Forthroid on Android: A QR-code based Information Access System for Smart Phones

Anastasios Alexandridis*, Paulos Charonyktakis*, Antonis Makrogiannakis*,
Artemis Papakonstantinou*, and Maria Papadopouli*†‡
*Department of Computer Science, University of Crete
*Institute of Computer Science, Foundation for Research and Technology - Hellas
†School of Electrical Engineering, KTH Royal Institute of Technology

***Abstract*—The Forthroid is a location-based system that "augments" physical objects with multimedia information and enables users to receive information about physical objects or request services related to physical objects. It employs computer-vision techniques and Quick Response codes (QR-codes). We have implemented a prototype on Android platforms and evaluated its performance with systems metrics and subjective tests. We discuss our findings and challenges in prototyping on Android OS. The analysis indicates that the network and the server are the main sources of delay, while the CPU load may vary depending on the specific Forthroid operation. The preliminary subjective test results suggest that users tolerate these delays and the offered services can be particularly useful.**

I. Introduction

The emergence of wireless networks and their continuous deployment world-wide has triggered a fast growth of location-aware and pervasive computing services and applications. We designed and evaluated the Forthroid, a location-based system and set of services, that enables users to obtain multimedia information about specific points of interest using their mobile phones and provides services related to physical objects (e.g., a printing service). The Forthroid can be easily deployed and used in indoor and outdoor environments, such as museums, campuses, and metropolitan areas. The Forthroid uses Quick Response codes (QR-codes), which are two-dimensional barcodes that can encode various types of information. QR-codes can be attached to physical objects. Already in several metropolitan areas, such codes have been placed, e.g., in stores, at the entrance of buildings to provide floorplan information, on walls for announcing upcoming events. An example of usage of the Forthroid is illustrated in Fig. 2.

The Forthroid architecture consists of QR-codes (associated with physical objects in proximity), Forthroid clients, that run on Android smart phones, and the Forthroid multi-threaded server. A Forthroid client may scan QR-codes (associated with a certain point of interest) and communicate with the Forthroid server via the Internet in order to retrieve multimedia information or request services associated with that point of interest. Although the community has proposed similar QR-code information access systems (e.g., [1]–[5]), to the best of our knowledge, there are no empirical studies that analyze their performance.

‡ Contact author: Maria Papadopouli (email: mgp@ics.forth.gr)

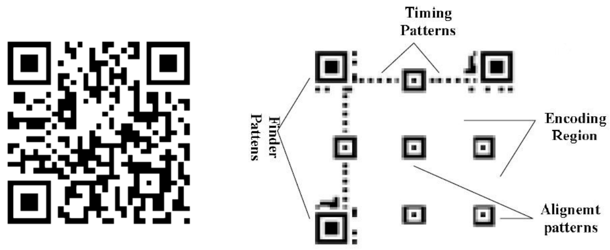
The Forthroid builds on our earlier work, the PhotoJournal [6]. The PhotoJournal is a novel location-based media sharing application that enables users to build interactive journals that associate multimedia files with locations on maps and share this information with other users. The contribution of this paper is twofold: it presents the design and architecture of the system, and its performance evaluation using subjective tests and system metrics, such as the delay, distance estimation error, battery consumption, and CPU load. Moreover, it discusses some of the main challenges that programmers often face, when developing similar applications. This paper is organized as follows: Section II presents related work, while Section III focuses on the architecture of the system. The performance of the system and the user study are discussed in Section IV. Finally, Section V summarizes our main conclusions and future work plans.

II. Related Work

The community has developed several exciting location-aware and mobile computing applications. For example, the CoolTown project [7] of the HP labs is a location-aware ubiquitous system that explores opportunities provided by the convergence of Web technology, wireless networks, and portable client devices to support "web presence" for people, places, and things by associating URLs with visual codes placed on objects. Several mobile computing applications have been using barcodes [1]–[5]. For example, Ljungstrand *et al.* [3] proposed barcode stickers, which can be attached to physical objects. Barcode stickers map web pages to physical objects and enable users to access the web by scanning an attached barcode.

A comparative performance analysis of several vision-based distance estimation algorithms is presented by Maidi *et al.* [8]. The authors developed a hybrid approach to distance estimation that mixes an iterative and an analytical method for pose estimation. Their approach is compared to several sophisticated vision-based algorithms. The execution time of their hybrid approach is 112 μs for one pose estimation and the mean distance error is 7.2 mm for a distance of 1 m.

Sony has developed a new type of marker named CyberCode [1], along with a visual tagging system with tracking capabilities. Ordinary mobile phones can easily recognize CyberCode tags, enabling the estimation of the 3D position and orientation



(a) (b)

Fig. 1. (a) A typical QR-code. (b) The main components of a QR-code.

(a) (b)

Fig. 2. (a) A user scans a QR-code in order to retrieve information about a point of interest at the Telecommunications and Networks Laboratory (FORTH). (b) A screen snapshot of the information returned to a user for a specific point of interest.

of the camera relative to the tag. TRIP [2] is a computer vision system which uses printable 2D circular markers for the identification and location estimation of objects. It can successfully recognize an object 98% of the cases with an average error of less than 3 cm and identify entities within 3 meters distance. Rukzio *et al.* [4] proposed a system that enables mobile devices to "interact" with posters using cameras and near-field networks. Rohs *et al.* [5] utilized 2D visual codes, similar to QR-codes, in order to retrieve object-related information. Their recognition algorithm can detect multiple codes simultaneously and compute the coordinates of the target in the coordinate system induced by the code.

III. System Architecture

The main entities of the Forthroid are the QR-codes and their main components, namely, the finder patterns, timing patterns, alignment patterns, and the encoding region (as shown in Fig. 1). A unique identifier is encoded in each QR-code. The Forthroid defines and uses two QR-code types, namely, the *information-aware* and *action-aware* QR-codes. The information-aware QR-codes are attached to points of interest and are associated with specific multimedia information, while the action-aware QR-codes map the corresponding point of interest with certain services that can be provided to users.

The Forthroid architecture employs the client-server paradigm: it consists of the application that runs on the mobile device (client) and the server. The client uses the ZXing Application Programming Interface (API) and the Barcode Scanner open-source application provided by Google, for scanning and decoding QR-codes. The server consists of two modules, namely, the information retrieval, and the distance estimation module. To support information retrieval, the server deploys a MySQL database that contains information regarding all points

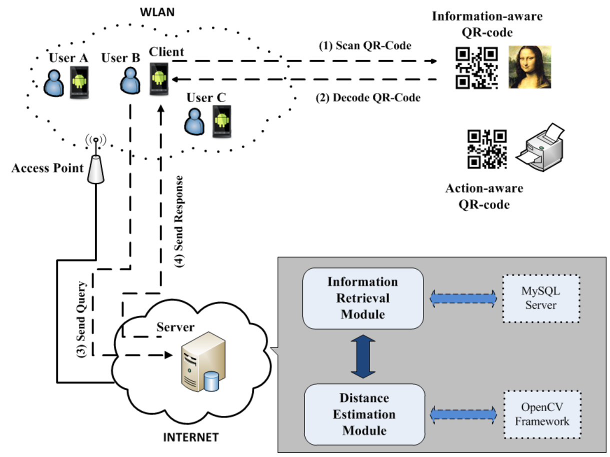


Fig. 3. The Forthroid architecture.

of interest, while the OpenCV framework is used to perform the distance estimation. The server is multi-threaded and can support a significant number of clients/requests.

Fig. 3 illustrates the general architecture of the Forthroid. A user holding an Android smart phone points to a QR-code (which is associated with a point of interest, for example a painting). The Forthroid mobile application detects the QR-code and decodes its unique identifier locally. A query for that point of interest is formed, containing the identifier of the specific QR-code and the captured image. The client sends the query to the server via the Internet. When the server receives a query, it determines the type of the QR-code based on the enclosed identifier. If the type corresponds to an information-aware QR-code, the information retrieval and distance estimation modules are invoked. The information retrieval module accesses the database to retrieve the appropriate information and the distance estimation module performs camera calibration. The server forms a response that contains the multimedia information about the point of interest and the estimated distance between the smart phone and the QR-code, and sends the response to the client.

In the case of an action-aware QR-code, action-specific operations are performed by the server. Currently, we have designed and evaluated the printing service: when a user scans a QR-code, which is attached to a printer, a list of all the points of interest he/she has visited is displayed. The user can then choose an item from this list and information about that item stored in the Forthroid server will be sent to the printer. The Forthroid server keeps track of the points of interest that recent Forthroid users have visited. Specifically, each time a client sends a query requesting information for a point of interest, the server creates a new entry in the database with the IP of the client and the identifier of the point of interest. Since IPs are dynamically assigned and a new client may have the same IP as an old one which has "left" the network, these entries expire after a certain period of time.

The main operations of the Forthroid take place during the preview and the query periods. Specifically, during the

TABLE I
VARIOUS EVENT TYPES. THE TERM T_i INDICATES THE TIME THE EVENT i WAS RECORDED AT THE CORRESPONDING MONITOR

Time	Event description
T_1	client successfully decodes a QR-code
T_2	client unicasts the query to the server
T_3	server receives the query
T_4	server sends back the response
T_5	client receives the response from the server
T_6	client displays the response to the user

TABLE II
DIFFERENT DELAY TYPES BASED ON THE RECORDED EVENT TYPES

Delay	Time
Server	$T_4 - T_3$
Network	$T_5 - T_4 + T_3 - T_2$
Android client	$T_2 - T_1 + T_6 - T_5$
Total	$T_6 - T_1$

preview period, the Forthroid client uses the mobile camera to *iteratively* scan the area until it is able to detect and decode a QR-code. The query period includes the time required for the client-server communication and the display of the results on the screen of the smart phone.

The distance of the smart phone from the plane that the QR-code is attached is estimated by camera calibration techniques. The camera calibration process determines the matrix M (Eq. (1)) that projects a 3D point P from the world coordinate system to a 2D point p in the coordinate system of the image [9]:

$$p = MP \tag{1}$$

where M is the 3x4 *projective or complete camera calibration matrix*. Eq. (1) can be further decomposed into:

$$p = K[R|t]P \tag{2}$$

where K is the 3x3 matrix of the *intrinsic* camera specific parameters and $[R|t]$ is the 3x4 *joint rotation and translation matrix*, i.e., the matrix of the *extrinsic* parameters. The extrinsic parameters describe the position of the camera in terms of orientation and translation with respect to the world coordinates. In order to estimate the extrinsic camera parameters, first the correspondences between points in the world and image coordinate systems must be identified. Then, the camera matrix M can be estimated using several methods for camera calibration, such as the Least Squares [10], nonlinear estimation [11], and iterative algorithms [9].

The Forthroid uses the corners of the finder patterns as the calibration pattern. Specifically, 36 correspondences are created and each of them is associated with a specific corner of a finder pattern square. The upper-left corner is used to define the origin of the world coordinate system. The camera calibration process is divided into two phases: the intrinsic parameters calibration, which is carried out only once, and the extrinsic parameter estimation. The intrinsic camera calibration is done using the MATLAB Calibration Toolbox [12]. When a query arrives at the server, the extrinsic parameter calibration is performed in order to estimate the distance of the smart phone from the QR-code. For that, first, the image is rotated counterclockwise so that the position of the finder patterns in the image matches the position of the finder patterns in the actual QR-code (lines passing from the centers of the finder patterns form a "Γ" shape). The angle of the rotation is estimated by inspecting the position of the centers of the finder patterns in the captured image and is included in the client query. Then, the corner detection procedure is carried out in order to create the world to image correspondences. The Harris corner detection algorithm [13] is applied on the three finder patterns of the QR-code. Specifically, Harris corner detector aims to find the twelve corners that correspond to the corners of the three squares inside each finder pattern. The resulted correspondences are then used in the extrinsic parameter estimation. Then, the 3D world coordinates of the location of the camera, i.e., camera center C, can be determined by Eq. (3) [14]:

$$C = -R^{-1}t \tag{3}$$

The distance of the user from the QR-code is the Euclidean distance of the camera center from the origin of the world coordinate system.

IV. PERFORMANCE ANALYSIS

To evaluate the performance of the Forthroid, empirical-based measurements and subjective tests were performed. Specifically, first we measured the delay, the distance estimation error, and the battery consumption of the Forthroid running on the Android device. Then, we performed a preliminary subjective study in the premises of our lab at FORTH-ICS.

The testbed includes an HTC Nexus One smart phone that runs Android 2.2 and a 2.66 GHz Core 2 Duo Dell Desktop with 2048 MB RAM and Linux Ubuntu Operating System server. The server is connected via FastEthernet and the smart phone (client) is connected via an IEEE802.11 Access Point (AP) to the FORTH-ICS infrastructure network.

A. Delay

As it is illustrated in Tables I and II, the total delay that a user experiences consists of the following components:

1) *Server delay*: it corresponds to the total time elapsed between the reception of a query by the server and the transmission of a response. Its dominant components are the distance estimation and the information retrieval modules.
2) *Network delay*: it includes the network and propagation delay.
3) *Android client delay*: it consists of the time for the query generation and the time for displaying the server response on the screen.

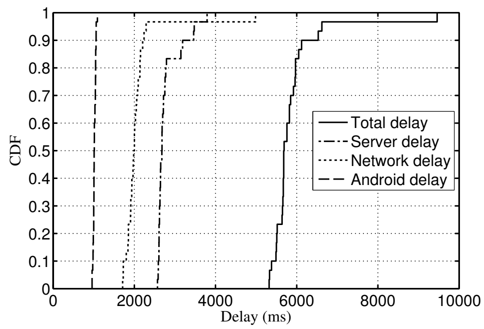


Fig. 4. Cumulative distribution function of the various types of delays.

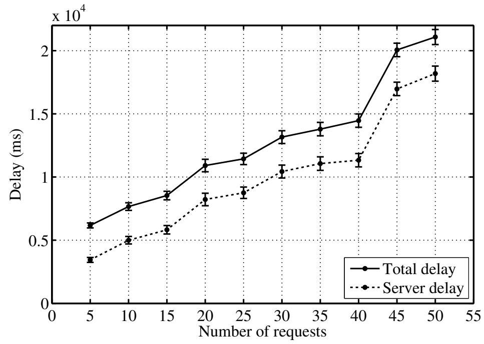


Fig. 5. Mean delay vs. number of concurrent requests (95% confidence intervals).

To measure these delays, the events shown in Table I were defined. When a specific event occurs, our software invokes a system call in order to get the current system time in milliseconds. During the evaluation phase, a Forthroid client scans an information-aware QR-code. This process was repeated 30 times. The median total, server, network, and android delay are 5.68, 2.68, 1.99 and 1.01 seconds, respectively, with a relatively small variance (as shown in Fig. 4).

To study the impact of the number of simultaneous requests on the delay, we used a desktop PC that sends to the Forthroid server a number of concurrent simulated queries for points of interest. We varied the number of concurrent requests from 5 to 50 (step of 5). Each experiment for a specific number of concurrent requests was repeated 10 times and the total delay, as well as the server delay, were measured (as shown in Fig. 5). A prominent increase in the delay occurs when the number of concurrent requests is 40 or more, due to the substantial increase in the memory required for processing them. As a result, some threads may have to wait for others to finish, increasing the server and the total delay.

B. Distance estimation

To test the accuracy of the distance estimation module, a QR-code of size 11x11 cm^2 is scanned at various distances, in the range of 40 cm up to 2 m, at every 10 cm. The aforementioned distance range (40 cm, 2 m) corresponds to the minimum and maximum distance, respectively, for which the ZXing library is able to detect and decode a QR-code. The

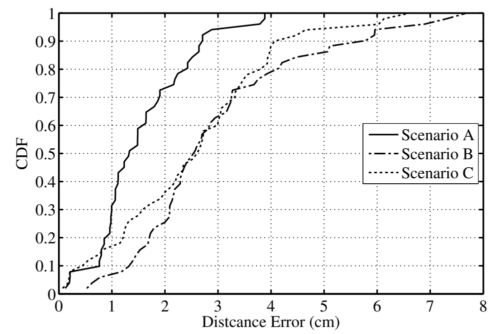


Fig. 6. Cumulative distribution function of the distance estimation error.

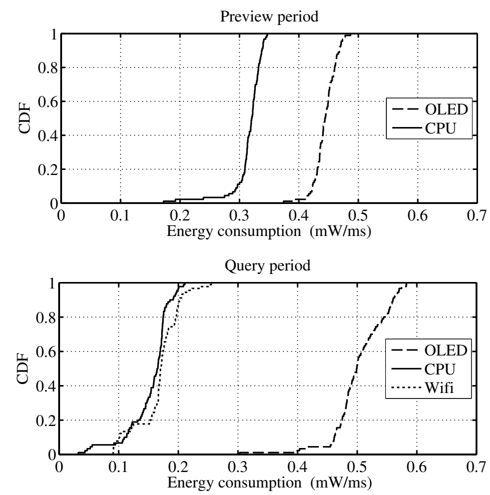


Fig. 7. Cumulative distribution function of the battery consumption during the preview and query periods.

following scenarios are employed in our measurements:

- **Scenario A**: The user is located in front of a QR-code, holding the camera parallel to the surface on which the QR-code is mounted.
- **Scenario B**: The user scans the QR-code holding the camera from different angles.
- **Scenario C**: The user holds the camera with an arbitrary rotation. As a result, the QR-code is rotated in the captured image. The user can stand either in front of the QR-code or at an arbitrary angle.

Each scenario was repeated three times for each distance (resulting in a total of 3x17 experiments). Fig. 6 illustrates the error between the reported and the actual distance. The median error for the scenario A is 1.3 cm, while for both the scenarios B and C, it is 2.6 cm. Moreover, the application reports a distance estimation at 94%, 84% and 87% of the cases for scenario A, B, and C, respectively. The distance estimation procedure fails when the correspondences are not correctly identified, e.g., blurred or shaky pictures.

C. Battery consumption

Given the energy constraints of mobile devices, a developer should consider the efficient memory allocation and CPU

TABLE III
QUESTIONNAIRE OF OUR PRELIMINARY SUBJECTIVE STUDY

Question 1	Easiness to get familiar with the Forthroid and use it
Question 2	Experience w.r.t. delay to obtain the information about the point of interest
Question 3	Usefulness of the application in general
Question 4	Easiness to successfully scan QR-codes

TABLE IV
PERCENTAGE OF USERS THAT PROVIDED A SPECIFIC SCORE IN EACH QUESTION

Question 1		**Question 2**	
Trivial	66.7%	Excellent	40%
Very Easy	33.3%	Very Good	33.3%
Satisfactory	0 %	Satisfactory	26.7%
Very Difficult	0%	Not Satisfactory	0%
Too Difficult	0%	Bad	0%
Question 3		**Question 4**	
		Trivial	26.7%
Very Useful	73.3%	Very Easy	40%
Useful	26.7 %	Satisfactory	33.3%
Not That Useful	0%	Very Difficult	0%
		Too Difficult	0%

usage. The battery consumption and the CPU usage of the Forthroid application were evaluated. The following experiment was carried out: During the preview period, the mobile device scans an information-aware QR-code. The battery consumption during the two periods was recorded using the PowerTutor application [15]. The experiment was repeated 90 times. In each run, the battery consumption due to the wireless interface, Organic Light-Emitting Diode (OLED) display, and CPU, was measured separately. The median consumption of the wireless interface is 0.17 mW/ms during the query period (as shown in Fig. 7). Since no wireless communication occurs during the preview period, the wireless interface consumes no battery. The consumption due to the CPU is much higher during the preview period than during the query period, since during preview, the client device *iteratively* tries to detect and decode a QR-code (which contributes to a great amount of CPU load). Note that fine-level conclusions about the battery consumption cannot be easily drawn, since the consumption due to OLED operations depends on the specific content and color displayed on the screen [16]. We also observed that in a few cases the battery consumption due to OLED operations is significantly lower than its median value (as shown in Fig. 7). In those cases, at the query period, the content changes from the specific image captured by the camera to an almost black screen, where the retrieved information is displayed. Moreover, the camera previewing snapshots are still on the display with low intensity. Such cases occur when a user holds the mobile device in such a way that the content changes rapidly, while reading the retrieved information.

The CPU usage of the Android device during the preview and the query period was also measured. For that, we implemented a more lightweight version of the Linux top command which displays a listing of the most CPU intensive tasks of the system. We modified it to measure the CPU usage on intervals shorter than one second (that the original top tool by default uses). During the preview period, the total CPU usage was 89.6%, out of which, the 67% was exclusively by the Forthroid process. During the query period, the total CPU usage was 49.6%, out of which only the 27.7% was utilized by our system. More sophisticated software and hardware mechanisms are required for performing more accurate battery consumption measurements.

D. Subjective study

We performed a preliminary user study to evaluate the user experience running the Forthroid application in the Telecommunications and Networks Laboratory at FORTH. Four information-aware QR-codes were attached on various posters and one action-aware QR-code was placed next to the printer (for the printing service). At first, we briefly presented the functionality of the Forthroid to the 15 colleagues, mostly students of the Institute of Computer Science at FORTH, who agreed to participate in the user study. We then asked them to use the system for at least 10 minutes. At the end, each user answered a questionnaire (shown in Table III) and evaluated his/her experience using the system (the scale/score for each question is shown in Table IV). All users submitted their responses in the questionnaire. The reported scores are shown in Table IV. Most of them reported that it was easy to use the Forthroid and tolerated the delay.

E. Discussion

We faced several challenges while developing the Forthroid on the Android platform related to the camera API, the screen, and GUI. During the preview period, the screen of the mobile device displays the area viewed by the camera. The preview resolution may vary from 176x144 pixels to 1280x720 pixels and differs from the picture resolution, which is the resolution to which a picture is taken by the camera, and varies from 640x480 pixels to 2592x1952 pixels. The preview and picture resolutions depend on the specific model of the device. The relatively low preview resolutions constraint the application, since camera calibration procedures require high resolution images in order to provide accurate results. We overcame this problem by capturing two pictures, one in the preview resolution for the QR-code decoding, and one in the picture resolution for the distance estimation. When a QR-code is detected, the application invokes the *takePicture()* method of the Camera API in order to capture a picture with the highest resolution available from the picture resolutions. As in the case of most portable mobile devices, Android smart phones have relatively small screens and, inevitably, application developers have to fit the graphical content on them. Moreover, in order to make the GUI user-friendly, we tried to keep it simple and provided a help button in the menu to guide users.

V. Conclusion and Future Work

This work focused on the Forthroid, a location-based information retrieval system, and analyzed the delay, the distance estimation error, and the battery consumption of the system. The performance analysis indicates that the heavyweight image processing functions result in a significant increase of the total delay. Moreover, the network delay is also considerable due to the overheard for transmitting large size images. On the contrary, the efficient hardware and software of the HTC Nexus One smart phone result in relatively small processing delays. A significant amount of the total CPU is "utilized" by the preview period, while the corresponding CPU load during the query period is smaller. However, during the query period, the wireless interface and the OLED become the main sources of energy spendings.

The preliminary subjective study indicates that such systems can be useful and encourages us to enhance its functionality and further analyze its performance. We plan to extend the Forthroid to support more services, e.g., enable the retrieved information to be sent via e-mail, be posted on a social networking page and on Google Maps. In addition, the Forthroid will be integrated with the PhotoJournal to create various multimedia journals of various visits [6]. We speculate that the integration of the Forthroid with Facebook and other social networking applications will enrich the online social networking experience of users. Also, we will improve the scanning/decoding of QR-codes process by providing additional (visual or auditory) "messages/clues" to users (which was also provided as feedback in the user study). Finally the subjective study will be extended to a larger and more diverse user population, for a longer period of time, and in various premises, in order to have more conclusive results about the quality of user experience.

Acknowledgment

The authors would like to thank Manolis Surligas for his help during the implementation of the lightweight top command for the evaluation of the CPU usage. Thanks also goes to Google for their equipment grant of Android smart phones (Google AndroidEDU programme), and the FORTH-ICS and KTH ACCESS for their support.

References

[1] J. Rekimoto and Y. Ayatsuka, "Cybercode: designing augmented reality environments with visual tags," in *Proceedings of DARE 2000 on Designing augmented reality environments*, Elsinore, Denmark, April 2000.

[2] D. Lopez de Ipina, P. R. S. Mendona, and A. Hopper, "Trip: A low-cost vision-based location system for ubiquitous computing," *Personal and Ubiquitous Computing*, pp. 206–219, May 2002.

[3] P. Ljungstrand, J. Redström, and L. E. Holmquist, "Webstickers: using physical tokens to access, manage and share bookmarks to the web," in *Proceedings of DARE 2000 on Designing augmented reality environments*, Elsinore, Denmark, 2000.

[4] E. Rukzio, A. Schmidt, and H. Hussmann, "Physical posters as gateways to context-aware services for mobile devices," *IEEE Workshop on Mobile Computing Systems and Applications*, pp. 10–19, December 2004.

[5] M. Rohs and B. Gfeller, "Using camera-equipped mobile phones for interacting with real-world objects," in *Advances in Pervasive Computing*, Linz/Vienna, Austria, April 2004.

[6] N. Kotilainen and M. Papadopouli, "You've got photos! the design and evaluation of a location-based media-sharing application," in *4th International Mobile Multimedia Communications Conference (Mobimedia)*, Oulu, Finland, July 2008.

[7] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic, "People, places, things: web presence for the real world," *Mob. Netw. Appl.*, pp. 365–376, October 2002.

[8] M. Maidi, J. Y. Didier, F. Ababsa, and M. Mallem, "A performance study for camera pose estimation using visual marker based tracking," *Machine Vision and Applications*, pp. 365–376, 2010.

[9] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer-Verlag New York Inc, 2010.

[10] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[11] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997.

[12] J. Y. Bouguet, "Camera calibration toolbox for Matlab." [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[13] C. Harris and M. Stephens, "A Combined Corner and Edge Detection," in *Proceedings of The Fourth Alvey Vision Conference*, Manchester, 1988.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[15] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, Scottsdale, AZ, USA, 2010.

[16] M. Dong, Y.-S. K. Choi, and L. Zhong, "Power-saving color transformation of mobile graphical user interfaces on oled-based displays," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, San Fancisco, CA, USA, 2009.