An Analysis of Navigation Algorithms for Smartphones Using J2ME

André C. Santos[1,3], Luís Tarrataca[1,3], and João M.P. Cardoso[2,3]

[1] IST - Technical University of Lisbon,
Avenida Prof. Dr. Cavaco Silva, 2780-990 Porto Salvo, Portugal
[2] University of Porto, Faculty of Engineering, Department of Informatics Engineering
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[3] INESC-ID, Taguspark,
Avenida Prof. Dr. Cavaco Silva, 2780-990 Porto Salvo, Portugal

Abstract. Embedded systems are considered one of the most potential areas for future innovations. Two embedded fields that will most certainly take a primary role in future innovations are mobile robotics and mobile computing. Mobile robots and smartphones are growing in number and functionalities, becoming a presence in our daily life. In this paper, we study the current feasibility of a smartphone to execute navigation algorithms. As a test case, we use a smartphone to control an autonomous mobile robot. We tested three navigation problems: Mapping, Localization and Path Planning. For each of these problems, an algorithm has been chosen, developed in J2ME, and tested on the field. Results show the current mobile Java capacity for executing computationally demanding algorithms and reveal the real possibility of using smartphones for autonomous navigation.

Keywords: Embedded computing, navigation algorithms, visual landmark recognition, particle filter, potential fields, mobile robotics, smartphones, J2ME.

1 Introduction

The potential of autonomous navigation is an important aspect for mobile robotics and for mobile devices with the goal of helping the user to navigate in certain environments. A mobile device such as a smartphone could be used to guide the user in museums, shopping centers, exhibitions, city tours, and emergency scenarios when a catastrophe occurs; to control more effectively home appliances like vacuum cleaners; to assist impaired people, etc.

However, to be efficient and effective, most navigation problems require computationally demanding algorithms. Bearing in mind the previous applications, this paper presents a performance study of three navigation algorithms when implemented using J2ME technology for mobile devices. To test those algorithms on the field, we use a system composed by a mobile robot and two smartphones. In this system, a smartphone executes the navigation algorithms and sends control

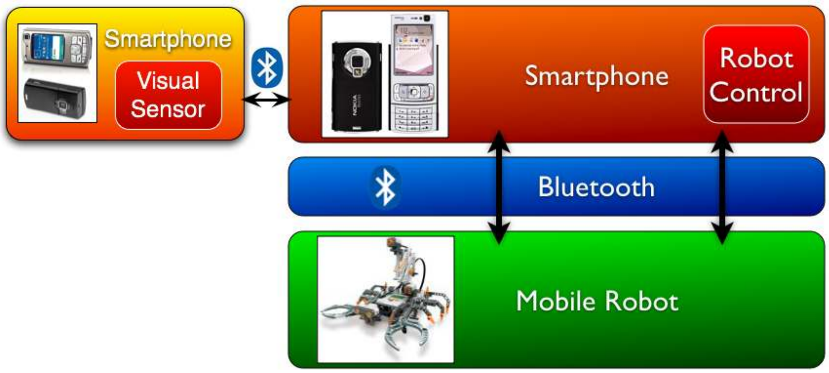

Fig. 1. System organization

instructions to the mobile robot using *Bluetooth*[1] (see the system organization presented in Figure 1). A second smartphone acts as an intelligent visual sensor, communicating processed visual information to the former smartphone.

By developing and studying the J2ME implementation of navigation algorithms on smartphones, we hope to be contributing to a clear understanding about the current capabilities of high-end smartphones and J2ME, and possibly to highlight future improvements on both.

This paper is organized as follows: section 2 gives an overview of navigation algorithms; section 3 presents the algorithms implemented and the experimental setup used; section 4 shows experimental results and section 5 presents some conclusions.

2 Autonomous Navigation

Autonomous navigation has been widely focused by the mobile robotics area [13]. Navigation is defined as the process or activity of accurately ascertaining one's position, planning and following a route. In robotics, navigation refers to the way a robot finds its way in the environment [13] and is a common necessity and requirement for almost any mobile robot.

Leonard and Durrant-Whyte [11] briefly described the general problem of mobile robot navigation by three questions ("Where am I?", "Where am I going?", and "How do I get there?"), each one addressed for a subcategory: Localization, Mapping and Path Planning.

2.1 Localization - "Where Am I?"

Localization is the process of estimating where the robot is, relatively to some model of the environment, using whatever sensor measurements are available. As the robot keeps moving, the estimation of its position drifts and changes,

[1] The Official Bluetooth Technology Info Site (www.bluetooth.com/bluetooth).

and has to be kept updated through active computation [13]. These updates are performed based on the recognition of special features in landmarks, sensor data and probabilistic models.

Localization uncertainty rises from the sensing of the robot, because of the indirect estimation process. The measurements besides being noisy, because of real-world sensor characteristics, may not be available at all times. Based on the uncertainty characteristics of the localization problem, similarly to other important mobile robotics problems, it has been tackled by probabilistic methods [20]. Amongst the most commonly used are Markov Localization [4] and Particle Filters [6].

2.2 Mapping - "Where Am I Going?"

The mapping problem exists when the robot does not have a map of its environment and incrementally builds one as it navigates. While in movement, the robot senses the environment, identifying key features which will allow it to register information of its surroundings. The main concern for the mapping problem is how the mobile robot does perceive the environment. There are many sensors used for mapping, being the most common sonar, digital cameras and range lasers. The complexity of the mapping problem is the result of a different number of factors [20], the most important of which are: size of the environment, noise in perception, and actuation and perceptual ambiguity.

Approaches for mapping have been accomplished considering the extraction of natural features from the environment (see [12]) and through the identification of special artificial landmarks (see, e.g., [16] and [1]).

2.3 Path Planning - "How Do I Get There?"

Path Planning is the process of looking ahead at the outcomes of possible actions, and searching for the best sequence that will drive the robot to a desired goal/location [13]. It involves finding a path from the robot's current location to the destination. The cost of planning is proportional to the size and complexity of the environment. The bigger the distance and the number of obstacles, the higher the cost of the overall planning. Path Planning techniques for navigation can be divided in local path planning and global path planning, which differ on the quantity of information of the environment they need to possess. Local techniques only need information of the environment that is near to the robot, while global techniques use full information of the environment.

There are many different approaches to path planning. A relevant Path Planning technique is the Artificial Potential Field [8].

3 Prototype and Navigation Algorithms Considered

The block diagram of the system is shown in Figure 1. A middleware component is responsible for the interaction between the smartphones and the mobile robot.

The navigation algorithms are executed in the smartphone, and the control instructions are passed to the robot via the middleware. Raw data from sensing by the robot are acquired by the middleware via Bluetooth interface.

3.1 Prototype

The prototype mobile robot consists of a Lego Mindstorms NXT[2] kit coupled with a smartphone (we have used the Nokia N80[3] and the Nokia N95[4])). The smartphone is positioned so its built-in camera faces the front of the robot, enabling it to act as an intelligent image sensor (Figure 2), which furnishes sensing meta-data to the main navigation system (a smartphone responsible for executing the navigation algorithms).

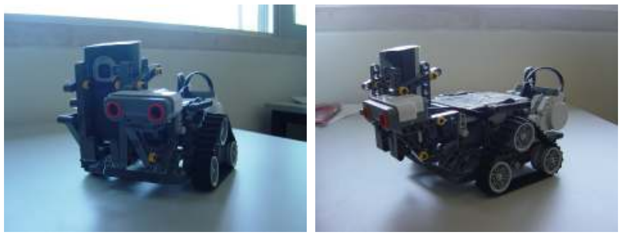

Fig. 2. Prototype mobile robot with a smartphone

Development for the smartphone was done in Java using its Micro Edition version (J2ME[5]). The choice of J2ME development was mainly due to Java's known portability among the most common mobile phone manufacturers. Development for the mobile robot was also done using a subset of Java supported by the JVM present in the custom firmware for the Lego's NXT Brick known as leJOS NXJ [19].

3.2 NXT Middleware

In order to provide seamless integration within the system, we developed a middleware component which helps J2ME application development for the smartphone to communicate with the NXT mobile robot. The core functionality of the middleware consists in providing abstractions for Bluetooth communication and also access to the mobile robot's sensors and actuators. The middleware component was developed in the Java programming language and was built on top of the leJOS NXJ firmware [19].

[2] The Lego Group (http://mindstorms.lego.com/)
[3] Nokia N Series N80 smartphone (http://www.nseries.com/products/n80/)
[4] Nokia N Series N95 smartphone (http://www.nseries.com/products/n95/
[5] Sun Microsystems - Java ME Technology (http://java.sun.com/javame/technology/)

3.3 Visual Landmark Recognition

For real-time mapping we rely on feature extraction by the visual sensor. With the objective of keeping the detection and recognition of the landmarks as fast as possible, the approach implemented in this work uses solid-color cylindrical artificial landmarks. The approach developed is similar to the method by [2]. In our approach, the visual system detects one landmark, recognizes its color, and calculates its distance and orientation to the visual sensor.

Color Segmentation represents the first step for detecting the landmark on a captured image by the smartphone. Previously, the landmark color features were gathered and analyzed, providing the means to empirically produce a set of rules in the RGB color space for detection of the colors used in the artificial landmarks. These rules detect the presence of a landmark in an image thus providing the corresponding landmark classification based on its color. E.g., for a green landmark, we used the rules presented in (1). R, G and B correspond to the red, green and blue color components of the RGB color space. The value X is an adjustment value, that is used to augment the green color component relatively to the red and blue.

$$(G \geq 130) \text{ and} (G > R + X) \text{ and } (G > B + X) \tag{1}$$

The color segmentation process transforms the captured image into a binary, black and white image as can be seen in Figure 3. White color pixels indicate the presence of the green range color and black pixels the absence of it.

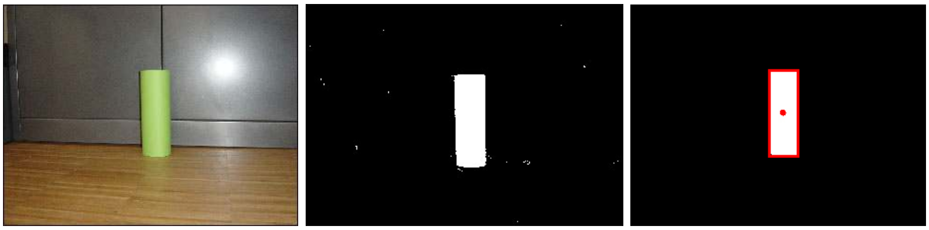

Fig. 3. Image captured with a green landmark (left image); Binary image after the application of the color segmentation (middle image) and landmark boundary detection after the application of the image noise reduction filter (right image)

Image Noise Reduction is necessary for the elimination of salt and pepper noise, caused by the color segmentation process. The noise may compromise better results in future steps and therefore needs to be reduced or removed. The filter implemented uses a 3×3 scanning window, that analyzes all the landmark pixels present in the image. The window checks if the pixels surrounding the current scanned pixel mostly belong to the landmark or the background. If they are mostly background ($\geq 50\%$) then the pixel is most likely noise and is erased (see Figure 3).

Landmark Boundary Detection is needed for more accurate calculations of distance and orientation. A minimum rectangular boundary contains the shape detected and is used to help cope with some small variations in the shape's perspective, that can vary according to the view from which the image was acquired.

Distance ($\boldsymbol{d}$) and Orientation ($\boldsymbol{\theta}$) from the visual sensor to the landmark is calculated based on one of the methods in [22]. By knowing the width, height and center point of the landmark and having already performed measurements for camera calibration, the distance and orientation information can be inferred from the landmark's size and position in the image (see (2) and (3)).

$$d_y = k_y \times \frac{1}{y'} \qquad d_x = k_x \times \frac{1}{x'} \qquad d = \frac{d_x + d_y}{2} \tag{2}$$

$$\theta = m \times x_{LandmarkCenter} + l \tag{3}$$

3.4 Particle Filter

For localization, we implemented the Particle Filter method, based on the approach presented in [17]. The environment is represented as an occupancy grid map, where each grid cell matches an area of the real environment at a specific ratio. Each grid cell can be assigned with estimation probabilities of the mobile robot's position or with a reference to the presence of an obstacle. The Particle Filter method can be divided into three main stages: **Prediction** which involves a motion and noise model for movement; **Update**, which concentrates on sensing the environment and altering the particles relevance weight value; and a **Resample** stage where the particle population is managed.

Motion Model is the robot's path planner, which is responsible for providing at each step a path for the mobile robot's movement. In this work two motion models were developed: an **explorer type motion model** which visits all free locations in the map and a **point to point motion model** which is a predefined obstacle-free path from one location in the environment to another.

Noise Model is responsible for the odometry error that is added to the robot's motion, based on the noise model provided in [17]. The odometry error considered was divided into rotation error and translation error. Both were experimentally established from the real odometry errors from the robotics kit used. Translation and rotation with noise are accomplished using a pseudo-random value, drawn as a sample from the Gaussian distribution.

Measurement Model provides, on each observation of the environment, necessary information for the **weighting function** which will update the particle's weights. In this implementation, the particle's weight is considered to be a numeric value w greater than 0. An observation consists on sensing the environment. Sensing is done by using a simulated observation from the information present in the internal map or by using the visual landmark recognition method presented earlier.

Resampling occurs when a considerable amount of particles within the particle population have weight values below a threshold and therefore have low contribution to the overall estimate of the robot's position. The resampling process recognizes particles with small weight values ($< threshold$) and replaces them with a random particle, whose weight value is higher than the resampling threshold ($\geq threshold$). This random replacement minimizes the problem of diversity loss. When all particles have weights bellow the *threshold* then a new random set of particles is generated.

Robot Position Estimate at a determined time t, is given by the **best particle** which has the maximum weight value within the current particle set.

3.5 Potential Fields

For path planning we use the Potential Fields approach [8] which is used for path planning and collision avoidance due to its mathematical simplicity and elegance, providing acceptable and quick results [10] in real-time navigation. This method is based upon the concept of attractive and repulsive forces, where the goal is seen as a global minimum potential value (attractive force), and all obstacles as high valued potential fields (repulsive force). The movement of the robot is then defined by the potential values present in its path, moving ideally from high to low potentials.

Our approach uses as basis the potential field functions presented by [5]. The most difficult problem for the Potential Field method, known as the local minima has been addressed using escape techniques (e.g., Random Escape, Perpendicular Vector Escape [21], Virtual Obstacle Concept Escape [15]). In order to provide a smoother robot movement, a lookahead function was implemented which prevents the mobile robot from falling into local minima locations by detecting them in advance.

4 Experimental Results

In this section, we present and discuss experimental results for the navigation problems: Mapping, Localization, and Path Planning. Here we evaluate the performance of the algorithms developed, by comparing executions between the used smartphones and a desktop PC (AMD Athlon 64 X2 Dual Core Processor at 2.20 GHz with 1GB of RAM and running Windows XP SP3), and analyzing the feasibility of smartphones for real-time autonomous navigation.

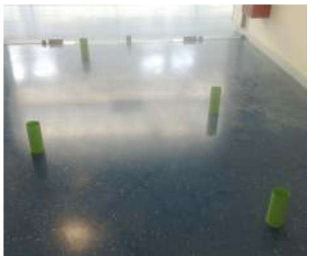

Fig. 4. Field environment for testing the navigation algorithms

A first study of performance was done with profiling results gathered from a PC MIDP emulator (Sun Java Wireless Toolkit[6]). Then, we conducted several experiments on the field. Figure 4 shows the experimental environment where field testing took place.

4.1 Mapping

Experiments with mapping test the application of the Visual Landmark Recognition method while trying to map the environment present in Figure 4. Tests executed indicated good identification of the landmarks colors, being illumination changes the main source of the incorrect identifications.

Using a single captured image and considering a good landmark detection and color segmentation process, the distance calculation revealed quite accurate presenting an average relative error of 5.715%. Considering the angle orientation measurement, it revealed reasonably accurate with an average relative error of 10.06%.

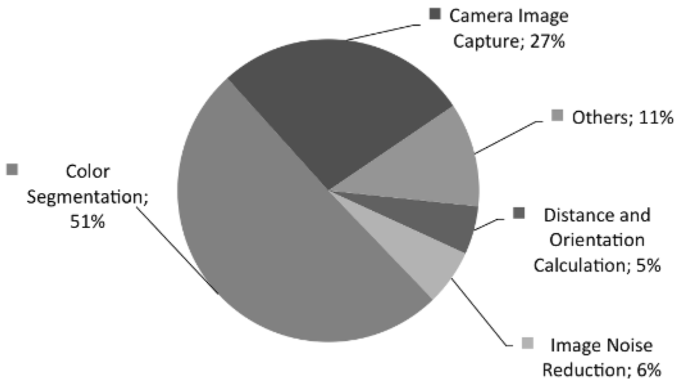

Fig. 5. Contribution to the overall execution time of each step associated to the Visual Landmark Recognition algorithm

[6] Sun Java Wireless Toolkit (http://java.sun.com/products/sjwtoolkit/)

Table 1. Execution time measurements for the Visual Landmark Recognition method

	PC	Nokia N80	Nokia N95
Execution time (ms)	453.00	3079.40	5824.30

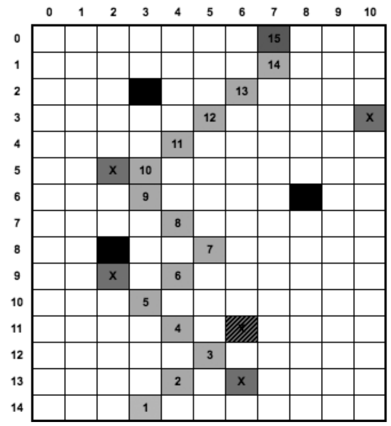

Fig. 6. Mapping results with estimated landmark positions

Figure 5 presents profiling results for capturing an image and applying the landmark recognition algorithms. Table 1 compares the execution time obtained when running the algorithms on the PC, on a Nokia N80, and on a Nokia N95.

Obviously, the PC is the fastest to execute the application. Comparing the two smartphones, execution time is slower in the Nokia N95 compared to the N80. The N95 has a more complex built-in camera with higher resolution, making it slower when capturing an image with J2ME.

When testing on-the-field, the mapping approach revealed less accurate than expected. The robot's movement and variable lighting conditions prevent the method from achieving its best results. Although this solution cannot be considered a very reliable method for accurate mapping purposes in real-time mobile robot navigation, it presents typical mapping tasks and it is used here as a benchmark for studying the performance obtained by the two smartphones used.

Figure 6 shows the achieved mapping accuracy using Visual Landmark Recognition on the environment presented in Figure 4. The grid presents the obstacles as black colored cells, obstacle estimates calculated in gray colored cells marked with an "X" character, and the path taken by the mobile robot is presented in lighter gray and marked with numbers.

4.2 Localization

Experiments for Localization were conducted considering only a global localization approach based on the Particle Filter.

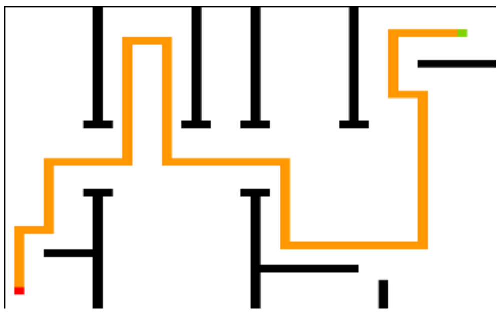

Fig. 7. Occupancy grid map for Particle Filter

For profiling the Particle Filter implementation, we considered one execution of the method with a total number of 1,000 particles and using the environment in Figure 7. We consider that the robot position estimation is only performed at the end of the mobile robot's movement. Figure 8 presents the percentages of execution time of the main phases of the Particle Filter method. Table 2 presents the execution time comparison between running the implemented localization application on a PC, on a Nokia N80, and on a Nokia N95 smartphone.

According to the experiments, the phase which was responsible for the highest percentage of execution time was the Prediction Phase with 48%. The Update phase followed with 41%. Finally, and considering the number of particles used and their distribution within the environment, the Resample Phase took 9% of the total execution time. The last 2% of execution time is spent by auxiliary tasks and by the attainment of the pose estimate.

Our next experience uses the Particle Filter method to localize the mobile robot in the environment presented in Figure 4. The Localization approach is implemented as a distributed system, were the Particle Filter approach is executed on a Nokia's N95 smartphone, considering 1,000 particles; and the measurement model as a visual sensor performed with the Visual Landmark Recognition method running on the Nokia N80 model.

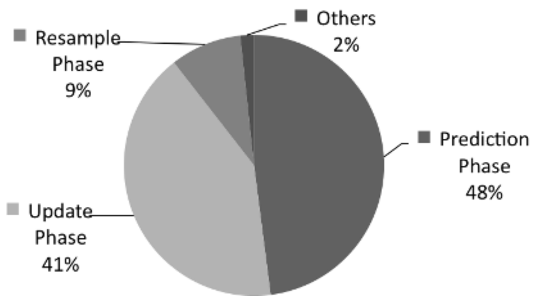

Fig. 8. Contribution to the overall execution time of each phase of the Particle Filter

Table 2. Execution time measurements for the Particle Filter method

	PC	Nokia N80	Nokia N95
Time (ms)	78.00	3618.00	1725.00

Results for five executions of this field experiment are presented in Table 3. Consider that the positions are given as xy position and θ orientation: $[x; y; \theta]$. The robot's real position at the end of the predefined path is $[7; 0; 90°]$. By analyzing Table 3, we can observe that only one of the experiments estimated the robot to be at its exact physical location. In the other four experiments, three were relatively close to the robot's real position, and the last one was very far from the robot's position.

Table 3. Estimations for the same real position ($[7; 0; 90°]$) for tests on-the-field using the Particle Filter method

Experiment	#1	#2	#3	#4	#5
Best Particle Position	$[4; 0; 90°]$	$[7; 0; 90°]$	$[9; 1; 90°]$	$[4; 0; 90°]$	$[0; 11; 180°]$

The random initialization of the particles makes the method difficult to predict, by providing very different results on different runs of the algorithm since areas in the environment can be highly populated with particles while others deprived from them (see experiments #1 to #5 in Table 3). One possible solution to this problem is the increase of the number of particles, but with high additional computational costs.

The visual sensing in the particle filter execution is time demanding and cannot, without further optimizations, be used to navigate mobile robots at high speed. Nevertheless, considering a slower motion, this solution was able to provide a mechanism for mobile robot localization.

4.3 Path Planning

The next experiments analyze the Potential Fields. Figure 9 illustrates the main stages of the algorithm and their contribution to the overall execution time. For this particular implementation, we used a lookahead value of 5 for local minima detection. As can be seen, this preemptive detection is responsible for about 80% of the overall execution time, while the potential calculations for the effective next movement occupies around 20% of the total time.

Table 4 shows the execution time for the path presented in Figure 10. The PC presents the lowest execution time, and the Nokia N95 has faster execution than the Nokia N80. These results were expected. Due to the mathematical requirements of the algorithm, it is able to execute faster in the N95, as it includes a more powerful main microprocessor.

When performing experiments on-the-field, the robot revealed some strange orientation changes when avoiding obstacles. This fact was never very noticeable

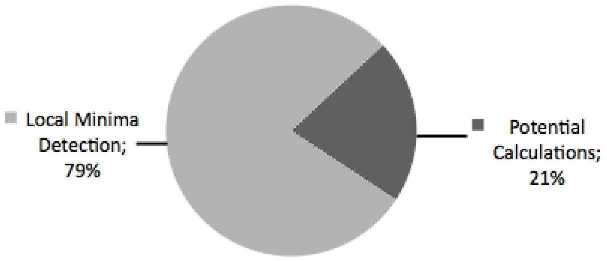

Fig. 9. Contribution to the overall execution time of each stage of the Potential Fields

Table 4. Time and Memory measurements for the Potential Fields algorithm

	PC	Nokia N80	Nokia N95
Average Step Time (ms)	11.00	377.00	278.75
Total Time (ms)	7664.60	253442.75	187882.75

in the simulations performed. We concluded that, even in the absence of local minima locations, some raw directional vectors cannot be directly applied for the robot's movement. Some of these directional vectors force the robot to perform expensive rotations that need to be smoothen beforehand.

Globally, the experiments revealed a considerable robustness of the current JVM available in the mobile devices used and the potential for those devices to execute complex navigation algorithms. However, the current J2ME platform makes it difficult to provide more efficient implementations of the algorithms used, which can be seen by the execution times presented. Nevertheless, their execution is possible and there is still room for further improvements.

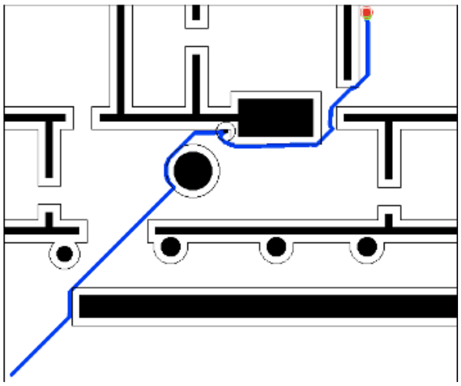

Fig. 10. Complex environment used for Potential Fields profiling

5 Conclusions

The work presented in this paper focused on a study of the viability to accomplish autonomous navigation with smartphones and J2ME. Tests with well known navigation algorithms (e.g., potential fields and particle filter) have been performed. To achieve realistic experiments, we use a mobile robot controlled by a smartphone, able to execute complex and computationally intensive navigation algorithms and to communicate with the robot via Bluetooth.

The mobile implementation of the algorithms revealed high consistency and robustness. The experiments on-the-field show that it is feasible to execute real-time navigation algorithms without too much tight constraints in high-end smartphones. Note, however, that the current processing capabilities of smartphones and J2ME can fully fulfill real-time requirements in environments where the smartphone might be used to assist the user (e.g., navigating in a city, shopping center).

From the experiments performed for visual landmark recognition, it is clear that future enhancements of J2ME should include the capability to acquire video streaming and to access individual frames. The current implementation needs to perform single image capture, which is too slow for real-time video processing.

Acknowledgments

We would like to acknowledge the donations of smartphones by Nokia, Finland.

References

1. Baczyk, R., Kasinski, A., Skrzypczynski, P.: Vision-based mobile robot localization with simple artificial landmarks. In: Prepr. 7th IFAC Symp. on Robot Control, pp. 217–222 (2003)
2. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color image segmentation for interactive robots. In: Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), vol. 3, pp. 2061–2066 (2000)
3. Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy (1992)
4. Fox, D.: Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation. PhD thesis, Institute of Computer Science III, University of Bonn, Germany (December 1998)
5. Goodrich, M.A.: Potential fields tutorial. Class Notes (2002)
6. Gordon, N., Salmond, D., Smith, A.: Novel approach to nonlinear/non-gaussian bayesian state estimation. IEEE Proceedings for Radar and Signal Processing 140(2), 107–113 (1993)
7. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering 82(Series D), 35–45 (1960)
8. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 5(1), 90–98 (1986)

9. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. The American Association for the Advancement of Science 220(4598), 671–680 (1983)
10. Lee, L.-F.: Decentralized motion planning within an artificial potential framework (apf) for cooperative payload transport by multi-robot collectives. Master's thesis, Department of Mechanical and Aerospace Engineering (December 2004)
11. Leonard, J.J., Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation 7(3), 376–382 (1991)
12. Lowe, D.G.: Object recognition from local scale-invariant features. In: The Proceedings of the 7^{th} IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
13. Matarić, M.J.: The Robotics Primer, 1st edn. MIT Press, Cambridge (2007)
14. Negenborn, R.: Robot localization and kalman filters. Master's thesis, Utrecht University, Netherlands (September 2003)
15. Park, M.G., Lee, M.C.: Artificial potential field based path planning for mobile robots using a virtual obstacle concept. In: Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003), vol. 2, pp. 735–740 (July 2003)
16. Prasser, D., Wyeth, G.: Probabilistic visual recognition of artificial landmarks for simultaneous localization and mapping. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation, vol. 1, pp. 1291–1296 (September 2003)
17. Rekleitis, I.: Cooperative Localization and Multi-Robot Exploration. PhD thesis, School of Computer Science, McGill University, Montréal (January 2003)
18. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In: Autonomous robot vehicles, pp. 167–193. Springer, New York (1990)
19. Solórzano, J., Bagnall, B., Stuber, J., Andrews, P.: lejos: Java for lego mindstorms, `http://lejos.sourceforge.net/` (last visited in June 2008)
20. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
21. Veelaert, P., Bogaerts, W.: Ultrasonic potential field sensor for obstacle avoidance. IEEE Transactions on Robotics and Automation 15(4), 774–779 (1999)
22. Yoon, K.-J., Jang, G.-J., Kim, S.-H., Kweon, I.S.: Fast landmark tracking and localization algorithm for the mobile robot self-localization. In: IFAC Workshop on Mobile Robot Technology, pp. 190–195 (2001)