Alien vs. Mobile User Game: Fast and Efficient Area Coverage in Crowdsensing

Manoop Talasila, Reza Curtmola, and Cristian Borcea
Computer Science Department
New Jersey Institute of Technology
Newark, NJ, USA
Email: mt57@njit.edu, crix@njit.edu, borcea@njit.edu

***Abstract*—Mobile crowdsensing enables real-time sensing of the physical world. However, providing uniform sensing coverage of an entire area may prove difficult. It is possible to collect a disproportionate amount of data from very popular regions in the area, while the unpopular regions remain uncovered. To address this problem, we propose a model for collecting crowdsensing data based on incentivizing smart phone users to play sensing games, which provide in-game incentives to convince participants to cover all the regions of a target area. We designed and implemented a first person shooter sensing game, "Alien vs. Mobile User", which employs techniques to attract users to unpopular regions. Our prototype Android game collects WiFi data to create a campus coverage map. The results from a user study show that mobile gaming ensures high coverage, and we observe that the proposed game design succeeds in achieving good player engagement. Furthermore, we compare three strategies for area coverage in terms of coverage time and coverage effort for users. The simulation results demonstrate that Progressive Movement is the best strategy because it manages to quickly entice users from popular regions to unpopular ones with a reasonable coverage effort.**

I. Introduction

Mobile sensors such as smart phones and vehicular systems represent a new type of geographically distributed sensing infrastructure that enables mobile people-centric sensing [1]–[3]. This new type of sensing can be a scalable and cost-effective alternative to deploying static wireless sensor networks for dense sensing coverage across large areas. mCrowd [4], Medusa [5], and our McSense platform [6]–[8] are some of the recent proposals for a common mobile crowdsensing platform to perform various sensing tasks supported by the smart phone sensors.

Mobile crowdsensing can be used to enable a broad spectrum of applications, ranging from monitoring pollution or traffic in cities to epidemic disease monitoring or reporting from disaster situations. Clients of this new sensing infrastructure include the population at large, researchers in many fields of science and engineering, as well as local, state and federal agencies.

A major challenge for broader adoption of the mobile crowdsensing systems is how to incentivize people to collect and share sensor data from a targeted area. Mapping an area with sensor data is a tedious effort when performed manually, and it is possible to collect a disproportionate amount of data from regions that are very popular compared to regions that are less popular [9]. Furthermore, sharing the sensed data may raise privacy concerns as it may require the sharing of private information such as location. Therefore, there is a need to find an efficient solution to incentivize the participants in collecting sensor data from an entire target area.

Many of the proposed mobile crowdsensing systems provide monetary incentives to smart phone users to collect sensing data. There are solutions based on micro-payments [10] in which small tasks are matched with small payments. The participants in mobile crowdsensing systems may require significant incentives to go out of their way and cover unpopular regions. Other techniques were also explored to motivate individuals to participate in sensing. For example, beneficial personal analytics are provided as incentives to participants through sharing bicycle ride details in Biketastic [11]. Another variety of incentive is enabling data bartering to obtain additional information, such as bargain hunting through price queries in LiveCompare [12].

In addition, there are gamification techniques proposed for crowd-sourced applications [13], [14]. However, to the best of our knowledge, no work has been done on using mobile games for incentivizing the participants in a mobile crowdsensing system to cover a targeted area uniformly. General gamification techniques for crowd-sourced applications cannot be directly applied in the context of uniform area coverage because we need to answer specific questions such as: What coverage strategies work best? What incentives mechanisms work best?

We propose to leverage gamification for fast and efficient area coverage in crowdsensing. Mobile sensing games use in-game incentives to convince participants to cover all the regions of a target area.

To demonstrate crowdsensing enabled by mobile gaming, we designed and implemented a first person shooter sensing game, "Alien vs. Mobile User", which can be played by mobile crowdsensing participants on their smart phones. The game involves tracking the location of extraterrestrial aliens on the campus map of our institution and destroying them. The game entices users to unpopular regions through a combination of alien-finding hints and higher number of points received for destroying aliens in these regions. The game was implemented in Android, and it collects WiFi signal data to construct the WiFi coverage map of the targeted area. The game also

leverages this WiFi data to provide indoor player localization; thus, players will be able to increase their scores by discovering aliens indoors. In addition, we performed extensive simulations to select the best gaming strategy for the prototype to ensure faster area coverage with minimum user effort.

Specifically, the paper makes the following contributions:

- To the best of our knowledge, we are the first to propose a model for automatically and uniformly collecting crowdsensing data across large areas based on incentivizing smart phone users to play mobile sensing games.
- We design and prototype "Alien vs. Mobile User", an Android-based mobile game that enables crowdsensing.
- We evaluate three strategies to attract users to unpopular regions in order to cover the entire target area, Localized, Random and Progressive, to understand which one leads to more efficient area coverage. Simulation results show the Progressive Movement strategy results in the lowest coverage latency. This strategy manages to quickly entice users from popular regions to unpopular ones in a natural way that results in a reasonable coverage effort for users.
- We demonstrate experimentally through a user study based on our game that mobile gaming can be a successful approach for efficient area coverage in crowdsensing. The results show that mobile gaming ensures high area coverage. Furthermore, the proposed game design succeeds in finding a good balance between attempting to attract users to the uncovered regions quickly and maintaining player interest in the game.

The rest of the paper is organized as follows. Section II describes the game design and implementation. Section III presents the alien movement strategies to cover the area. The simulation results for these strategies are discussed in Section IV. Section V presents the results of our user study. We discuss related work in Section VI. The paper concludes in Section VII.

II. Alien vs. Mobile User Game

The game is a first person shooter game played by mobile users on their smart phones while moving in the physical environment. Since the goal of the game is to densely cover a large area with sensing data (i.e., cover all regions of the area), it is essential to link the game story to the physical environment. Broadly speaking, in our game, the players must find aliens throughout an area (e.g., our campus) and destroy them. In the process, players collect sensing data as they move through the area. Although the game could collect any type of sensing data available on the phones, our implementation collects WiFi data (*BSSID, SSID, Frequency, Signal strength*) to build a WiFi coverage map of the targeted area. The motivation to play the sensing game is twofold: 1) The game provides an exciting *real-world gaming experience* to the players, and 2) The players can *learn useful information about the environment* such as the WiFi coverage map which lists the locations having best WiFi signal strength near the player's location.

The game contains the following entities/characters:

- **CGS**: The Central Game Server (CGS) controls the sensing game environment on the mobile devices and maintains the players' profiles and the sensing data collected from the sensing game.
- **Player**: Users who play the sensing game on their mobile devices. The players are rewarded with points for shooting and/or destroying the aliens. All players can see the current overall player ranking on a shared leaderboard.
- **Alien**: A negative role character that needs to be found and destroyed by the game players. Aliens are controlled by CGS according to the sensing coverage strategy.

A. Design

Game story: All the aliens in the game are hiding at different locations across the targeted area. Players can see the aliens on their screens only when they are close to the alien positions. This is done in order to encourage the players to walk around the area to discover aliens; in the process, we collect sensing data. At the same time, this makes the game more unpredictable and potentially interesting. The game on the phones periodically scans for nearby aliens and alerts the players when aliens are detected; only one detected alien is shown to the player at a given time. The player locates the alien on the game screen and starts shooting at it using the game buttons. When an alien gets hit, there are two possible outcomes: if this is the first or second time the alien is shot, the alien escapes to a new location to hide from the player. To completely destroy the alien, the player has to find it and shoot it three times. Hints of the alien's new location are provided after it is shot and escapes to another place. In this way, the players are provided with an incentive to cover more locations. The aliens hiding in the targeted area are common to all the players. Thus, different players can detect and shoot the same alien at different times as long as the alien was not destroyed.

The sensing side of the game: Sensing data is collected periodically when the game is on. Since this collects location traces, the players will be made aware of the potential privacy risk when they install the game. Nevertheless, the goal is to build games that are attractive enough to convince players to trade-off privacy for fun. The placement of aliens on the map will ultimately ensure full sensing coverage of the area. The challenge, thus, is how to place/move the aliens to ensure fast and uniform coverage while maintaining a high player interest in the game. The CGS moves aliens on the game map to desired locations (i.e., uncovered regions) using one of the alien movement algorithms proposed in Section III.

In the initial phases of sensing, CGS moves the alien to a location which is not yet covered, but later on it moves the alien intelligently from one location to another by considering a variety of factors (e.g., less visited regions, regions close to pedestrian routes, or regions where the client who needs the data requires high sensing accuracy). CGS helps the player who shoots an alien by providing game hints such as revealing the direction in which the alien has escaped. Generally, the alien will escape to farther away regions and the player might be reluctant to follow despite the hints provided by CGS. To

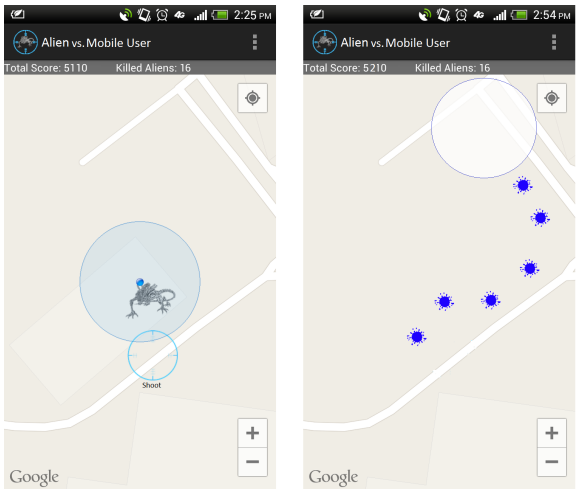

Fig. 1. Alien vs. Mobile User sample screenshots: finding the alien (left); alien trail (right).

increase the chances that players follow the alien, we provide more points for shooting the alien for a second time and even more for the third shot (which destroys the alien).

The more active players are, the more bullets they are able to collect in the game. The bullets available for collection are placed around the players and maintained individually for each player's view of the game based on three main factors: the number of bullets collected so far by the player, the total play time of the player, and the game level of the player. The number of bullets around each player increases linearly with these factors to keep the active players motivated to continue playing. But, placing too many bullets around the players could make the game less interesting. Therefore, CGS limits the bullet count to a certain threshold.

Indoor localization: This is necessary in order to cover the building floors. By default, Android uses WiFi triangulation in conjunction with GPS to estimate the location of the player in the (X, Y) plane while indoors. Due to potentially larger localization errors as compared to GPS, it is possible that the player and the alien she is shooting at are not in the same square, even though they appear to be in the same square in the game. The sensing reading is associated with the position of the alien (known to CGS), and thus it will reflect the localization error.

We leverage the barometric pressure sensor available in most Android phones to determine the Z coordinate, specifically the building floor. For accurate estimation of floor levels, we initially measured the altitude at ground level near each campus building. For the phones without barometric pressure sensor, we perform indoor localization based on the fingerprinting done by other players who visited the same area and whose phones have barometric pressure sensors. Specifically, these players have recorded the mappings between the estimated location and the strengths of the WiFi signals measured at that location. Thus, the players will be helped to hunt down aliens indoors by the sensing data they collect (i.e., WiFi signals).

B. Implementation

A prototype of the game has been implemented in Android and is compatible with smart phones having Android OS 2.2 or higher. When the player opens the game on her smart phone, the map of the player's current location is displayed on the game screen. An alien appears on the map when the player is close to the alien's location, as shown in Figure 1 (left). The player can target the alien and shoot it using the smart phone's touch screen. When the alien escapes to a new location, its "blood trail" leading to its new location is provided on the map as a hint to track it down (as shown in Figure 1 (right)). The server side of the game is implemented in Java using one of the Model View Controller frameworks involving EJBs/JPA models, JSP/HTML views, and servlets, and it is deployed on the Glassfish Application Server.

III. Alien Movement Strategies

In this section, we propose three strategies for placing and moving the aliens in order to cover the targeted area efficiently.

A. Assumptions and Definitions

The target area that needs to be covered is divided into small square cells (in our user study we consider the cell size 10m x 10m). Inside buildings, the target area includes both the ground floor as well as the upper floors. Each alien can be used to cover K squares before it is destroyed (in our game, aliens are destroyed after being shot 3 times, so $K = 3$). We assume there cannot be more than one alien in one square at any moment. Hence, the minimum number of aliens required to cover the area is $TotalNumberOfSquares/K$.

Clients/applications may need more than one reading per square due to reliability issues. Thus, the game has a parameter that indicates how many times each square must be covered by sensor readings.

In this section, we use the following terms:

- *Square*: The square is the smallest unit of coverage. Aliens and players move to squares.
- *Available Square*: A square for which the number of collected data points is lower than the desired number of data points.
- *Region*: A larger portion of the targeted area which contains K squares.
- *Popular Region*: A region where a substantial number of data points have been collected, but the number of data points required by the client has not been reached.
- *Unpopular Region*: A region where no data point has been collected.
- *Healthy Alien*: An alien that was never shot in the game.

B. Localized Movement Strategy

In this strategy, every alien is assigned to a single region, and it moves only in that region when shot by the player as shown in Figure 2 for alien A1. This strategy does not require the players to move much to hunt down the aliens, and thus, it could be successful.

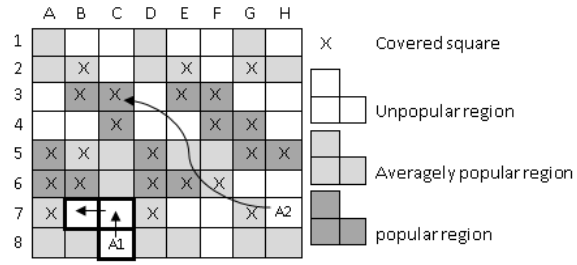

Fig. 2. Localized Movement strategy: Example of CGS moving the alien A1 locally in the assigned highlighted region when A1 is shot by the player. As part of the greedy process, CGS moves the healthy alien A2 from an unpopular region to a popular region.

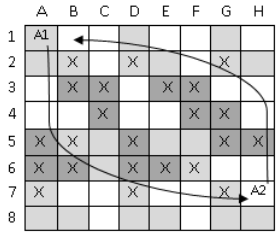

Fig. 3. Random Movement strategy: Example of CGS swapping alien A1 (which is shot) with the healthy alien A2 from an unpopular region.

CGS maintains statistics about popular and unpopular regions based on the data collected from players. Thus, CGS is able to adapt to the collected information and execute a greedy process which periodically moves the healthy aliens from unpopular regions to popular regions as shown in Figure 2 for alien A2. When the player shoots this healthy alien, CGS moves the alien back to its originally assigned region. A "blood trail" hint is provided to the player to indicate the location where the alien has escaped. This process places the aliens at highly popular regions to increase the probability of being found by the player and helps in luring the players to unpopular regions.

Issue: The main concern with this strategy is that the aliens are restricted to one region, and thus, the players may eventually predict aliens' locations. Therefore, the Localized Strategy may lead to a simple game plan and may become boring for players, ultimately leading to a decrease in the number of players.

C. Random Movement Strategy

In this strategy, the aliens are not restricted to a region and they can be placed in any square randomly by CGS. Thus, the strategy addresses the issue of easily predicting the alien's location. Basically, CGS swaps the alien which is shot in the game with a healthy alien from an unpopular region chosen randomly as shown in Figure 3. A1 is moved to a square in an unpopular region, chosen randomly from all the unpopular regions at that time. A2 is moved to an available square near to the location where A1 was shot. When there are no available unpopular regions to choose (i.e., regions with no coverage), CGS chooses the square from the regions with the minimum coverage.

The purpose of the alien swap is to take advantage of the popular regions from the starting phases of the game

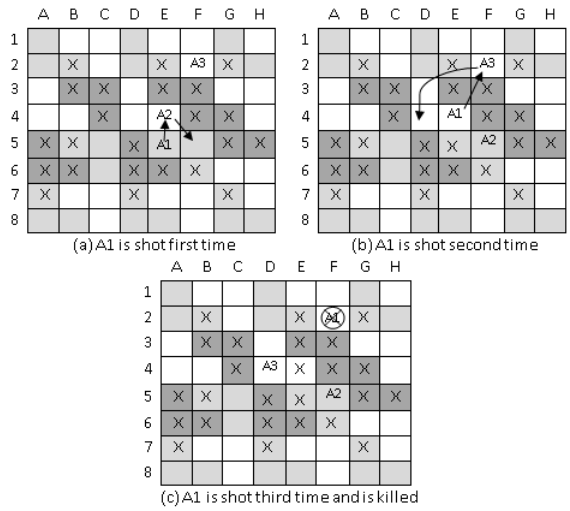

Fig. 4. Progressive Movement strategy: Example of CGS moving the alien to the closest unpopular region.

deployment, instead of waiting for a long period to adapt to the collected data (i.e., identify the top few popular regions) as performed in the Localized Movement strategy. Hence, the shot alien is moved from the popular region to an unpopular one to lure the player there. At the same time, we bring the alien from the unpopular region to an uncovered square of the popular region where there is a higher chance for players to encounter it. In the Random Movement strategy, new aliens are added at random time intervals.

Issues: Even though randomness helps CGS to place the aliens at unpredictable squares, CGS may end up placing the aliens too far from the player's current location. If this situation happens in the early phases of the game deployment, then players may lose interest in tracking the alien. The players might be more interested to track the aliens if the aliens are closer to their current region, though at a square that is not easy to predict. This should be done at least until CGS achieves a good game adoption rate.

D. Progressive Movement Strategy

In this strategy, the aliens are moved to the nearest unpopular regions when they are shot by the player. This addresses the issue of randomly placing the alien too far from the player's location. Basically, CGS swaps the alien which is shot in the game with the closest healthy alien from an unpopular region as shown in Figure 4. In the figure, when A1 is shot the first time, CGS chooses the closest available square from all the unpopular regions, which is A2's square. A2 is then moved to the next available square near A1's location as shown in Figure 4(a). A similar swap happens between A1 and A3 when A1 is shot again. A1 is finally destroyed when it is shot for a third time. Steadily, CGS covers the targeted area in a progressive fashion by starting from the popular regions as main centers and slowly expanding the coverage toward the unpopular regions.

CGS chooses a square from the closest unpopular region at that time. When there are no available unpopular regions,

Algorithm 1 Progressive Movement Strategy Pseudo-Code

Notation:

A1: The alien that needs to be moved to another location by CGS.
A2: The other alien in the game to whose location the A1 is moved by CGS.
currentSquare: The square where A1 is shot by the player.
chosenSquare: The square to which A1 is moved by CGS.
unPopularRegionsList: The list of unpopular regions.
mostUnpopularRegion: The region with highest number of uncovered squares.
getCurrentSquare(): Get the current square details of the alien.
getAlien(): Get the alien from the square if any alien is at that square.
getNearestAvailableSquare(currentSquare): Get the available square closest to the currentSquare.
updateCurrentSquare(chosenSquare): Update the currentSquare of the alien with the chosenSquare.
updateCoveredSquare(currentSquare): Set the coverage indicator for the currentSquare.
getNearestSquare(mostUnpopularRegion,currentSquare): Get the square details for the square in the most unpopular region which is closest to the currentSquare.
getRegionsWithMostAvailableSquares(sqCount): Get the list of unpopular regions having available squares of count sqCount.
sortByDistanceAscend(unPopularRegionsList, currentSquare): Sort the unPopularRegionsList by the distance between each region to the currentSquare in ascending order.
getRegionWithHealthiestAlien(): Get the region which contains the healthiest alien.

moveAlien(A1):

currentSquare = A1.getCurrentSquare()
chosenSquare = chooseSquareFromUnpopularRegions(currentSquare)
if $chosenSquare == NULL$ **then**
    The targeted area is fully covered and the game is complete
else
    A2 = chosenSquare.getAlien()
    A1.updateCurrentSquare(chosenSquare)
    if $A2 \neq NULL$ **then**
        A2.updateCurrentSquare(getNearestAvailableSquare(currentSquare))
    updateCoveredSquare(currentSquare)

chooseSquareFromUnpopularRegions(currentSquare):

mostUnpopularRegion=getNearestRegion(getMostUnpopularRegionsList(), currentSquare)
chosenSquare = getNearestSquare(mostUnpopularRegion,currentSquare)
return chosenSquare

getMostUnpopularRegionsList():

regionSize = 3 //constant
for $sqCount = regionSize$ to 1 **do**
    unPopularRegionsList=getRegionsWithMostAvailableSquares(sqCount)
    if $unPopularRegionsList \neq NULL$ **then**
        break
return unPopularRegionsList

getNearestRegion(unPopularRegionsList, currentSquare):

unPopularRegionsList=sortByDistanceAscend(unPopularRegionsList, currentSquare)
if multiple regions have the same distance **then**
    mostUnpopularRegion = getRegionWithHealthiestAlien()
else
    mostUnpopularRegion = unPopularRegionsList[0]
return mostUnpopularRegion

CGS chooses the square from the regions with the minimum coverage. When all the regions with same number of available squares are at same distance from the alien's location, CGS selects the square with the "healthiest" alien (i.e., never shot or shot the fewest times).

The purpose of this swap process is two-fold: 1) CGS pulls the players into unpopular regions to cover the targeted area efficiently, and 2) the game is made challenging because the players cannot predict the alien's moves without knowing the overall game details such as health status of all the aliens and the coverage status of all the squares. This process avoids the issues present in the other two strategies. Since our results show that this strategy works best, we present its detailed operation in Algorithm 1.

E. Number of Aliens in the Game

In our game, it is essential to set the number of aliens in such a way as to achieve a good balance between efficient coverage and maintaining player interest in the game. Empirically, our goal is to allow players to encounter aliens every so often, while not being able to predict the alien's movement. The number of aliens depends on the number of squares, the number of players, and the stage of the game (e.g., in early stages most regions are not covered, and in late stages most regions are covered).

For each strategy, CGS adds new aliens every few minutes near the players' locations depending on the game status to keep them interested in tracking the aliens. For each player, CGS allocates G_S aliens for every T_{GS} minutes time period, where $T_{GS} = 2 \times NumOfGameStages \times T$. $NumOfGameStages$ is a parameter that counts how many times each square must be covered. This is important for clients/applications that do not want to rely on only one reading per square due to reliability reasons. Thus, our game allows multiple readings for each square. However, the game works in stages: it first attempts to cover each square once, then to cover each square twice, and so on. T is the time period after which CGS attempts to add a new alien. This formula allows CGS to add more aliens in the final stages of the game to pull the players into the most unpopular regions.

CGS adds a new alien at the player's location only if the player satisfies either of the following conditions. First, the player has not found the alien in the last T minutes. Second, the player tracked and killed the alien in last T minutes. The intuition behind these conditions is that in both scenarios the player is interested in tracking the alien. CGS adds the new alien only if the player has moved from her last location in T minutes. In case the player has found the alien in the last T minutes and ignored it, CGS alerts the player about the last found alien instead of adding a new alien again after T minutes.

IV. Simulations

This section presents the evaluation of the alien movement strategies, which have a major impact on the efficiency of the area coverage. The three main goals of the evaluation are:

TABLE I
PLAYER TYPES

Player Type	Player Type Description
PT1	Active player who always tracks the alien.
PT2	Tracks the alien if it is in 50 meters range or if she can deliver the third shot that destroys the alien.
PT3	Tracks the alien only if it is in 50 meters range.
PT4	Player never tracks the alien; shoots the alien only when found in her travel path.

TABLE II
NUMBER OF PLAYERS FROM EACH PLAYER TYPE ASSIGNED TO THE THREE BEHAVIORS/SCENARIOS

	BH1	BH2	BH3
PT1	0	3	6
PT2	2	3	4
PT3	4	3	2
PT4	6	3	0

(1) Compare the coverage latency for the three strategies, (2) Measure the coverage effort as experienced by players, and (3) Compare the area coverage over time for the strategies and identify the best strategy to implement in the prototype. We use the NS2 network simulator for the experiments [15].

A. Entropy of the area coverage

The goal of the game is to cover the targeted area as fast as possible and with the least possible effort from the players. In order to measure the player effort, we propose to look at the "system utilization": a perfect utilization requires that each square be covered uniformly with the minimum number of sensing readings required by the client who will use the data. In this case, the entire effort of the players is utilized. Any additional readings will lead to wasted player effort. Therefore, we need a metric to capture this idea.

The entropy of the area coverage presented in Equation 1 can be used to observe the overall system utilization. The greater the entropy value, the more uniformly the area is covered, and thus the player effort is not wasted.

$$H = -\sum_{i=1}^{N} \frac{v_i}{V} \log(\frac{v_i}{V}) \quad (1)$$

Where v_i is the number of visits by the players in square i; V is the total number of visits by the players of all visited squares ($V = \sum v_i$); and N is the total number of squares in the targeted area. The equation is similar to the deployment entropy calculated in [16], where a swarm of micro air vehicles are deployed to maximize the area coverage.

In the game context, the entropy is the measure of uniformity with which the players visit the squares in the targeted area. Once the game completes and the area is fully covered, the entropy of the area coverage quantifies the completeness and uniformity with which the players covered the area.

B. Simulation Setup

The simulation is set up in a 500m X 500m area with 12 players moving at 2m/sec. The simulation area is divided into 2,500 squares of size 10m x 10m. We used the following settings for the parameters defined in Section III: $NumOfGameStages$=5 (each square must be covered at least 5 times), T=3, and G_S={5..9} for game stages 1 to 5, respectively. Sensing is performed once per second by each player. The players get 100, 200 and 300 points, respectively, for the three shots needed to destroy the alien.

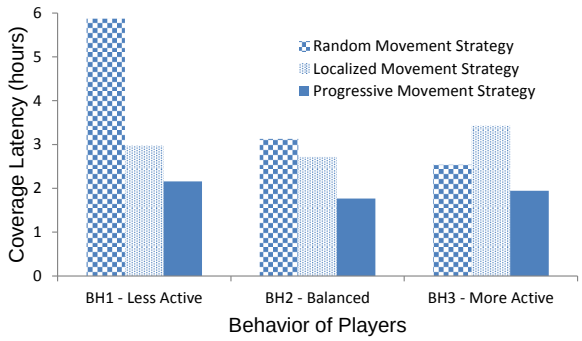

Fig. 5. Area coverage latency of the three alien movement strategies.

Although player behavior is very difficult to predict, we define four player types as shown in Table I. These player types are allocated to three simulation scenarios as defined in Table II: Behavior 1 (BH1) in which less active players are in majority, 2) Behavior 2 (BH2) in which the player types are balanced, and 3) Behavior 3 (BH3) in which highly active players are in majority.

C. Simulation Results

Area Coverage Latency. Figure 5 shows the coverage latency of the three strategies for the three simulation scenarios (BH1, BH2, BH3). The best coverage latency is achieved by the Progressive Movement strategy in all scenarios. This is one good reason to pick this strategy in future deployments of the game. In addition, this strategy also has techniques to keep the players interested by making the alien movement unpredictable.

The Random Movement strategy performs the worst in BH1 and BH2. In BH1, there are no active players to play the game which is the reason it takes more time to cover the targeted area. But with the increase in the number of active players, this strategy improves until it overcomes the Localized Movement strategy in BH3. In the Progressive and Localized strategies, good results are achieved even for BH1 because these strategies move the alien closer to the player who shot it and the PT2 type players track the alien in 50 meters range.

In BH3 scenario, surprisingly, the Localized and Progressive strategies take longer to complete than in the other two scenarios. The reason is that the active players are following similar paths to the unpopular regions. Thus, the rate of new squares covered is slower in BH3 compared to BH2.

Area Coverage Entropy. Figure 6 shows the entropy values calculated at the end of the simulation for the three strategies. As expected, the Random strategy performs best as it achieves the most uniform coverage. In the Localized and Progressive strategies, the increase in active players leads to a decrease in entropy as the active players cover the squares faster than in the Random strategy (in BH2 and BH3). Thus, the overall coverage is not uniform which results in lower entropy. When

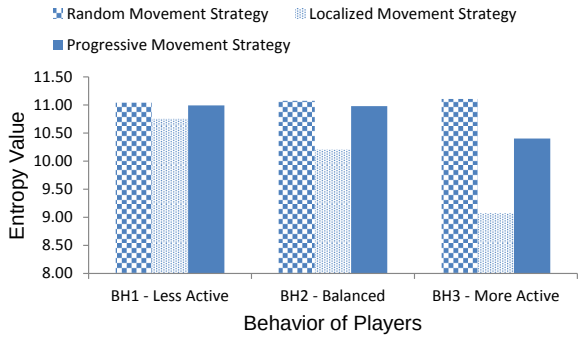

Fig. 6. Entropy of the area coverage for the three alien movement strategies.

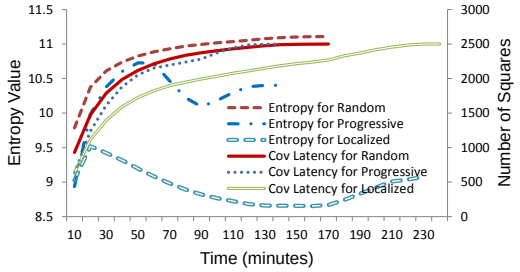

Fig. 7. The entropy and latency over time for the three alien movement strategies in BH3 scenario.

the number of active players is low, the entropy is similar for the three strategies.

When comparing the Progressive and the Random strategies for BH3, we see that it is difficult to conclude which one is better: Progressive leads to lower latency, while Random leads to higher entropy. However, Progressive is clearly better for the other two scenarios as it achieves lower latency and comparable entropy.

Area Coverage over Time. Figure 7 shows the entropy and latency over time for the three strategies in the BH3 scenario, which we observed to be the most complex in terms of selecting the best strategy. The results show that the goal of covering each square at least $NumOfGameStages$ times (set to 5 in these simulations) is achieved in less time by the Progressive strategy compared to the other two strategies.

Statistical Analysis. To understand the statistical significance of the performance difference between the three proposed alien movement strategies, we conducted the ANOVA test over the coverage latency (20 simulation runs for all three behaviours). We used the non-parametric Jonckheere-Terpstra method to test for ordered differences among the three strategies and found that the Progressive strategy has the lowest latency with $p < 0.0056$ for the three behaviours. Based on this result as well as the coverage entropy result, we conclude that the Progressive strategy is the best overall and use it in our prototype implementation.

V. User Study Evaluation

We ran a user study (performed with our institution's IRB approval) for 35 days, in which students used their Android devices to play our game [17] and collect WiFi data both outdoors and indoors throughout the campus of our institution. The campus area is divided into small squares of size 10m X 10m. The game is available in the Google Play Store. A total

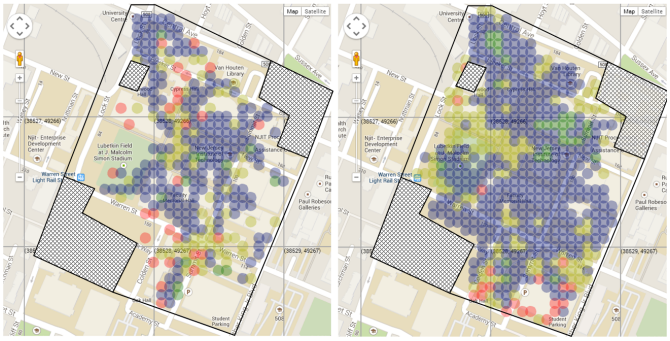

Fig. 8. Coverage of the University campus in the first four weeks of the studies: 46% crowdsensing with micro-payments (left) vs. 87% crowdsensing with mobile gaming (right). A number of areas have been removed from the map as they are not accessible to students.

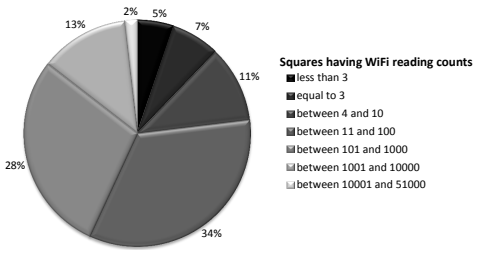

Fig. 9. Distribution of square coverage using the mobile game.

of 53 players registered, then installed and continued playing the game.

The main goals of this user study are: (1) quantify the performance of crowdsensing enabled by mobile gaming in terms of area coverage efficiency, and (2) analyze the effectiveness of mobile gaming as an incentive for crowdsensing.

Area coverage efficiency. Using the WiFi sensing data from our previous work McSense [6]–[8], a micro-payment based crowdsensing system, we present a comparison of area coverage between crowdsensing enabled by mobile gaming and crowdsensing enabled by micro-payments. Figure 8 overlays the collected WiFi data over our campus map for both methods. The WiFi signal strength data is plotted with the following color coding: green for areas with strong signal; blue for areas with medium signal; yellow for areas with low signal; and red for areas with no Campus WiFi signal. Overall, the mobile gaming approach doubles the area coverage compared to the micro-payment approach (87% vs. 46% of the campus). The complete details of the WiFi area coverage maps for both studies are available on-line under the game website [18].

To understand the effort put by players into achieving this high coverage, we compute the entropy of the area coverage. The entropy is calculated at the end of the user study using the counts of sensor readings for each of the covered squares in the area (cf. Equation 1). The calculated value for our study is 6.3, while the optimal value is 9.2 (i.e., when every square in the area is uniformly covered with the same number of sensor readings). This result shows that the players' effort is substantial, and future work should focus on additional game strategies to reduce this effort. To understand better the potential for improvement, Figure 9 shows the distribution of square coverage across the entire area. We consider only the squares that have been covered. In the user study, we required

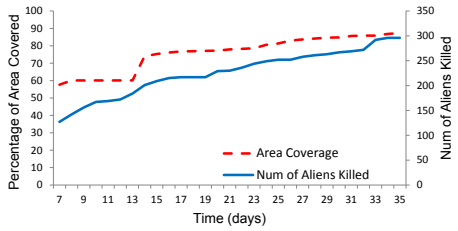

Fig. 10. Correlation between number of aliens killed and area coverage (we started to record this data in day 7).

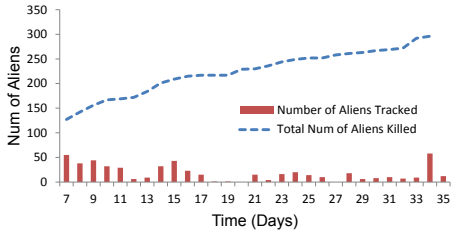

Fig. 11. Efficacy of Progressive Movement strategy: total number of aliens killed vs. number of aliens tracked by the players over time.

that each square be covered with at least three WiFi readings. Nevertheless, we also plot the squares covered with one or two readings. From the plot, we observe that 15% of the squares are covered with over 1000 readings each. We believe that these squares cover the campus center building and the labs, and the high number of readings is just a by-product of the fact that students spend most of the time there. The next two categories, between 11 and 1000 readings, represent 45% of the squares. Our future goal is to devise strategies that reduce the coverage in these categories while, at the same time, increase the coverage of the remaining categories.

Analysis of Mobile Game Incentives. Figure 10 presents the correlation between the number of aliens killed and area coverage. The results show that our strategy of placing the aliens strategically across the area achieves its goal, as we observe a clear correlation between the number of aliens killed and area coverage. For a more in-depth analysis, we investigate the performance of the Progressive Movement strategy. Figure 11 shows the total number of aliens killed over time and the number of aliens tracked by the players. We observe that the players have been actively tracking the aliens in the first two weeks, and this results in more aliens killed; consequently, higher area coverage was achieved. After that, since only areas located farther away from the players normal places and paths are left, we see a decrease in the number of tracked aliens. However, the players are still interested in killing the aliens on their paths. Finally, in the last week, we announced a number of prizes for the best ranked players. The prizes were of little monetary value. However, they work well to incentivize the players to track the aliens to the least popular regions as demonstrated by the sudden spike toward the end of the period. Given the symbolic value of these prizes, they could be considered similar to game-based incentives. Thus, they do not change the incentive assumptions of our user study.

Another factor that is expected to incentivize the users to continue playing, is the number of bullets that can be collected from around them. Figure 12 presents the correlation between the number of bullets and the number of aliens killed. As

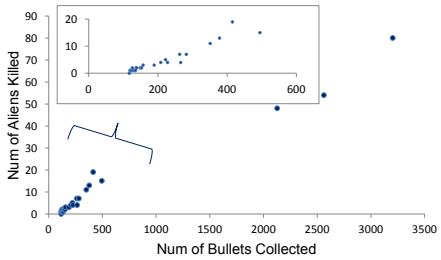

Fig. 12. Correlation between number of bullets and aliens killed. The calculated Pearson's correlation value is 0.98 (high positive correlation).

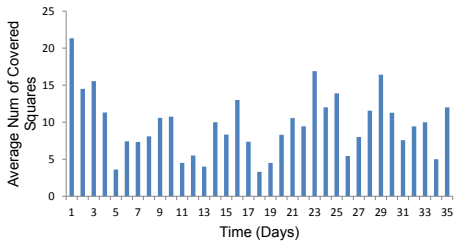

Fig. 13. Average number of covered squares per player per day.

already demonstrated, the number of killed aliens is a good indicator for area coverage. We observe two types of players: 1) players who collect many bullets around them, and 2) players who collect few bullets. The top 3 ranked players collected a lot of bullets, and subsequently were able to kill more aliens. These results provide two insights. First, the number of collected bullets is indeed a good, although indirect, indicator for area coverage. Second, this incentive may have to be re-designed to impact a larger number of users.

The next set of results analyze how active the players were during the study. Figure 13 shows the average number of covered squares per player per day. We observe a weekly cycle, in which the players are more active during the early part of the week. This is due to two reasons: (1) at the beginning of each week, we emailed the latest ranking to all participants and this proved to be a good incentive for players; (2) the students spend less time on campus during the weekend. This behavior can be leveraged in future games. For example, the game can be designed to provide additional in-game incentives for known periods of low activity. As expected, the players also show a clear daily pattern (Figure 14), which can be leveraged in future games as well. For instance, more aliens should be placed around the players during periods of high activity (e.g., 2PM-6PM).

Finally, we evaluate the indoor area coverage, which could be very difficult outside labs and classrooms. Since some offices are closed to students, while some buildings contain administrative offices where students do not "dare" to go, we believe that the 35% coverage of the upper floors achieved in the study is a promising result. The players mostly covered the hallways and open spaces in each building. Our conjecture is that incentives designed specifically for these places would probably help to increase this coverage. In addition, aliens must not be placed in inaccessible places. A crowdsourcing approach, in which players mark the inaccessible places while playing the game may prove the best solution for this problem.

Figure 15 plots the correlation of active players and the

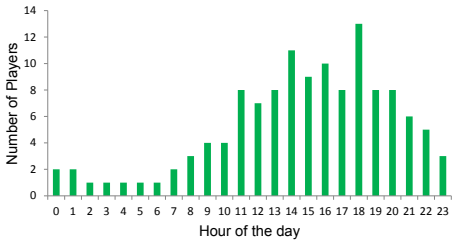

Fig. 14. Average daily patterns of active players.

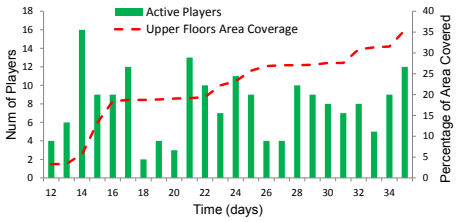

Fig. 15. Correlation of active players and the number of squares covered at floor levels > 1 over time in the last two weeks of the user study.

number of squares covered at upper floor levels over time (we started to place aliens on upper floors on day 12). These results demonstrate that indoor coverage correlates well with the number of active players, and the pattern is similar to outdoor coverage.

At the end of the user study, we collected game feedback from the players to understand the effectiveness of the game design by asking them "What made you to continue playing the Alien vs. Mobile User game?". We received answers from 16 players. The responses show that the majority of the players were curious about the game, and they liked tracking the aliens hiding in the campus buildings. Other reasons for playing, based on their counts in the players' answers, were: moving to the next game levels and being on top of the leaderboard; competing with friends; winning game achievements; and checking the game graphics. Lastly, very few players were interested to win the game prizes announced at the end.

VI. Related Work

There is significant literature on using gamification techniques in crowdsourcing. However, there is very little in terms of applying such techniques in crowdsensing. BioGame [13], a crowdsourced game, shows that in cases where the medical diagnosis is a binary decision (e.g., infected vs. healthy), it is possible to make accurate decisions by crowdsourcing the raw data (e.g., microscopic images of specimens/cells) using entertaining digital games. This game does not involve location or area coverage, which are major issues in mobile crowdsensing.

There are a few crowdsourcing games [19], [20] that accomplish sensing tasks by requiring the explicit participation of the players using the phone's keypad. Our crowdsensing game performs automatic sensing and does not require players to provide manual input nor to complete tasks not related to the game story. Like in any other mobile game, players can simply enjoy playing. Thus, our game has a higher probability to maintain the players' interest over time.

The work in [21] motivates mobile users to participate in a serious game that records as many audible signals as possible from different traffic lights. This data can be processed and integrated with Google maps to provide route accessibility information for blind pedestrians. The authors mention the possibility of automatic data gathering from smart phones' sensors, but did not discuss any specific model or architecture in detail. This game is limited to specific areas (i.e., around traffic lights) that are easily covered by people. Our work uses gamification to entice the users to many areas, some of them unpopular, to ensure uniform coverage of large areas.

BudBurst [14] is a smart phone application for an environmental participatory sensing project that focuses on observing plants and collecting plant life stage data. The main goal is "floracaching", for which players gain points and levels within the game by finding and making qualitative observations on plants. This game is also an example of motivating participatory sensing. Another participatory sensing game, Who [22], is used to extract relationship and tag data about employees. It was found useful for rapid collection of large volumes of high-quality data from "the masses". None of these participatory sensing games addresses the problem of area coverage in the context of crowdsensing.

Existing work in mobile health such as BeWell [23] utilizes smart phone sensing to assess the mobile users' wellbeing through scores based on their daily activities. In BeWell, an animated aquatic ecosystem is shown with three different animals, the behavior of each being affected by changes in the users' wellbeing. Thus, the users are motivated to maintain a healthy lifestyle. In a similar direction, our mobile game focuses on utilizing simple game graphics and in-game incentives to motivate smart phone users for achieving cost-effective crowdsensing.

Treasure [24] is a mobile game that collects the same data with our game. In Treasure, selected players using PDAs play against their opponents in a specific open space such as a large lawn where players have to collect coins that are scattered in the game area and upload the collected coins back to the server when they find WiFi connectivity. This game is not intended for sensing, and it has not been designed for collecting sensing data; the WiFi data is just used to help players quickly upload the collected coins. Furthermore, this game did not attempt to study area coverage efficiency. The main focus of this game is on player's gaming experience and their tactics and strategies in a multi-player game. In addition, the players are compensated to participate in the game. Our game is designed to enable fast crowdsensing coverage of large areas, and the players are not compensated: the fun of playing the game is the only incentive.

Micro-payments have been studied as an incentive for users to complete tasks in crowdsourcing (Amazon MTurk [25]), to control "free-riders" in peer-to-peer networks [26], [27], and to meter web content usage [28]. They have also been examined in the context of participatory sensing [10], [29], [30]. Some of the key findings are that incentives can be highly beneficial in recruiting participants and that micro-payments have the potential to extend participant coverage both spatially and temporally. However, attracting people to unpopular places

could be difficult and expensive. For example, the results from a recent crowdsensing study [9] show that many places will be infrequently visited. Our game, on the other hand, focuses on attracting players to such infrequently visited places and succeeds in covering the targeted area.

VII. Conclusions

In this paper, we explored a model for collecting crowdsensing data across large regions based on incentivizing smart phone users to play mobile sensing games. After analyzing three strategies to attract users to unpopular regions in order to cover the entire target area, we concluded that Progressive Movement leads to the lowest coverage latency while incurring a reasonable user effort as measured by the entropy of the area coverage. We designed a mobile sensing game, "Alien vs. Mobile User", which incorporates game story-related incentives to convince participants to cover all the regions of a target area. The game was prototyped for Android-based mobile devices and deployed as part of a user study in our campus. The user study results demonstrate that mobile gaming ensures high area coverage. Furthermore, the results show that our game is able to strike a good balance between attempting to attract users to the uncovered regions quickly and maintaining player interest in the game.

Acknowledgment

This research was supported by the National Science Foundation under Grants No. CNS 1409523, CNS 1054754, and DUE 1241976. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] O. Riva and C. Borcea, "The Urbanet Revolution: Sensor Power to the People!" *Pervasive Computing, IEEE*, vol. 6, no. 2, pp. 41–49, 2007.

[2] Smart phone sensing research @ Dartmouth college. [Online]. Available: http://sensorlab.cs.dartmouth.edu/research.html

[3] Urban sensing research @ UCLA. [Online]. Available: http://urban.cens.ucla.edu/

[4] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "mCrowd: A Platform for Mobile Crowdsourcing," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. ACM, 2009, pp. 347–348.

[5] M. Ra, B. Liu, T. La Porta, and R. Govindan, "Medusa: A Programming Framework for Crowd-Sensing Applications," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys'12)*. ACM, 2012, pp. 337–350.

[6] M. Talasila, R. Curtmola, and C. Borcea, "Improving Location Reliability in Crowd Sensed Data with Minimal Efforts," in *Proceedings of the 6th Joint IFIP/IEEE Wireless and Mobile Networking Conference (WMNC'13)*, 2013.

[7] G. Cardone, L. Foschini, C. Borcea, P. Bellavista, A. Corradi, M. Talasila, and R. Curtmola, "Fostering ParticipAction in Smart Cities: A Geo-Social CrowdSensing Platform," *IEEE Communications Magazine*, vol. 51, no. 6, 2013.

[8] M. Talasila, R. Curtmola, and C. Borcea, "ILR: Improving Location Reliability in Mobile Crowd Sensing," *International Journal of Business Data Communications and Networking*, vol. 9, no. 4, pp. 65–85, 2013.

[9] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the Coverage and Scalability of Place-centric CrowdSensing," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13)*. ACM, 2013, pp. 3–12.

[10] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava, "Examining Micro-Payments for Participatory Sensing Data Collections," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (Ubicomp'10)*. ACM, 2010, pp. 33–36.

[11] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: Sensing and Mapping for Better Biking," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'10)*. ACM, 2010, pp. 1817–1820.

[12] L. Deng and L. P. Cox, "LiveCompare: Grocery Bargain Hunting Through Participatory Sensing," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications (HotMobile'09)*. ACM, 2009, pp. 4:1–4:6.

[13] S. Mavandadi, S. Dimitrov, S. Feng, F. Yu, R. Yu, U. Sikora, and A. Ozcan, "Crowd-sourced BioGames: Managing the Big Data Problem for Next-Generation Lab-on-a-Chip Platforms," *Lab on a Chip*, vol. 12, no. 20, pp. 4102–4106, 2012.

[14] K. Han, E. A. Graham, D. Vassallo, and D. Estrin, "Enhancing Motivation in a Mobile Participatory Sensing Project through Gaming," in *Proceedings of 2011 IEEE 3rd international conference on Social Computing (SocialCom'11)*, 2011, pp. 1443–1448.

[15] Network Simulator - NS2. http://www.isi.edu/nsnam/ns/.

[16] W. Zheng-jie and L. Wei, "A Solution to Cooperative Area Coverage Surveillance for a Swarm of MAVs," *International Journal of Advanced Robotic Systems*, vol. 10, no. 398, pp. 1–8, 2013.

[17] Monsters vs NJIT. [Online]. Available: https://play.google.com/store/apps/details?id=com.mtlabs.games.avn

[18] Alien vs. Mobile User Game website. [Online]. Available: http://web.njit.edu/~mt57/avmgame

[19] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based Crowdsourcing: Extending Crowdsourcing to the Real World," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI'10)*. ACM, 2010, pp. 13–22.

[20] I. Celino, D. Cerizza, S. Contessa, M. Corubolo, D. Dell'Aglio, E. D. Valle, and S. Fumeo, "Urbanopoly - a Social and Location-based Game with a Purpose to Crowdsource your Urban Data," in *Proceedings of the 2012 IEEE International Conference on Social Computing (SocialCom'12)*, 2012, pp. 910–913.

[21] C. E. Palazzi, G. Marfia, and M. Roccetti, "Combining Web Squared and Serious Games for Crossroad Accessibility," in *Proceedings of 2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH'11)*, 2011, pp. 1–4.

[22] I. Guy, "Crowdsourcing in the enterprise," in *Proceedings of the 1st ACM International Workshop on Multimodal Crowd Sensing (CrowdSens'12)*. ACM, 2012, pp. 1–2.

[23] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell, "BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing," in *Proceedings of The 5th International ICST Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth'11)*, 2011, pp. 23–26.

[24] L. Barkhuus, M. Chalmers, P. Tennent, M. Hall, M. Bell, S. Sherwood, and B. Brown, "Picking Pockets on the Lawn: The Development of Tactics and Strategies in a Mobile Game," in *Proceedings of The Seventh International Conference on Ubiquitous Computing (UbiComp'05)*, 2005, pp. 358–374.

[25] Amazon Mechanical Turk. [Online]. Available: http://www.mturk.com

[26] B. Yang and H. Garcia-Molina, "PPay: Micropayments for Peer-to-Peer Systems," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003, pp. 300–310.

[27] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge, "Incentives for Sharing in Peer-to-Peer Networks," in *Proceedings of The Second International Workshop on Electronic Commerce (WELCOM'01)*, 2001, pp. 75–87.

[28] R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes," in *Proceedings of The 1996 International Workshop on Security Protocols*, 1996, pp. 69–87.

[29] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment Framework for Participatory Sensing Data Collections," in *Proceedings of The 8th International Conference on Pervasive Computing (Pervasive'10)*, 2010, pp. 138–155.

[30] M. Musthag, A. Raij, D. Ganesan, S. Kumar, and S. Shiffman, "Exploring Micro-Incentive Strategies for Participant Compensation in High-Burden Studies," in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp'11)*. ACM, 2011, pp. 435–444.