Human-Like Agents for a Smartphone First Person Shooter Game Using Crowdsourced Data

Christoforos Kronis, Andreas Konstantinidis, and Harris Papadopoulos

Department of Computer Science and Engineering, Frederick University, Nicosia, Cyprus

Abstract. The evolution of Smartphone devices with their powerful computing capabilities and their ever increasing number of sensors has recently introduced an unprecedented array of applications and games. The Smartphone users who are constantly moving and sensing are able to provide large amounts of *opportunistic/participatory data* that can contribute to complex and novel problem solving, unfolding in this way the full potential of *crowdsourcing*. Crowdsourced data can therefore be utilized for optimally modeling human-like behavior and improving the realizablity of AI gaming. In this study, we have developed an Augmented Reality First Person Shooter game, coined AR Shooter, that allows the crowd to constantly contribute their game play along with various spatio-temporal information. The crowdsourced data are used for modeling the human player's behavior with Artificial Neural Networks. The resulting models are utilized back to the game's environment through AI agents making it more realistic and challenging. Our experimental studies have shown that our AI agents are quite competitive, while being very difficult to distinguish from human players.

1 Introduction

The widespread deployment of Smartphone devices with their powerful computing capabilities and their ever increasing number of sensors has recently introduced an unprecedented array of applications (e.g., Google Play features over 650,000 apps with over 25 billion downloads[1]). A crowd of Smartphone users can be considered as a number of individuals carrying Smartphones (which are mainly used for sharing and collaboration) that are constantly moving and sensing, thus providing large amounts of *opportunistic/participatory data* [1–3]. This real-time collection of data can allow users to transparently contribute to complex and novel problem solving, unfolding in this way the full potential of *crowdsourcing* [4]. There is already a proliferation of innovative applications [5] founded on opportunistic/participatory crowdsourcing that span from assigning tasks to mobile nodes in a given region to provide information about their vicinity using their sensing capabilities (e.g., noise-maps [6]) to estimating road traffic delay [7] using WiFi beams collected by smartphones rather than invoking expensive GPS acquisition and road condition (e.g., PotHole [8].)

The real-time collection of realistic crowdsourced data can also unveil opportunities in the area of Artificial Intelligence, such as modeling human-like behavior [9–11] for social and socio-economic studies as well as for marketing and/or entertainment purposes. The latter is traditionally linked to game AI [12, 13], which as a term is mainly

[1] Sept. 26, 2012: Android Official Blog, `http://goo.gl/F1zat`

H. Papadopoulos et al. (Eds.): AIAI 2013, IFIP AICT 412, pp. 161–171, 2013.

used for describing non player characters (NPCs). The vast majority of AI games mainly relies on old AI technologies, such as A* and finite state machines [13]. More modern AI games, such as Unreal Tournament, Half-life, Warcraft and Supreme Commander that require more sophisticated AI utilize more recent methods such as reinforcement learning, neuroevolution, etc. (see e.g. [10, 11, 14]). However, most of these modern games rely on desktop-like game-playing [11] utilizing data based on the human's decisions only and ignoring spatio-temporal or other sensory information. Here it is important to note that the year 2011 has been extremely important for the Computing field, as the number of smartphones exceeded for the first time in history the number of all types of Personal Computers combined (i.e., Notebooks, Tablets, Netbooks and Desktops), marking the beginning of the post-PC era[2] and therefore the beginning of real-time Smartphone gaming.

In this study, we developed an Augmented Reality First Person Shooter game [11], coined AR Shooter, that allows the crowd to constantly contribute their game play along with various spatio-temporal information. The crowdsourced data are collected by utilizing almost every sensor of the Smartphone device and they are used for training AI agents. Specifically, the AR Shooter is founded on a cloud-based framework that is composed of a back-end that collects the data contributed by the crowd, which are used for training a group of Artificial Neural Networks (ANN) to model the human player's behavior. The resulting models are in turn incorporated back into the game's environment (front-end) through AI agents, making it more realistic and challenging. Our experimental studies have shown that our AI agents are not easy to distinguish from human players and perform much better than non-intelligent agents.

2 Problem Definition

It is straight forward to take a well-defined game, add some control structures for agents and begin teaching them. First person shooter (FPS) games and especially augmented reality first person shooter games are not well defined. There are no defined player tactics in an FPS game. The main purpose of the game is the survival of the fittest, to shoot or to be shot. The player is rewarded higher score the more competitive opponents he/she eliminates. But if the player is "trigger happy" or not aware of his surroundings he might end up getting caught off guard. The penalties for death are severe, letting the player wait for 15 seconds before continuing the competition. Meanwhile the other opponents have the opportunity to pick up geo-located health resources or even finish off a mutual opponent, whom the player has been hunting since the beginning of the game, leaving him/her with no points for the elimination. Even expert players cannot fully explain their tactics, but only give out general tips like not rushing into many enemies or staying in the same position for a large amount of time. Furthermore, there are many exceptions to these rules that make it extremely difficult to hard code a rule based system.

Initially, it is important to figure out the actions to be considered and the data needed for deciding what action to make. Ideally an FPS learning model would consider every

[2] Feb. 3, 2012: Canalys Press Release, `http://goo.gl/T81iE`

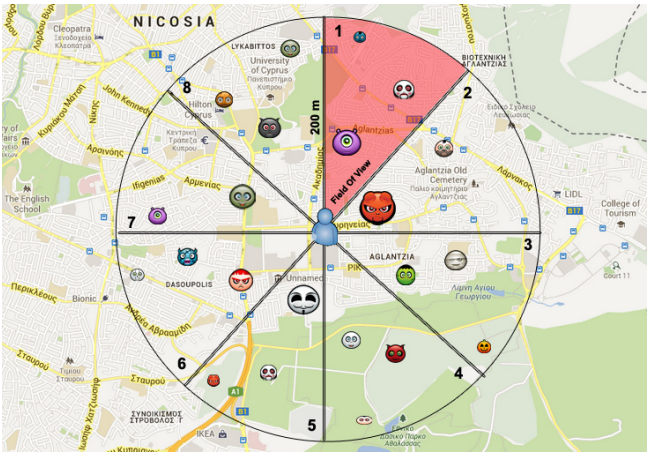


Fig. 1. The surrounding environment of the players is split into eight sectors in order to collect spatio-temporal information. The closest an opponent's avatar is to the user, the larger appears in the augmented reality view.

action made at any given time. In this study, however, we consider the following three basic actions:

- move: north/northeast/east/southeast/south/southwest/west/northwest/don't move
- select weapon: melee/assault/shotgun/sniper
- shoot: shoot/don't shoot

The heart of the problem is to make a choice for any of these decisions. How does an agent know if it's time to move north, change weapon and shoot? In order to achieve human like behavior we must model these decisions after actual players and learn how these decisions are made. The problem now becomes a more traditional machine learning problem. But many of the decisions made by players are reflexive and don't have an explanation behind them. Therefore, a good set of features must be found to adequately represent the environment of the game, but the more the features being considered, the higher the complexity of the model.

2.1 Feature Set and Data Set

The set of actions investigated in this study are movement, weapon selection and whether to shoot or not. These three actions will be the output of the model. The next step is to decide the information needed for making these decisions. One of the most important pieces of information is the enemy locations. Spatial data about enemy location were determined by breaking the surrounding environment of the agent into 8 sectors with a radius of 200 meters as shown in Figure 1. The sectors are relative to the player as actual world position is not as important as information about the player's immediate surroundings. Moreover the distance, health and sector of the closest enemy as well as the distance and sector of the closest resource available were recorded. Furthermore,

Table 1. The data used

Input (current information about the world around the player)	
Player Latitude	Current latitude of the player
Player Longitude	Current longitude of the player
Player Health	Current health of the player
Player weapon	Current weapon of the player
Player heading	Current heading of the player
Player speed	Current speed of the player
Player points	Current points of the player (one point for every kill, reset to zero when player dies)
Player time alive	The current time alive of the player
Closest Resource distance	Distance of the closest resource
Closest Resource sector	Sector to the closest resource
Closest Enemy Distance	Distance to the closest enemy
Closest Enemy Health	Health of the closest enemy
Closest Enemy Sector	Sector of the closest enemy
Enemy Shooting Sector	Sector of the enemy shooting at the player
Enemy Shooting Health	Health of the enemy shooting at the player
Enemy Shooting Distance	Distance of the enemy shooting at the player
Enemy Shooting Weapon	Selected Weapon of the enemy shooting at the player
Enemies per sector	The total number of enemies per sector
Player Shooting	If the player is shooting an enemy
Output (collected at next time step after input is sampled)	
Move direction	0 = do not move, 1 = move north etc.
Weapon	0 = melee,1 = assault, 2 = shotgun, 3 = sniper
Shoot	0 = do not shoot, 1 = shoot

information about the enemy shooting at the player is useful like distance, sector and health.

In addition to enemy information, the player must be constantly aware of his health, heading, time alive, selected weapon and speed. This will allow us to determine tactics like retreat or advance. Table 1 summarizes the features considered in this study along with a brief description. These data were measured every five seconds. The output was a combination of the three basic actions: (i) move, (ii) select weapon and (iii) shoot.

3 Proposed Framework

In this section, the proposed framework is introduced, beginning with a general overview of the architecture and followed by a description of the ANNs used for modeling the human-like behavior of our AI agents and an introduction to the Graphical User Interface of our Windows Phone prototype system.

3.1 Overview

The proposed framework (see Figure 2) is mainly composed of the cloud-based back-end and the Smartphone front-end. The back-end consist of a Java server with a SQL

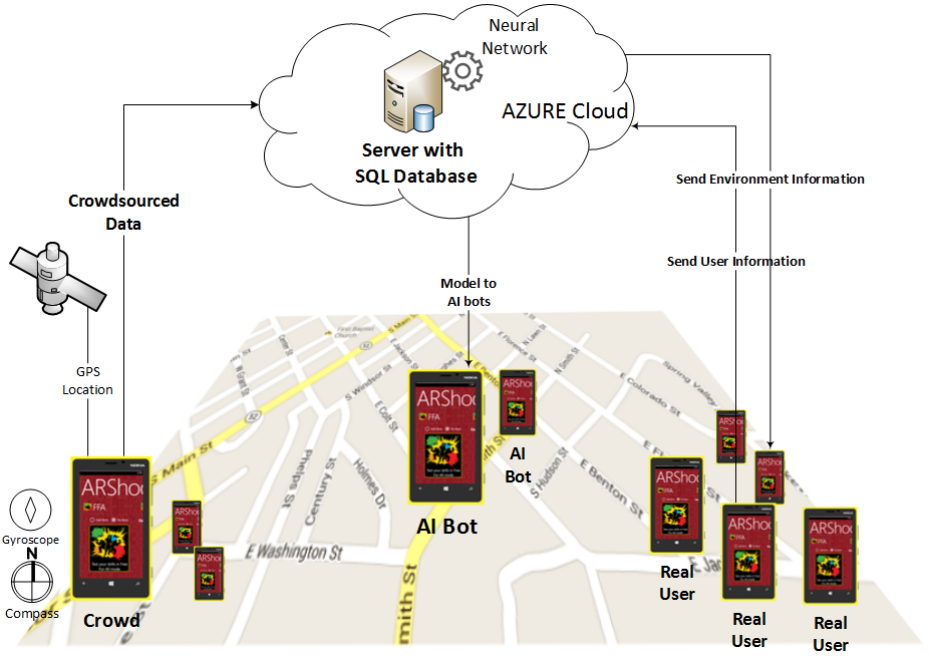
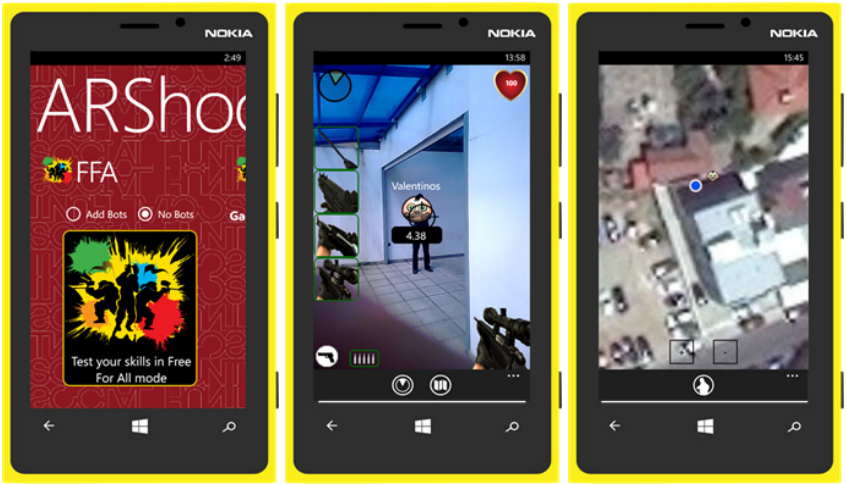


Fig. 2. The proposed architecture. The crowd contributes spatio-temporal information to the server on the clowd (left). The server uses the crowdsourced data to train AI agents that are deployed in the game's environment (center). The real users exchange information with the server regarding their location, the opponents' locations, health etc. (right).

database, which are deployed on the Microsoft's Azure cloud. The server side is responsible for (1) collecting the data contributed by the crowd, which are stored and maintained in the database, as well as (2) coordinating the game-play of the AR Shooter.

1) Collecting the Crowdsourced Data: Each user is able to locally store his/her gameplay on his/her Smartphone. The data stored are those summarized in Table 1. In order to collect the users' data almost all sensors and connection modalities of the smarthone are utilized including the compass, gyroscope, accelerometer, GPS, 3G, WiFi and NFC. Note that the Near Field Communication (NFC) technology is only utilized when it is available on the Smartphone for mainly sharing resources with other team-players, such as life, weapons, ammunition etc. Then the user uploads to the server the collected data after finishing playing. Here it is important to note that a permission is granted from the user for sharing his/her data at login.

2) Coordinating the Game-Play: The server is responsible for the communication between the real Smartphone users as well as for coordinating the game play of the NPCs and the "dummy" agents (i.e., non-intelligent agents that mainly rely in random decisions). The communication between the real-users and the server is based on the TCP/IP protocol. The users's utilize their sensors to calculate their heading, location



(a) Main Page (b) Augmented Reality view (c) Map view

Fig. 3. The GUI of the ARShooter Wars- Main Page and Views

and field of view (this is only needed in the augmented reality view that is introduced next). The server on the other hand, forwards to the users information about nearby users (e.g.,their health and location) and geo-located resources (such as medical packs, ammunition and weapons).

3.2 Graphical User Interface

The Graphical User Interface of our prototype system is interactive as well as easy to learn and use. It includes different view types and game modes and various features. Figure 3 (a) shows its main page. Figure 3 (b) shows the Augmented Reality view of a user, indicating his/her field view, an opponent that is engaged by the user's target (in the center of the smartphone screen) as well as the user's health in the upper right corner, his/her ammunition in the left bottom corner and a selection of weapons in the left side of the screen. Figure 3 (c) shows the map view with the user denoted by a solid circle and the opponents with their selected avatars. Note that the icon in the bottom center of the smartphone screen can be used to switch between the augmented reality and the map views.

Moreover, Figure 4 (a) shows the store where users can choose weapons, ammunition, etc., as well as unlock more advanced features (e.g., grenades) after collecting a specific amount of points (based on their kills). The specifications (i.e., range, damage and angle) of the weapon are summarized just below its image. Figure 4 (b) shows the animation when the user is being shot followed by a vibration of the smartphone and a decrease of his/her health (the decrease is based on the weapon that the opponent used.) Finally, Figure 4 (c) shows the screen that appears when a user is eliminated giving details about the opponent that eliminated him/her along with the user's number of kills. The user automatically re-enters the game after 15 seconds.

(a) The user selects a weapon

(b) The users is shot

(c) The user is eliminated and must wait for 15 seconds to re-enter the game

Fig. 4. The GUI of the ARShooter Wars - User functionality and actions

The ARShooter supports both a multi-player and a single-player modes. In the multi-player mode, the user plays against other human users, the "Dummy" agents and the AI agents. In the single player mode, the user plays only against the agents. In both modes, the game is in real time and in a real environment.

3.3 Human-Like Behavior Modeling

The developed model comprised of three ANNs, one for each of the three decisions: movement, weapon selection and shooting. The ANN were implemented using the WEKA data mining workbench. They had a two-layer fully-connected structure, with 26 input neurons and eight, four and one output neurons, respectively. Their hidden and output neurons had logistic sigmoid activation functions. They minimized mean squared error with weight decay using BFGS optimization. The network parameters were initialized with small normally distributed random values. All input features were normalized by setting their mean to zero and their variance to one. The combination of outputs produced by the three ANN given the current input formed the set of actions the agent would take next.

For determining the number of hidden neurons to use, a 10-fold cross-validation process was followed on the initially collected data trying out the values 5 to 50. The Root Mean Squared Errors (RMSEs) obtained are summarized in Table 2 with the best values denoted in bold. The number of hidden units that gave the best RMSE for each ANN was then used in the final model.

Table 2. The Root Mean Squared Error for each model (move, select weapon, shoot) with 5-50 hidden units. The best results are denoted in bold.

# of Hidden Units:	5	10	15	20	25	30	35	40	45	50
Movement:	0.119	0.120	0.110	0.110	0.104	0.107	0.107	**0.092**	0.097	0.098
Shoot/Do not Shoot:	0.014	0.012	0.007	0.007	0.013	0.009	0.006	0.005	0.005	**0.004**
Select Weapon:	0.24	0.22	0.213	0.213	0.199	0.19	0.189	**0.182**	0.188	0.187

4 Experimental Studies

This section summarizes the experimental setup used during our experimental studies and introduces two experimental series for evaluating the performance and human-like behavior of our AI agents.

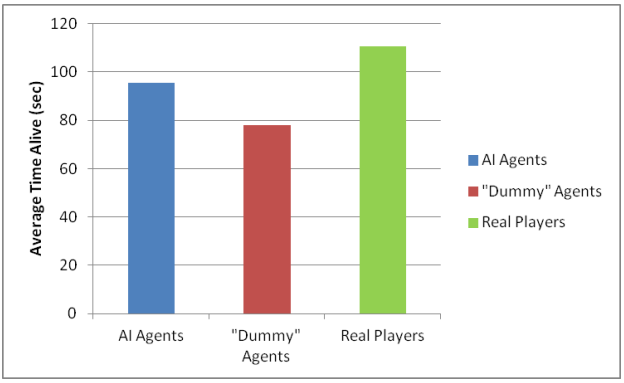
4.1 Experimental Setup

DataSet: The data were collected by sampling the game play of ten players, from which we have collected more than 8000 individual decisions. For each decision the information described in Table 1 were recorded. Data were perpetually collected using crowdsourcing as explained earlier in Subsection 3.1 and added to the data set.

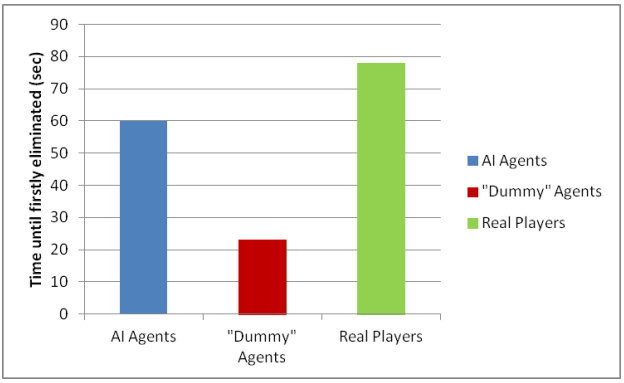
Evaluation and Settings: The performance of the proposed AI agents was compared against a number of human players and a set of "dummy" agents, which are mainly making random decisions, in terms of time (in seconds) that the agent stayed alive in the game and number of kills (game points). Finally, the last experiment of this section evaluates how well the AI agents mimic human behavior by asking ten real-users to try finding the AI agent through the map view in 60 seconds in an environment composed of one AI agent and several other human users. The results below are averaged over five individual runs.

4.2 Experimental Results

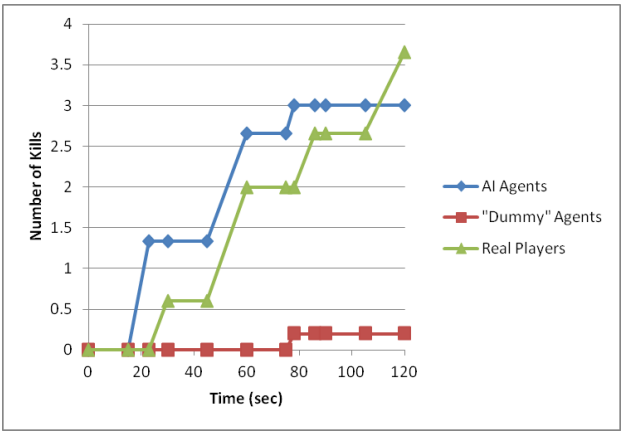
Experimental Series 1 - Performance: In experimental series 1, we have evaluated the average performance of ten AI agents against ten "Dummy" agents and ten human users with respect to the average time they stayed alive in the game (note that there is a 15 seconds penalty each time they are eliminated) and the number of kills they achieved in 120 seconds as well as the time they stayed alive until they are eliminated for the first time. The results of Figure 5 show that the AI agent performs much better than the "Dummy"-random agents and similar to the real users in all cases. Moreover, the AI agents achieve more kills than the human players at the beginning of the game, showing that they adapt faster to the environment. However, the human users manage to stay alive for a longer time than the AI agents on average and achieve more kills after around 120 seconds.



(a) Average Time Alive in 120 seconds (with penalty 15 sec after elimination)



(b) Time Alive until eliminated for the first time



(c) Number of Kills in 120 seconds

Fig. 5. Experimental Series 1 - Performance: AI agents compared against "Dummy" agents and real-players in terms of average time alive, number of kills and average time until they have been eliminated for the first time

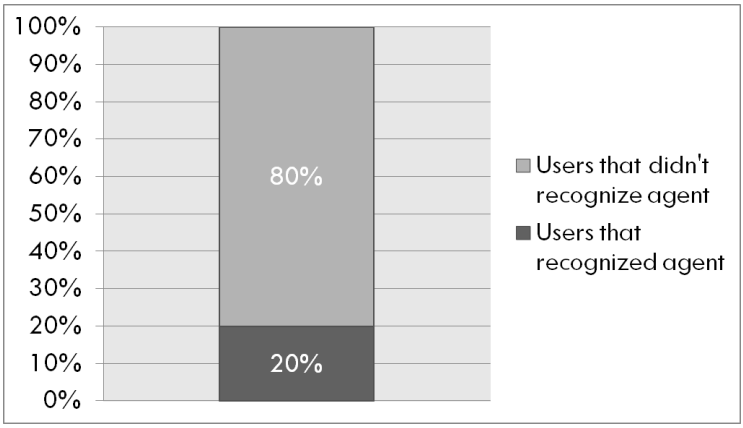


Fig. 6. Experimental Series 2 - Human-Like Behavior: A case study for evaluating how well the AI agent approximates the human behavior in the ARShooter Wars

Experimental Series 2 - Human-Like Behavior: In experimental series 2, we have evaluated how well our AI agents mimic the human behavior by asking ten players to identify the AI agents with respect to nine other human players in 60 seconds using the map view (see Figure 3 (c)). The results that are summarized in Figure 6 show that 80% of the players did not manage to identify the AI agent because its behavior was very close to that of the human users.

5 Conclusions and Future Work

In this study, we have developed an Augmented Reality First Person Shooter game, coined AR Shooter, that allows the crowd to constantly contribute their game play along with various spatio-temporal information. Then three ANN are trained using the crowdsourced data to model the human player's behavior. The resulting model is in turn utilized back into the game's environment through AI agents making it more realistic and challenging. Our experimental studies have shown that our AI agents have good performance, but most importantly they are very difficult to distinguish from human players.

Acknowledgments. This work was supported by the Mobile Devices Laboratory (MDL) grant funded by the Department of Computer Science and Engineering of the Frederick University, Cyprus and Microsoft Cyprus. We would like to thank MTN Cyprus for assisting our experimental studies by providing free 3G Sim Cards and Mr. Valentinos Georgiades, Microsoft Cyprus for his support and valuable comments. We are also grateful to the MDL Developers Constantinos Marouchos, Yiannis Hadjicharalambous and Melina Marangou for helping in the development of the AR Shooter Wars game.

References

1. Campbell, A., Eisenman, S., Lane, N., Miluzzo, E., Peterson, R., Lu, H., Zheng, X., Musolesi, M., Fodor, K., Ahn, G.: The rise of people-centric sensing. IEEE Internet Computing 12(4), 12–21 (2008)
2. Das, T., Mohan, P., Padmanabhan, V.N., Ramjee, R., Sharma, A.: Prism: platform for remote sensing using smartphones. In: MobiSys (2010)
3. Azizyan, M., Constandache, I., Choudhury, R.-R.: Surroundsense: mobile phone localization via ambience fingerprinting. In: MobiCom (2009)
4. Chatzimiloudis, G., Konstantinidis, A., Laoudias, C., Zeinalipour-Yazti, D.: Crowdsourcing with smartphones. IEEE Internet Computing (2012)
5. Konstantinidis, A., Aplitsiotis, C., Zeinalipour-Yazti, D.: Multi-objective query optimization in smartphone social networks. In: 12th International Conference on Mobile Data Management, MDM 2011 (2011)
6. Rana, R.K., Chou, C.T., Kanhere, S.S., Bulusu, N., Hu, W.: Ear-phone: an end-to-end participatory urban noise mapping system. In: IPSN, pp. 105–116 (2010)
7. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: SenSys 2009: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, pp. 85–98. ACM, New York (2009)
8. Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H.: The pothole patrol: using a mobile sensor network for road surface monitoring. In: MobiSys, pp. 29–39 (2008)
9. Conroy, D., Wyeth, P., Johnson, D.: Modeling player-like behavior for game ai design. In: Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, ACE 2011, pp. 9:1–9:8. ACM, New York (2011)
10. Harrison, B., Roberts, D.L.: Using sequential observations to model and predict player behavior. In: Proceedings of the 6th International Conference on Foundations of Digital Games, FDG 2011, pp. 91–98. ACM, New York (2011)
11. Wang, D., Subagdja, B., Tan, A.H., Ng, G.W.: Creating human-like autonomous players in real-time first person shooter computer games. In: 21st Annual Conference on Innovative Applications of Artificial Intelligence (IAAI 2009), Pasadena, California, July 14-16 (2009)
12. Missura, O., Gärtner, T.: Player modeling for intelligent difficulty adjustment. In: Proceedings of the ECML 2009 Workshop From Local Patterns to Global Models, LeGo 2009 (2009)
13. Yannakakis, G.N.: Game ai revisited. In: Proceedings of the 9th Conference on Computing Frontiers, CF 2012, pp. 285–292. ACM, New York (2012)
14. Kienzle, J., Denault, A., Vangheluwe, H.: Model-based design of computer-controlled game character behavior. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 650–665. Springer, Heidelberg (2007)