\bar{x}, \bar{y}, \bar{z})$ Magnitude of Mean $(\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2})$ Variance $\{var(x), var(y), var(z)\}$ Correlation $\{corr(x,y), corr(y,z), corr(x,z)\}$ Covariance $\{cov(x,y), cov(y,z), cov(x,z)\}$
	Frequency Domain	Energy $(\frac{\sum_{j=1}^{N}(m_j^2)}{N})$, m_j is FFT component Entropy $(-\sum_{j=1}^{n}(p_j * log(p_j))$, p_j is FFT histogram
Tier-2	M : Number of each MA in the frame D_{min}: Minimum duration of each MA in the frame D_{mean}: Mean duration of each MA in the frame D_{max}: Maximum duration of each MA in the frame D_{stdev}: Standard deviation of the duration of each MA in the frame C: Number of MA changes in the frame	

Table 3: Selected Features Used for Activity Recognition

5.1.2 Parameters for MA and HA Detection

We experimented with three different sets of MA labels:

1 *MA-2*, a *coarse-grained* set of 2 MA labels, consisting of just *{Stationary (sitting, standing) and Moving (all others)}*.

2 *MA-3*, a *medium-grained* set of 3 MA labels, consisting of just *{Stationary (standing), Moving (stepping, walking), and Others (all others)}*.

3 *MA-4*, a *fine-grained* set of 4 MA labels, consisting of the activities *{Stationary (standing), Stepping, Walking, and Others (all others)}*,

These three choices for MA labels helps us to understand how the HA classification accuracy would change, given finer or coarser grained locomotive labels at the lower-tier.

5.1.3 MA & HA Accuracy

Figure 5 shows the MA-level accuracy for the all MA labels, as a function of the frame length T_f. The results are obtained through an 100-fold cross validation study performed using the J48 classifier implemented in Weka[9]. The results show that the classification accuracy for MA-2 is clearly superior to MA-3 and MA-4 (as expected), remaining above 97% for all values of T_f. The classification accuracy for MA-3 and MA-4, on the other hand, increases as the frame size is increased, reaching approximately 84.1% (MA-3 when $T_f = 3$secs) and approximately 76.4% (MA-4 when $T_f = 3$secs) as their peaks.

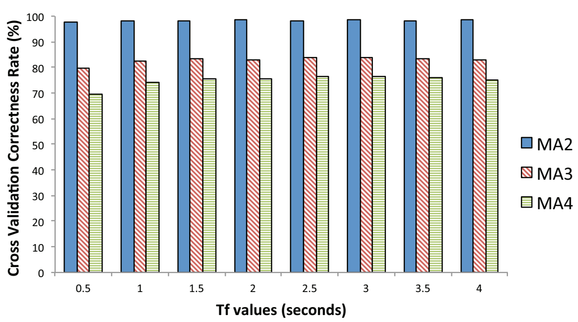
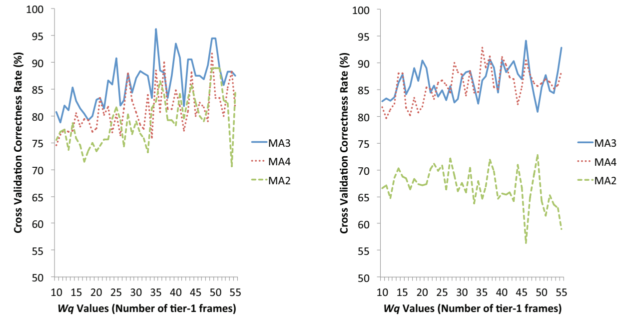


Figure 5: MA Detection Accuracy vs. Frame Length

We next plot, in Figure 6, the choice that the MA labels has on the higher-layer 'queuing detection' accuracy vs. the window size W_Q; we use $T_f = 3$secs, which provided the overall highest MA-level accuracy in MA detection). Given that different *types* of real-world queues have significantly different service times (observed in Section 3.2.1), we expect that the impact of W_Q will be different for different queue types. Accordingly, the accuracy values are plotted separately for two different types of queuing venues in Singapore: *F&B* & *Movie Ticketing*). We observe two important characteristics:

a) When we focus on HA classification accuracy, MA-3 and MA-4 based classification outperform MA-2 (clearly describing MAs in terms of *moving* vs. *stationary* does not help to separate queuing behavior as strongly), often providing classification accuracy gains of 10-20%. This motivates us to *choose MA-3 as our preferred set of MA labels in the subsequent design and implementation of* QueueVadis.

b) The optimal choice of the window size W_Q (based on the best cross validation rate) depends on the category (type) of the queue, and we clearly see that the accuracy can be poor if W_Q is too small. Intuitively, for queues with larger service times, W_Q must be larger, as the user will exhibit the characteristic "stepping" movement only over longer time intervals. Even for the same queuing type, the highest accuracy occurs at different values of W_Q, due to the short and medium term (as observed in Figure 1) fluctuations in service times.



(a) F&B (b) Ticketing

Figure 6: HA Detection Accuracy vs. Window Size

5.2 Key Takeaways

Our detailed micro-study of MA and HA classification at multiple queuing venues leads to a few interesting design choices for the classifier in the *QueueVadis'* client:

- *Micro-Activity Settings:* We choose i) T_f= 3secs, ii) and $N_m = 3$, given by {*Stationary, Moving, Others*}, as the most robust parameter choices for queuing detection.
- *Choice of* W_Q*:* We realize that there is no unique and optimal choice for W_Q, even for a single queue *type*, as the service times can show high variance. Hence, we shall next describe how the *QueueVadis* client uses *multiple, concurrent layer-2 classifiers*, each tuned to a specific W_Q.

6. QUEUEVADIS CLIENT

In this section, we describe how the *QueueVadis* client operates on a personal mobile device to detect an individual's queuing episodes. The *QueueVadis* client operates in three-states, illustrated in Figure 7: *Detection Initiation (DI)* , *Concurrent Detection (CD)* and *Queuing Termination (QT)*. In the standard operational paradigm, *QueueVadis* starts off in the low-energy *DI* state (with the accelerometer sensor and the classifier logic turned off) and remains there until it receives external "location triggers", indicating that the individual *may* potentially be queuing. At that time, *QueueVadis* transitions to the *CD* state, where it activates the accelerometer sensor and the classifier, so as to detect the onset of a queuing activity. Once a queuing activity is detected, *QueueVadis* transitions to the *QT* state, where it now instead begins to look for markers that the ongoing queuing activity has ended, at which point it transitions back to the *DI* state.

Detection Initiation (DI): *QueueVadis* assumes the existence of an external location tracking system, that triggers the transition to the *CD* state, only when an individual's location suggests that queuing may be possible. In our experimental system deployed on the SMU campus, we currently have such a Wi-Fi based location system operationally deployed. As an episode of queuing is highly likely to occur on or near only certain locations (e.g., the food court on the campus or at the movieplex), we expect the *QueueVadis* client to stay in the low-power *DI* state for most of the day (e.g., when the user is working in her office or riding on the bus).

Concurrent Detection (CD): When the individual is at locations where queuing may be possible, the *QueueVadis* client activates the two-tier classifier described previously. One important difference from past work is that the *QueueVadis* client activates *multiple tier-2 (HA) classifiers, each with a different value of* W_Q *in parallel*, to

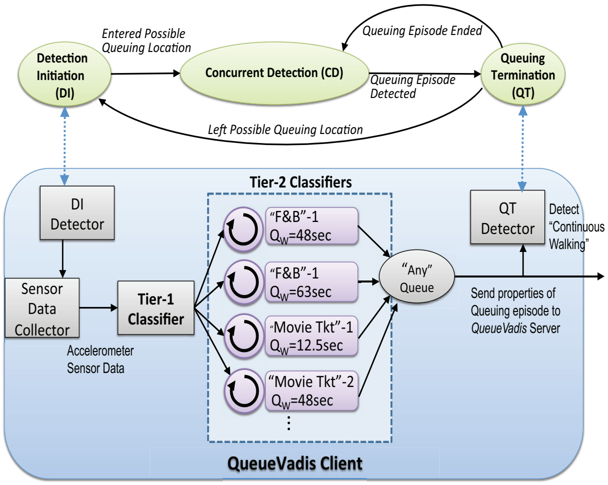


Figure 7: Client Implementation & State Transitions

accommodate the possibly wide variations in $\frac{1}{\mu}$. More specifically, the current implementation of the *QueueVadis* client implements 4 tier-2 classifiers (details of each of the window choices are provided in Table 4) for each *type* of queue corresponding to W_Q values of $Avg - Std.Dev$, Avg, $Avg + Std.Dev$ and $Avg + 2 * Std.Dev$ of $\frac{1}{\mu}$ for that queue type (to capture situations where the service time is shorter or longer than the average). Moreover, to support multiple queue types (e.g., F&B, Movie ticketing, Taxi Stand etc.), the classification engine uses 4 classifiers for each of these types. Each tier-2 classifier also contains an independent *Smoother* component that filters out transient values in the HA output stream (e.g., it outputs "queuing" only if two consecutive labels of the HA stream indicate "queuing".) Most importantly, *QueueVadis* declares that a person is queuing if *any of the (smoothed) concurrent tier-2 classifiers* has an output of "queuing", indicating that the user's movement pattern matches at least one of the many possible patterns of service times (ranging from fast to moderate to slow).

Queue Type	Tier-2 Window Size (W_Q) avg. - std. dev.	avg.	avg. + std. dev.	avg. + 2*std. dev.
Airport Check-in	5	20	35	51
Airport Boarding	1	3	4	6
F&B	3	7	12	16
Ticketing	3	18	13	18

Table 4: Concurrent tier-2 Classifiers

Queuing Termination (QT): Careful analysis of our videos of queuing behavior collected at multiple locations showed that, in almost all cases, *each individual walked continuously away from the counter having received service*. Moreover, the typical "stepping duration", while the customer was waiting in the queue, was seen to be around 5.7 secs. Accordingly, *QueueVadis' QT* logic detects the end of a queuing episode when 3 consecutive layer-1 frames (corresponding to a duration of $2{*}T_f(3)$= 9 seconds) indicate 'walking' as the likely MA. This assures early detection of the end of a queuing episode, while preventing *QT* from being falsely triggered by movements while still in the queue.

6.1 Client Implementation

Our current implementation is written in Java for the Android 4.x platform and implements the overall pipelined, multiple tier-2 classifier design outlined in Figure 7. The application data processing pipeline, implemented entirely in memory, is as follows: (1) retrieval of the raw sensor data, (2) lower-layer feature vector extraction, (3) MA sequence computation (once every 3 secs) (4) higher-layer feature vector extraction, followed by (5) higher-layer HA stream extraction (multiple in parallel). In addition to generating these labels, the client also computes the start time T_s and end time T_e of each queuing episode, as well as the *specific movement sequence while queuing* and transmits it to the *QueueVadis* server.

6.2 Client Evaluation

We now evaluate the accuracy, latency, and power consumption of our *QueueVadis* client.

6.2.1 User Study

To evaluate the performance of the *QueueVadis* client, we conducted multiple user studies at several real world venues around the city. We conducted 15 sessions of experiments at 6 different venues in 2 different countries, as shown in Table 5: 2 sessions (one during a *peak* period at noon and another during *off-peak* afternoon hours) at an F&B outlet on a Singapore-based university campus, and one session each from 2 different movie theater ticketing counters. At the F&B outlet in Singapore, each study session consisted of 10 participants queuing up consecutively to buy a beverage of their own choice; at the theaters, we had 5 participants queuing up consecutively and actually buying a movie ticket at the counter; similar studies were conducted in Tokyo as well. Besides the *QueueVadis* client, each person also carried a second phone (Samsung Galaxy S III) for sensor data collection and time-stamping.

Country	Queue Type	Venue Name	Number of Participants	Number of Sessions
Singapore	F&B	University Cafe	10	4
Singapore	F&B	University Food Court	13	27
Singapore	F&B	Starbucks	6	2
Singapore	Ticketing	GV Movie Theater	5	2
Singapore	Ticketing	Cathay Movie Theater	5	1
Tokyo	F&B	Starbucks	5	4
Tokyo	Ticketing	"109" Cinemas	6	3

Table 5: Queue Detection User Studies

6.2.2 Queuing Detection: Accuracy and Latency

We first check: can the *QueueVadis* client detect an individual's queuing activity correctly? To investigate this question, we fed the entire trace of each individual's activity trace (i.e., their raw accelerometer stream), consisting of the queuing episodes at the designated venues through our *QueueVadis* implementation. We noted that *QueueVadis* offered **100%** accuracy in queuing detection for each individual — in other words, *QueueVadis* was able to correctly classify all queuing episodes in our user study. Additionally, when we ran *QueueVadis* 's queue detection analysis for approximately 5 hours of Micro Activity accelerometer data traces (not including queuing activity, collected for other research purpose in our university), the false positive rate queuing activity detected by *QueueVadis* was 16.6%.

Given that *QueueVadis* can detect the queuing activities correctly, we next ask: how accurate is *QueueVadis* in detecting both the start time (T_s) and the end time (T_e) of each queuing episode? To answer this question, we computed the time difference between the *true* time & the *QueueVadis' estimates* (using multiple concurrent tier-2 classifiers) of T_s and T_e for each individual queuing episode. We also calculated the overall percentage-error in estimating the total queuing duration (i.e., $T_w = T_e - Ts$).

Table 6 provides the *median* and *standard deviation* of these errors (across all the queuing episodes), separately for the F&B and Movie Ticketing venues. We see that:

Error	F&B Venues		Ticketing Venues	
	Median	Std. Dev.	Median	Std. Dev.
T_s (secs)	9.2	47.1	7.7	100.9
T_e (secs)	4.0	92.1	1.5	2.8
T_w (%)	4.8	30.2	4.1	27.6

Table 6: Estimation Err. (Start, End) and Tot. Queuing Times

- The estimation errors for start and end times are typically very low–less than or equal to 10 secs, indicating that the *CD* and *DT* logic in the *QueueVadis* client is quite successful in detecting important queuing-related events. However, the errors in estimating T_s are larger than those for estimating T_e. This is due to the fact that *QueueVadis' QT* component provides early detection of the end of a queuing episode, by using 3 consecutive frames (=9 secs) of walking, thereby bounding this detection latency. On the other hand, T_s requires detection of 2 consecutive windows of activity, and can be incorrect by, on average, a window size of $\frac{W_Q}{2}$ frames.
- The overall estimation error for the total queuing duration is also low–less than 10%. This suggests that the *QueueVadis* client is pretty effective in estimating the real queuing delay experienced by a user. In particular, given that the total wait time at the F&B venue was approx. 4 minutes (240 secs) and 10 minutes at Theaters, a 10% estimation error would translate to errors of less than minute at *F&B* and 2 minutes at Theater. We believe that this level of accuracy should be acceptable for most people in the *Where To Go?* & *Waiting Worth It* scenarios.

It is important to point out that the estimation errors were much higher (by a factor of at least 5-6) when we experimented with variants of the *QueueVadis* client that did not have multiple concurrent tier-2 classifiers, but instead either used only one classifier or at most one classifier per queue *type*.

6.2.3 Energy Consumption

To quantify the potential energy overheads of the *QueueVadis* client, we measured its average power consumption (measured over a test duration of 10 minutes using the Monsoon power monitor[11] and repeated twice) on a Galaxy SIII phone in 5 typical scenarios– each with a varying number of concurrent tier-2 (HA) classifiers. From Figure 8, we see that *QueueVadis* consumes 29.245 mW when we only enable its accelerometer sensors, and the energy number slightly increases to 31.85 mW when both HA and MA classifiers are running. When *QueueVadis* runs 16 HA classifiers concurrently, it consumes 34.005 mW, indicating that our use of multiple concurrent classifiers imposes insignificant energy overheads.

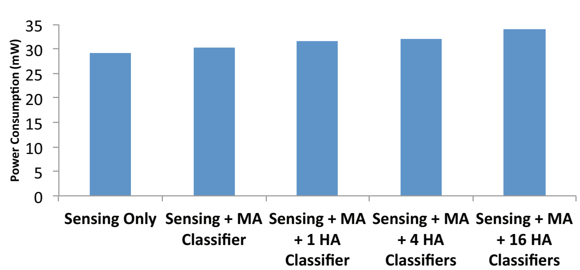


Figure 8: *QueueVadis* Avg. Power Consumption

Moreover, we also studied the usefulness of relying on coarse location triggers to selectively activate *QueueVadis*. We conducted a user study with 8 participants (5 in Tokyo and 3 in Singapore), where each participant carried a *QueueVadis* client-enabled device for 8 hours during the daytime, and also logged their high-level location as they went about their daily activities. Assuming that *QueueVadis* would be activated whenever the participant approached a potential queuing location (e.g., food service counters, coffee shops or automated teller machines), our real-world traces showed the largest activation duration (among the 8 participants) to be 25 minutes, over the 8 hours. Even assuming a *QueueVadis* client with 16 concurrent HA classifiers, the corresponding energy consumption is approximately 14.2 milliWatt-hours (mWh), or less than 0.5% of Galaxy S3's battery capacity (7.98Wh).

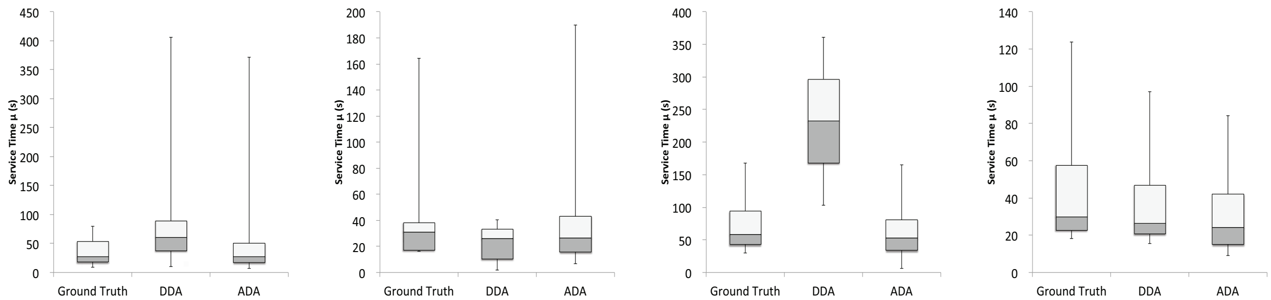
7. AGGREGATE PROPERTIES OF A QUEUE

We now focus on the *QueueVadis* server's ability to compute the aggregate properties of a specific queue, based on reports provided by corresponding *QueueVadis* clients. More specifically, we are interested in two key properties of a queue of direct relevance to our *Where To Go?* and *Waiting Worth It* scenarios: *(a)* the total wait time T_w that a person is likely to experience and *(b)* the service time $\frac{1}{\mu}$ experienced by the customer at the service counter.

7.1 Estimating Service Times

Given our empirical evidence of the variability of service times, even in a single queue, the *QueueVadis* server focuses on computing various statistical properties (such as the mean and variance) of the random variable $\frac{1}{\mu}$, as opposed to a single estimate of the service time. We have investigated two different algorithms for computing the distribution of the service times of a particular queue:

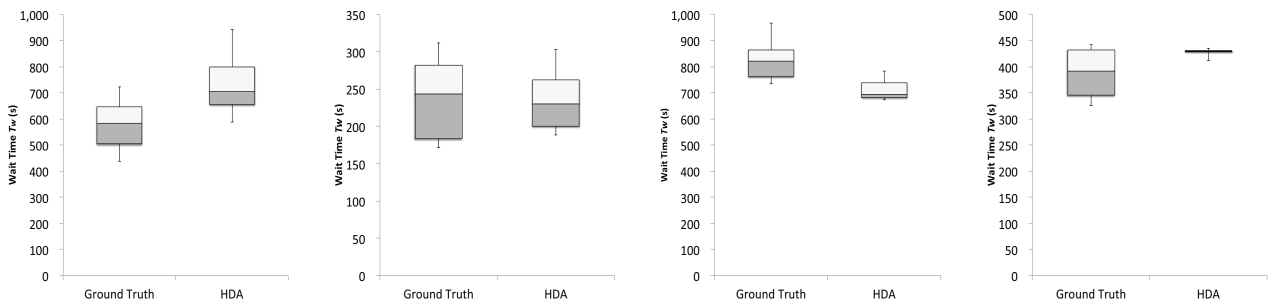
- *Departure-driven Detection* Algorithm (**DDA**): Here, we derive the service times based only on the end time (T_e) of each individual's queuing episode. In particular, when the *QT* component of the *QueueVadis* client detects the end of an individual's queuing episode, it sends the T_e estimate to the *QueueVadis* server. If the server now receives these values from two successive individuals (denoted by customers i and $i+1$), then the service time experienced by the $i+1^{th}$ customer can be estimated as $T_e(i+1) - T_e(i)$. This approach, however: *a)* works only if each successive customer in the queue has the *QueueVadis* client; *b)* assumes that the queue is never empty, i.e., the service for customer $i+1$ starts immediately after the departure of customer i.
- *Activity-centric Detection* Algorithm (**ADA**): This is a more sophisticated technique, based on the assumption that a queuing individual, say customer i, will typically remain stationary in the queue (while the person at the counter is being served), and will move forward only when the customer being served leaves, and the person currently at the head of the queue moves to the service counter. Accordingly, if t_1 and t_2 denote the successive times (frames) where the individual exhibits a new *stepping* movement (and is thus stationary for the entire duration (t_1, t_2)), then, from the perspective of this individual, the service time can be approximated as $t_2 - t_1$. In our proposed algorithm, each *QueueVadis* client periodically transmits its set of $t_2 - t_1$ values to the *QueueVadis* server. On receiving and aggregating such sets of values from multiple clients, the server can then estimate the relevant statistics for $\frac{1}{\mu}$. **This approach has the advantage of being practical as it is applicable even when only a subset of the queuing customers have the *QueueVadis* client.** On the other hand, as this approach may be sensitive to 'noise' in the micro-activity (movement) pattern (e.g., a queuing customer can often not move forward each time the queue advances, but



(a) F&B Peak (Singapore) (b) F&B Off-Peak (Singapore) (c) F&B Peak (Tokyo) (d) Ticketing (Tokyo)

Each bar represents 0th, 25th, 50th, 75th, and 100th percentiles in the distribution from the bottom to the top.

Figure 9: Box Plots of Service Times ($\frac{1}{\mu}$) at 4 Locations: Ground Truth vs. Estimates



(a) F&B Peak (Singapore) (b) F&B Off-Peak (Singapore) (c) F&B Peak (Tokyo) (d) Ticketing (Tokyo)

Each bar represents 0th, 25th, 50th, 75th, and 100th percentiles in the distribution from the bottom to the top.

Figure 10: Box Plots of Wait Times (T_w) at 4 Locations: Ground Truth vs. Estimates

take multiple steps after several customers have been served), *ADA* eliminates *outliers*, by discarding the bottom and top 5 percentile readings.

7.1.1 Evaluation Results

Figure 9 plots the boxplots of the service times computed in 3 different ways: the video-annotated ground truth and the two algorithms presented in Section 7.1, for the four location/times discussed before. We see that:

- The DDA algorithm is not very robust, as it often overestimates (and also underestimates) the service time. This error arises because consumers, in real life, often do not walk away immediately after receiving service, but instead wait for their friends to finish purchasing before walking away together. This leads to an overestimate of their service time (and an underestimate of the next person's service time).
- The ADA algorithm, on the other hand, proves to be remarkably robust in estimating the distribution of service times, and in fact, typically provides mean estimates within 5-10 seconds of the ground truth.

Robustness to Low Participation Rates: ADA is particularly attractive as it does not require all queuing participants to have the *QueueVadis* system. We studied the statistics of the estimated mean of the service times computed by ADA vs. the ground truth, when only a fraction of the queuing individuals were assumed to have *QueueVadis* clients. To perform this study, we computed the mean of the ADA estimates over all possible combinations (corresponding to the specified fraction) of the queuing individuals, and also computed the variance of this mean estimate. Table 7 plots these values for an F&B outlet in Singapore, and also demonstrates that ADA can provide robust estimates in practical situations, where only a small fraction of the queuing individuals may be expected to have *QueueVadis*.

	Ground Truth	Fraction of the Individual with *QueueVadis*					
		100%	80%	60%	40%	20%	10%
Mean (sec)	25.01	21.81	21.54	22.02	22.17	21.24	22.70
Mean Stdev.	N/A	11.53	11.21	11.32	11.29	10.34	12.99

Table 7: Service Time ($\frac{1}{\mu}$) vs. fraction of *QueueVadis* users

7.2 Estimating the Total Wait Time

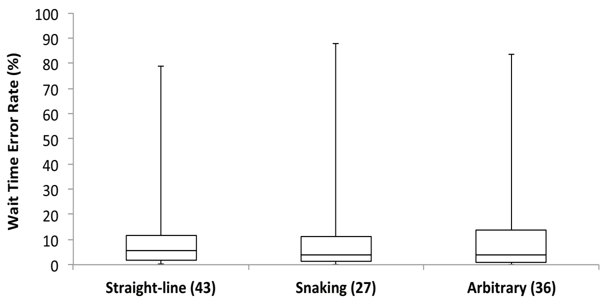
To estimate the total wait time, we devised the *History-Driven Estimation* algorithm (**HDA**). In HDA, the *QueueVadis* server receives the estimated total wait time $T_w(i)$ from the i^{th} customer and aggregates all these reports from multiple customers. It then computes a weighted "moving average" of these $T_w(i)$ values (giving greater weightage to recently departing customers) to predict the wait time likely to be experienced. HDA can possibly suffers from two drawbacks: *(i)* as the fraction of customers with *QueueVadis* clients decreases, the $T_w(i)$ reports get more sporadic, causing its accuracy to degrade; *(ii)* it implicitly assumes that the queue arrival rate (λ) remains constant. In particular, if there is a sudden surge in the number of people who've joined the queue, HDA will continue to underestimate the true queuing delay, until these people complete their transaction and leave the queue.

7.2.1 Evaluation Results

We now study the difference between the true and estimated total wait times (T_w), based on the HDA method proposed in Section 7.2. Figure 10 plots the boxplots of the true and estimated wait times (computed by the *HDA* technique) for the four venues/locations. We see that HDA's estimated wait times tally quite well with the real wait times in all 4 cases, with errors in the median values being around 10-15% in all cases. This indicates that our queue estimation technique can prove to be fairly useful to users for both the *Where To Go?* and *Waiting Worth It* scenarios.

7.3 Robustness to Various Queue Shapes

To quantitatively study how *QueueVadis* works across different real-world queue shapes, we classified our 106 queues into 3 classes: (i) *Straight-line*, where the users queued in a straight line (e.g., the Fruits and the Drinks queue in Figure 2); (ii) *Snaking*, where the queues had 180° turns where the user's movement got reversed (e.g., the Malay queue in Figure 2 and 2 movie theater ticketing queues) and (iii) *Arbitrary*, where the queues had a more free-form shape with one or more acute-angled turns (e.g., the Indian and the Western queue in Figure 2). Figure 11 plots the percentage error in the wait time estimates T_w (as box plots) for each queue class (with the number of distinct queues for each class). Since the median estimation errors are uniformly low (less than 10%) for all 3 classes, we posit that *QueueVadis* works *across arbitrary queue shapes*.



Each bar represents 0th, 25th, 50th, 75th, and 100th percentiles in the distribution from the bottom to the top.

Figure 11: Wait Time Errors (Different Queue Shapes)

7.4 Handling Premature Departures

We also studied *QueueVadis'* ability to handle situations where a user leaves a queue prematurely (even though Section 3.2.4 show this to be a rare event). 4 users left a queue prematurely (users 1, 2, 3 & 4 leaving when they had 1, 2, 3 & 4 people ahead of them in the queue, respectively); each user repeated this behavior on 5 different occasions. Separately, we manually recorded the true T_w value of the individual immediately in front of the user, as an appropriate estimate of the T_w that the user would have experienced if she had completed the service transaction after queuing. Table 8 shows that the mean T_w values estimated by *QueueVadis* for each prematurely departing user gets smaller, expectedly, smaller (compared to T_w of the person in front who completed queuing), the earlier a user departs prematurely from the queue.

More importantly, we computed the average T_w (using only ground truth observations who had completed queuing) to be 183.3 secs, while the average T_w of the prematurely departing users was much smaller (149.3 secs). However, after applying *QueueVadis'* outlier elimination logic to all T_w readings (consisting of both 'completed queuing' and 'prematurely departing' users), T_w was estimated to be 160.9 msec (i.e., $\sim$ 10% lower than the true value), even for our extreme scenario *where half of the queuing instances consisted of premature departures*: this demonstrates our robustness to the occasional case of an individual leaving a queue mid-stream.

	Number of people in front when leaving			
	1	2	3	4
Average (T_w)	186.6 (218.4)	102.3(152.0)	108.1(175.0)	80.6 (176.8)

Table 8: Wait Time: Premature (vs. Complete Queuing)

8. DISAMBIGUATION ENGINE

The final piece in the *QueueVadis* puzzle is the *Disambiguation Engine*, which detects if two customers are queuing in the same or different queues (so as to assign an individual's service or wait time estimates to the correct queue). The proposed disambiguation engine combines two orthogonal principles: (i) phase-shifted similarity in the movement patterns between people in the same queue, and (ii) similarity/differences of *directions* of movement trajectories of people in the same/different queues, respectively.

8.1 Cross-Correlation of MA streams

In this approach, we look at the (standing, movement) sequences of a pair of individuals as two time series and measure their *cross-correlation function*. Our intuition is that individuals in the same queue will exhibit, albeit ideally, *time-shifted* copies of the same underlying movement sequence (as everyone will move forward when the person at the head of the queue gets dequeued). More specifically, the cross-correlation component of the *QueueVadis* server's disambiguation engine uses the *MA-2* set of micro-activity labels, provided to it by participating *QueueVadis* clients, as our focus here is on purely looking for movement similarities (and not on identifying queuing, as that has already been performed by the *QueueVadis* client). In particular, given two sequences of such time series, $\mathcal{X}$ and $\mathcal{Y}$, the cross-correlation function is computed as:

$$c_{XY}(k) = \begin{cases} \sum_{t=1}^{N-k} X_t Y_{t+k} & k = 0, \cdots, N-1 \\ \sum_{t=1-k}^{N} X_t Y_{t+k} & k = -1, \cdots, -(N-1) \end{cases} \quad (1)$$

Figure 12 illustrates cross-correlation between two such pairs of consecutively queuing customers (one pair in the same queue, the other pair in adjacent but *distinct* queues) at SMU's food court–we can see that (due to the similar phase-shifted movement pattern in the same queue), the *maximum* correlation value is much higher (approx. 0.6) for the same-queue pair, as opposed to the different-queue pair (approx. 0.35) whose movements are less-synchronized.

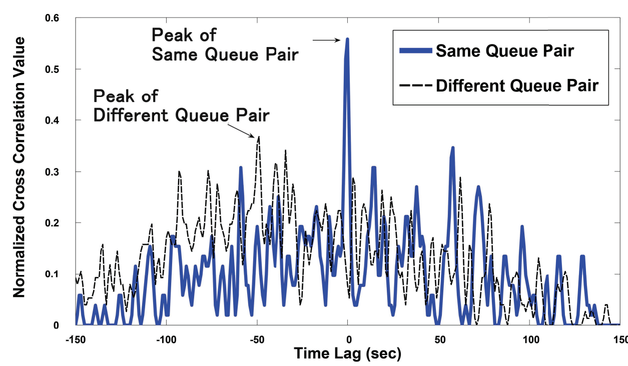


Figure 12: Cross-Correlation (Pair in Same vs. Diff. Queue)

Through extensive empirical studies over our datasets, we saw that a high and unimodal cross-correlation peak always exists for customers in the same queue, whereas relatively low and multi-modal peak is often observed for different queue customers. Accordingly, the *disambiguation engine* uses two cross-correlation features over C_{XY} to classify incoming pairs of queuing users: *a)* the largest cross-correlation value, denoted by $c_{XY}(\tau^*) = \max c_{XY}(.)$; and *b)* R_{XY}, the fractional difference between the first and second peak (denoted as $c_{XY}(\tau')$) in $c_{XY}(.)$, computed as: $\frac{c_{XY}(\tau) - c_{XY}(\tau')}{c_{XY}(\tau')}$.

Table 9 summarizes the classification accuracy achieved using a Naive Bayes classifier over MA-2 streams collected at 6 different venues in Singapore (4 food court F&B stalls, 1 movie theater ticketing counter, and 1 Starbucks) from 41 *real-world* queuing individuals. Results are reported using a 10-fold cross validation study over 114 pairs of MA-2 streams (61 pairs in the same queue and 53 in different queues). Note that all the "same queue" training data are collected from consecutive customers or only one additional customer between them. Subsequently, to study the effect of varying the participation rate, we conducted additional studies, collecting 3 sessions' data from 10 users queuing in a single queue. Table 10 plots the accuracy (*recall*) of such queue disambiguation, as a function of K, the number of intermediate users between a pair of queuing individuals. (Specifically, a "0" means that the two individuals queue consecutively, while "1" means that the individuals are separately by a single other user). We see that this precision decreases rapidly with K, falling below 50% when 3 or more people separate the pair, reflecting the lack of synchronized movement (in real world queues) among people who are separated by more than 2 individuals. Thus, in practice, the correlation classifier provides a very reliable *positive* indicator of co-queuing (for users who queue consecutively or are separated by 1-2 individuals), but cannot definitively indicate that users are in separate queues.

		Predicted Class		Accuracy
		Same Queue	Diff. Queue	(%)
Class	Same Queue	47	14	77.04
	Diff. Queue	3	50	93.34

Table 9: Confusion Matrix of the Naive Bayes Model

No. of Intermed. People (K)	0	1	2	3	4
Classification Accuracy	0.83	0.73	0.55	0.34	0.21

Table 10: Same-Queue Classification Accuracy vs. *K*

8.2 Direction of In-Queue Movement

To bolster the queue disambiguation accuracy, we investigate an alternative feature–the *trajectory, or sequence of directions, in which a user moves while queuing.* This method is motivated by Figure 2, which suggests that, in the real-world, different queues in the same space have distinct trajectories. In this approach, we thus utilize the smartphone-embedded *magnetic compass* sensor to observe the *directional component* of an individual user's trajectory during those frames that are classified (using the MA-3 classifier) as "step" (a probably reliable indicator of the direction of a user's 'shuffle forward' in-queue movement). Moreover, to accommodate the noise in the compass data, we quantize the readings into '45 deg' octants–an individual's sequence of movements is then represented as a sequence of values $\mathcal{D} = [d_1, d_2, \ldots,]$, where the i^{th} element indicates the directionality of the i^{th} stepping activity, with $d_i \in \{1, \ldots, 8\}, \forall i$. We further convert each sequence $\mathcal{D}$ into a vector $\mathcal{S} = [s_1, s_2, \ldots, s_8]$, where the 8 elements represent the fraction of each direction's movement count. Figure 13 illustrates this quantized representation of the trajectory of two queuing individuals.

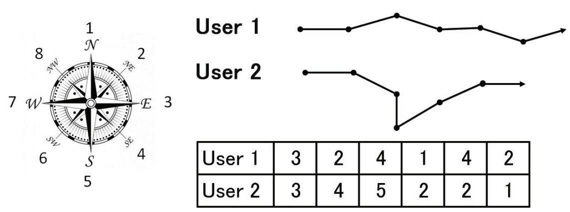


Figure 13: Movement-Orientation Based Similarity Measures

The similarity in the trajectory of a pair of individuals i and j is then expressed via the similarity between $\mathcal{S}(j)$ and $\mathcal{S}(k)$ (using the dot product of $\mathcal{S}(j)$ and $\mathcal{S}(k)$). Table 11 lists the same queue classification accuracy using the Direction of In-Queue Movement approach, computed by varying the percentage of *QueueVadis*-equipped users (average of all combinations). Table 12 summarizes the classification accuracy achieved by combining the Cross-Correlation approach and the Direction of In-Queue Movement approach. Note 1: the "same queue" training data is collected from the customers who have a variable number of intermediate people between them. Note 2: the disambiguation accuracies reported here use the trajectory of the entire queuing episode (which can last several minutes). Hence, this approach cannot directly address the "Waiting Worth It" scenario, where the disambiguation must be performed immediately after the queuing onset: this issue remains an open problem.

Frac. of People with *QueueVadis*	100%	80%	60%	40%	20%
Classification Accuracy	0.78	0.74	0.71	0.63	0.60

Table 11: Same-Queue Classification Accuracy (Compass-based) vs. Participation Rate

		Predicted Class		Accuracy
		Same Queue	Diff. Queue	(%)
Class	Same Queue	72	28	72
	Diff. Queue	17	42	73.7

Table 12: Confusion Matrix of the Two Approaches Combined

9. DISCUSSION

While *QueueVadis* performs reasonably, across a variety of queues (of different types, and with different shapes) even in dense urban spaces (e.g., foodcourts in malls or university campuses), there are several issues that *QueueVadis* can tackle in the future:

- We have not considered queues that are dynamically reconfigured–e.g., at airport security, the opening of a lane can make people in an existing queue walk briskly for up to 15 seconds. This brisk walking motion will confuse *QueueVadis'* activity detection mechanisms.
- We have also not evaluated *QueueVadis* in scenarios with *group or herding dynamics.* A bus stop offers an example of such a queue, where a tour group might board a bus together (and thus exit the queue almost simultaneously).
- In our experiments, we observed that individuals usually had their smartphone inside their pockets or held it in their hands. It is possible that users may use their smartphone more actively (e.g., Tweeting their friends) and habitually while queuing, thereby reducing the activity classification accuracy. However, such usage may also offer opportunities for finer-grained sensing–e.g., provide more accurate compass-based directional estimates when the phone is being used.

10. RELATED WORK

Mobile Sensing & Activity Recognition: Approaches such as CenceMe [10], Jigsaw [8] and Escort [6] have applied feature-based classification on mobile phone accelerometer data to classify everyday *locomotive* activities (such as sitting, standing and walking). Hierarchical activity models have been used, in both supervised and unsupervised learning based approaches, to recognize semantic *Activities of Daily Living* (ADLs), from underlying locomotive and gestural signatures-e.g., Huynh et. al [7] used LDA-based topic models to discover common recurring MA patterns for different ADLs. *QueueVadis'* two-tier classification model is borrowed from the SAMMPLE framework [16], which classified HAs in home and office environments.

Mobile Sensing-based Queue Detection: LineKing [4] was one of the first smartphone systems to detect human waiting behavior & wait times in specific places such as a coffee shop. It used wait times as a proxy for queuing delay (assuming that all people in the coffee shop are queuing by default). The more-recent QueueSense approach [13] adopts a participatory mechanism similar to ours: it identifies relationships among multiple queuing individuals, using accelerometer and compass sensor data. One of the major design differences is that QueueSense's individual queuing activity recognition relies on collaborative exchanges with neighboring nodes via Bluetooth, whereas a *QueueVadis* client simply relies on its own sensor data. Like *QueueVadis*, QueueSense also modified the classification interval for different queue types. In general, *QueueVadis'* evaluation tackled several additional real-world artifacts, such as low participation rates, multiple close-range queues, highly variable queuing delays and premature user departures. More recently, Wang et. al [15] used the signal strength evolution of a Wi-Fi AP to deduce the in-queue movement behavior of an individual, and thus infer individual queuing delays. This infrastructure-based approach did not, however, investigate the challenge of multiple distinct queues in close proximity.
Other Queue Estimation Systems: Video analytics has been used to infer queue lengths and wait times (e.g., [5, 14]) at fixed, well-known locations (such as a stadium entrance). However, these systems are hard to deploy ubiquitously.

11. CONCLUSION

In this paper, we presented *QueueVadis*, a two-tier energy-efficient queue detection system that can provide both the aggregate-level and the individual-level queue properties (even when multiple queues are too close to be distinguished purely by location) using just cellphone sensor data. We implemented and tested *QueueVadis* with different types of real-world queues; our results show that *QueueVadis* provides practically useful estimates of the current wait and service times with a relatively small median error. In particular, for F&B venues, we found that the estimation error for total wait times is 1 minute (making a scenario such as "Where To Go?" feasible); moreover, the start time of a queuing episode is estimated with a median error less than 10 seconds (making services such as "Waiting Worth It" feasible, when the users are already associated unambiguously to a single queue).

12. ACKNOWLEDGMENT

We thank the anonymous reviewers and especially our shepherd, Marco Zuniga, for their help in improving the paper's presentation and technical content. In addition, we especially thank Vigneshwaran Subbaraju for his data collection and activity recognition software, Rahul Majethia for helping to collect and analyze preliminary queuing data, Takuya Takimoto for helping to conduct user studies in Japan, and Jin Nakazawa and Hideyuki Tokuda for their advice in refining the content of the paper. Finally, the authors gratefully acknowledge the support of the students and staff of LiveLabs and Hide Tokuda lab for helping to collect large amounts of queuing data. This work was supported partially by Singapore Ministry of Education Academic Research Fund Tier 2 under research grant MOE2011-T2-1001, partially by the National Research Foundation, Prime Minister's Office, Singapore, under the IDM Futures Funding Initiative , and partially by the Ministry of Education, Culture, Sports, Science and Technology (MEXT, Japan) Grant-in-Aid for the "Research and Development for Big Data Use and Application" and for the "Program for Leading Graduate Schools".

13. REFERENCES

[1] 11 longest queue restaurants in singapore (with no reservation). `http://sethlui.com/longest-queue-restaurants-singapore/`.

[2] 500 mminute waiting time at toy story mania in tokyo disney sea. `http://nlab.itmedia.co.jp/nl/articles/1207/16/news009.html`.

[3] Balan, R. K., Misra, A., and Lee, Y. LiveLabs: Building an in-situ real-time mobile experimentation testbed. *Proc. of HotMobile*, Santa Barbara, CA, 2014.

[4] Bulut, F., Yilmaz, Y., Demirbas, M., Ferhatosmanoglu, N., and Ferhatosmanoglu, H. Lineking: Crowdsourced line wait-time estimation using smartphones. *Proc. of MobiCASE*, Seattle, WA, Oct. 2012.

[5] Chan, A., Liang, Z.-S., and Vasconcelos, N. Privacy preserving crowd monitoring: Counting people without people models or tracking. *Proc. of IEEE CVPR*, Anchorage, AL, June 2008.

[6] Constandache, I., Bao, X., Azizyan, M., and Roy Choudhury, R. Did you see bob? using mobile phones to locate people. *Proc. of MobiCom*, Chicago, IL, Sept. 2010.

[7] Huynh, T., Fritz, M., and Schiele, B. Discovery of activity patterns using topic models. *Proc. of UbiComp*, Seoul, Korea, 2008.

[8] Lu, H., Yang, J., Liu, Z., Lane, N., Choudhury, T., and Campbell, A. The jigsaw continuous sensing engine for mobile phone applications. *Proc. of SenSys*, Zurich, Switzerland, Oct. 2010.

[9] Machine Learning Group at the University of Waikato. *Weka 3: Data Mining Software in Java*. `http://www.cs.waikato.ac.nz/ml/weka/`.

[10] Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., Zheng, X., and Campbell, A. T. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. *Proc. of SenSys*, 2008.

[11] Monsoon Solutions Inc. *Monsoon Power Moniter*. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[12] Parameswaran, V., Shet, V., and Ramesh, V. Design and validation of a system for people queue statistics estimation. *Video Analytics for Business Intelligence*, pages 355–373. Springer, 2012.

[13] Qiang, L., Han, Q., Cheng, X., and Sun, L. Queuesense: Collaborative recognition of queuing on mobile phones. *2014 IEEE International Conference on Sensing, Communications and Networking (SECON)*, 2014.

[14] Segen, J. and Pingali, G. A camera-based system for tracking people in real time. *Proc. of ICPR*, Vienna, Austria, 1996.

[15] Wang, Y., Yang, J., Chen, Y., Liu, H., Gruteser, M., and Martin, R. P. Tracking human queues using single-point signal monitoring. *Proc. of MobiSys*, Bretton Woods, NH, June 2014.

[16] Yan, Z., Chakraborty, D., Misra, A., Jeung, H., and Aberer, K. Sammple: Detecting semantic indoor activities in practical settings using locomotive signatures. *Proc. of ISWC*, Zurich, Switzerland, June 2012.

[17] Zhang, Z., Venetianer, P. L., Lipton, A. J., et al. A robust human detection and tracking system using a human-model-based camera calibration. *The Eighth International Workshop on Visual Surveillance*, 2008.