Autonomous Mobile Robot Navigation using Smartphones
Extended Abstract

André Guilherme Nogueira Coelho dos Santos
Instituto Superior Técnico

November 2008

Abstract

Robots are present everywhere, from lawn mowers and pool cleaners to space vehicles. Within robotics, special attention is given to mobile robots, since they have the capacity to navigate in their environment.

Mobile phones are today another reality that none of us live without. From immensely limited devices, to small, cheap and rich smartphone platforms, mobile phones are today one of the most used embedded systems in the world. A smartphone is a fully functional computer with added communication capacity, built-in video camera and media players, just to name a few.

With the proliferation of embedded systems in our lives, we decided to study how an integration between mobile robots and smartphones could contribute to the tasks that these robots perform today. We studied the current feasibility of a smartphone executing navigation algorithms in order to control mobile robots. We developed such a mobile system where we tested the three main navigation problems, Mapping, Localization and Path Planning.

Keywords: Visual landmark recognition, particle filter, potential fields, mobile robotics, smartphone, J2ME.

1. Introduction

This paper presents the development of a navigation system composed by a mobile robot and a smartphone. In this system, the mobile robot is controlled by the smartphone, where navigation algorithms are processed, generating controls that are transmitted back to the mobile robot using bluetooth communication (see the system organization presented in Figure 1).

Our work established the following main objectives:

- Develop a prototype considering the system architecture provided in Figure 1;
- Research and develop navigation algorithms for mobile robots considering their implementation on embedded systems;
- Study and analyze smartphone feasibility, capacity and performance when executing computationally expensive algorithms (and also considering image capture as a visual sensor).

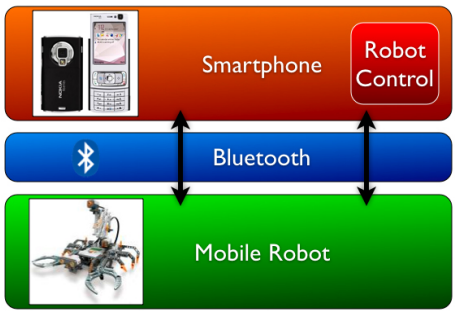

Figure 1: System organization.

We believe that the use of a smartphone to control a robot has many real life applications. In this work we contribute with the study of integration solutions for joint systems of mobile robots and smartphones; as well as the implementation of computationally demanding navigation algorithms in limited embedded devices.

This paper is organized as follows: section 2 gives an overview of navigation in mobile robotics; section 3 presents our system, with the definition of the prototype developed and algorithms implemented; section 4 shows experimental results and section 5 presents some conclusions and future work.

2. Mobile Robotics Navigation

A robot is an autonomous system that is able to sense its environment, and to act on it to achieve goals. A mobile robot adds the fact that it is not confined to one

specific location, as it has the capability of moving in its environment. The main characteristic that defines an autonomous robot is the ability to act on the basis of its own decisions, and not through the control of a human [14].

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 [14] and is a common necessity and requirement for almost any mobile robot.

Robot navigation is a vast field and can be divided into subcategories for better understanding of the problems it addresses. Leonard and Durrant-Whyte [12] briefly described the general problem of mobile robot navigation by three questions, 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, and has to be kept updated through active computation [14]. These updates are performed based on the recognition of special features in landmarks, sensor data and probabilistic models.

Robot localization is a key problem in making truly autonomous robots. If a robot does not know where it is, it can be difficult to determine what to do next. In order to localize itself, a robot has to access relative and absolute measurements which return feedback about its driving actions and the situation of the environment. Given this information, the robot has to determine its location as accurately as possible [15]. Uncertainty rises from the sensing of the robot because the estimation process is indirect, the measurements are noisy and problematic because of real-world sensors and measurements may not be available at all times.

Based on the uncertainty characteristics of the problem, localization, as other important mobile robotics problems, has been tackled by probabilistic methods [24]. 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 does the mobile robot perceive the environment. To locate obstacles, detect distances, observe landmarks, etc., the mobile robot must have a sensing mechanism that enables measurement collection. The sensors used for mapping depend on the type of mapping that needs to be done. Most common sensors are sonar, digital cameras and range lasers. Approaches for mapping have been accomplished considering the extraction of natural features from the environment (see [13]) and through the identification of special artificial landmarks (see, e.g., [19] and [1]).

The complexity of the mapping problem is the result of a different number of factors [24], the most important of which are: Size of the environment; noise in perception and actuation; perceptual ambiguity and cycles.

2.3. Path Planning - "How do I get there?"

Path Planning is the process of looking ahead at the outcomes of the possible actions, and searching for the sequence of actions that will drive the robot to the desired goal [14]. 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 to the overall planning. The cost of planning is a very important issue for real-time navigation needs, as the longer it takes to plan, the longer it takes to find a solution. Path Planning techniques for navigation can be divided into two subcategories:

- **Local Path Planning** are solutions that do not imply much complexity since they use only local information of the environment. They often do not offer optimal solutions and also have the common problem of local minima.

- **Global Path Planning** takes into account all the information of the environment at the same time. Unfortunately this type of planning is not appropriate for real-time obstacle avoidance because of the high processing needs for all the environment's data.

There are many different approaches to path planning which try to solve the problem using different techniques. Two relevant Path Planning techniques are the Artificial Potential Field [8] and approaches based on the Ant Colony [3].

2.4. SLAM

SLAM (Simultaneous Localization and Mapping) [21] is a technique that involves both localization and mapping. It is used by robots and autonomous vehicles to build up a map within an unknown environment while at the same time keeping track of their current position.

This approach is complex since it involves both localization and mapping processes both with uncertainties associated. One main concern in SLAM is keeping the uncertainty, for both robot position and landmark position, controlled, thus keeping errors as low as possible. For this double uncertainty, SLAM usually uses Kalman Filter [7] and Particle Filter [6] methods.

3. Approach

The block diagram of the system is shown in Figure 1. There is a middleware component responsible for the interaction between the smartphones and the mobile robot. The navigation algorithms are executed in the smartphone, and the control orders are passed to the robot via the middleware.

3.1. Prototype

As mentioned before the prototype developed consists on a mobile robot (Lego Mindstorms NXT kit [11]) able to navigate through the environment with an attached smartphone (Nokia N80 [17] or Nokia N95 [16]). 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).

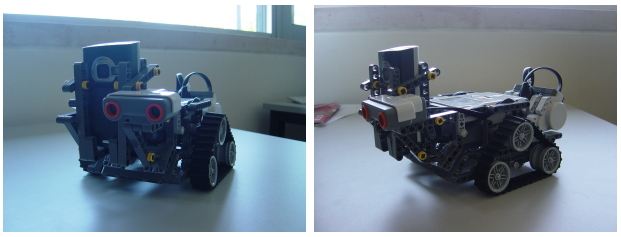

Figure 2: Prototype mobile robot with smartphone.

Development for the smartphone was done in Java using its Micro Edition version (J2ME [23]). Development for the mobile robot is done also using Java with the addition of a custom firmware for the Lego's NXT Brick known as leJOS NXJ [22].

3.2. NXT Middleware

In order to provide seamless integration within the system, we developed a middleware component (Figure 3). The main objective of the middleware component is to facilitate application development for the system composed by the smartphone and the NXT-based 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 (see Table 1). The middleware component was developed in the Java programming language and was built on top of the leJOS NXJ firmware [22].

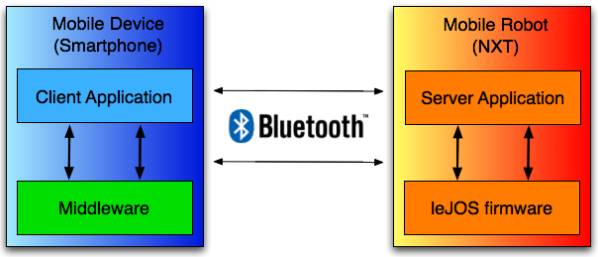

Figure 3: NXT Middleware component schematic.

Methods	
connect()	forward(velocity)
disconnect()	backward(velocity)
getSensors()	stop()
getSensorValue(sensor)	travel(distance)
	rotate(angle)

Table 1: List of current middleware methods.

3.3. Visual Landmark Recognition

For real-time mapping we rely on fast feature finding 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 artificial landmarks. The approach developed is similar to the method by [2]. In our approach, the visual system detects one landmark, and by recognizing features with interest, classifies the landmark and calculates its distance and orientation to the visual sensor.

In this approach, only one landmark is identified per image, which can become a limitation in more complex environments.

3.3.1. Landmark Design

The design of the artificial landmark defines the quality and difficulty of its detection and recognition. We use cylindrical shaped objects with solid color as artificial landmarks (Figure 4). Multiple landmarks need to be distinct from one another, in order to diminish

uncertainty when searching for a specific color, since different colors will have different associations.

Figure 4: Landmark Colors (green, blue and red).

The cylindrical landmarks are made out of A4 cardboard colored paper and have 4 cm in radius and 21 cm in height. The proposed size makes it easy and fast to create landmarks and accomplishes the objective of being easily identifiable in an indoor environment within a considerable range. The cylindrical shape provides a simple visible landmark for the mobile robot, having a rectangular representation from any side viewpoint where the mobile robot observation can take place.

3.3.2. Color Segmentation

The first step for detecting the landmark consists on performing color segmentation on the captured image. Previously, the landmark color features were gathered and analyzed, providing the means of empirically producing a set of rules in the RGB color space for color detection. These rules will detect the presence of a landmark in an image thus providing the corresponding landmark classification.

For the green landmark, we used the rules present 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.

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

In Visual Landmark Recognition, the color segmentation process transforms the captured image into a black and white image as can be seen in Figure 5. White color pixels indicate the presence of the green range color and black pixels the absence of it.

3.3.3. Image Noise Reduction

Salt and pepper image noise may be present because of the color segmentation process. The noise present may compromise better results in future steps and therefore needs to be reduced or removed. The solution relies on the application of an image noise reduction filter.

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 are mostly belong to the

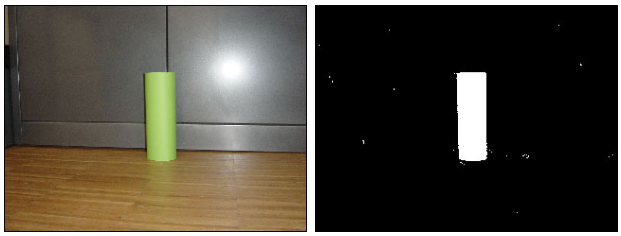

Figure 5: Green Landmark (left image) and Color Segmentation application (right image).

landmark or the background. If they are mostly background ($\geq 50\%$) then the pixel is most likely noise and is erased.

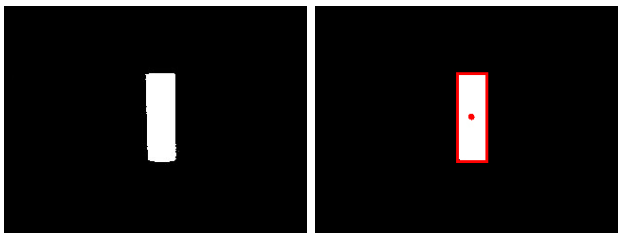

Figure 6: Green landmark with image noise reduction (left image) and with bounding rectangle (right image).

The application of the noise reducing filter implemented can be seen on the left of Figure 6. The filter removes the sparse noise that is present after the color segmentation stage. To the right of Figure 6 the bounding box and center point are marked in red. With the frontiers recognized, size measurements can be more correctly performed.

3.3.4. Minimum Bounding Rectangle

For more accurate calculations of distance and orientation the landmark boundaries are identified. A boundary rectangle 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.

3.3.5. Distance and Orientation

For the calculation of the distance (d) and orientation (θ) from the visual sensor to the landmark we based our equations on one of the proposed methods in [26]. 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.

The distance between the visual sensor and the landmark is inversely proportional to the image's landmark length. For a more correct distance calculation we use both the width and height of the image. They are computed independently and the final distance is given by the average of the two. Equations (2) present the expressions used.

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

For the angle orientation of the landmark we used the Equation (3). The orientation of the landmark regarding the visual sensor is computed using the x position of the center point calculated for the landmark. The x coordinate is used to calculate the angle of the landmark position in the captured image using m and l which are constants derived from the camera calibration.

$$\theta = m \times x_{LandmarkCenter} + l \quad (3)$$

3.4. Particle Filter

The Particle Filter method implemented in this work is based on the approach presented in [20]. The environment is represented as an occupancy grid map, where each grid cell matches an area of the real environment at a specific ratio. Occupancy grid map consists of an environment's 2D representation where 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.

3.4.1. Motion Model

The 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 order to provide the mobile robot with an appropriate movement, so that the Particle Filter method could be properly tested and executed, 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.

3.4.2. Noise model

Odometry noise was added to the robot's motion, modeled as a gaussian distribution, based on the noise model provided in [20]. The possible odometry error considered for the noise 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 is accomplished using a pseudo-random value, drawn as a sample from the guassian distribution.

3.4.3. Measurement Model

The measurement model will provide on each measurement 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.

A measurement consists on an observation from the environment. This observation can be accomplished by using a single straight observation from the information present in the internal map representation or by using the visual landmark recognition method presented earlier.

3.4.4. Resampling

Resampling occurs when a considerable amount of particles within the particle population have weight values below a threshold and therefore have low contribution on the overall estimate of the robot's pose.

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.

3.4.5. Robot Pose Estimate

In order to estimate the mobile robot's position at a determined time t, we chose the **best particle approach**, which has the maximum weight value within the current particle set.

3.5. Potential Fields

The Potential Fields Approach [8] is very used for path planning and collision avoidance due to its mathematical simplicity and elegance. It is simple to implement and can easily provide acceptable and quick results [10] in real-time navigation.

This method is based upon the concept of attractive and repulsive forces. Since the objective of any path planning algorithm is to flee from obstacles and move towards a desired goal, in potential fields, the goal is seen as a global minimum potential value, and all obstacles as high valued potential fields. 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 was addressed using the definition of escape techniques

(e.g., Random Escape, Perpendicular Vector Escape [25], Virtual Obstacle Concept Escape [18], Simulated Annealing Escape [9]).

In order to provide a smoother robot movement, a lookahead function was implemented which prevents the mobile robot from falling into local minima locations having them detected 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, and analyzing the feasibility of smartphones for real-time applications. Performance monitoring with profiling results using a PC MIDP emulator, and field tests are also conducted. Figure 7 shows the testing environment used with green landmarks posing as obstacles where field testing takes place.

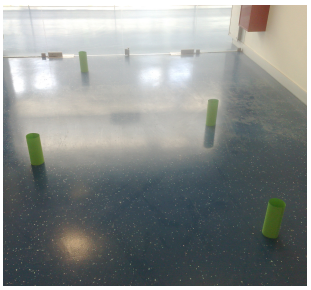

Figure 7: Field environment for testing.

4.1. Mapping

Experiments with mapping test the application of the Visual Landmark Recognition method while trying to map the environment present in Figure 7.

Tests executed indicated good identification of the landmarks colors. Illumination changes were identified as the main problem 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. The average absolute error was 3.6 *cm* and the average relative error was 5.715%. Considering the angle orientation measurement, it revealed also quite accurate with an average absolute error of 2.11° and average relative error of 10.06%.

Figure 8 presents profiling results of the process of capturing an image and applying the landmark recognition algorithms. Table 2 compares the execution's time and memory usage between the PC, Nokia N80 and Nokia N95.

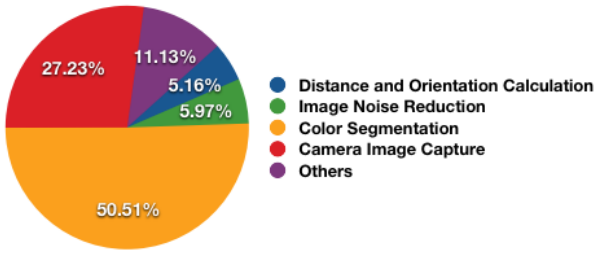

Figure 8: Contribution to the overall execution time of each step associated to the Visual Landmark Recognition.

	PC	Nokia N80	Nokia N95
Time (ms)	453.00	3079.40	5824.30
Memory $(bytes)$	360368.00	163557.60	157840.80

Table 2: Time and Memory measurements for the Visual Landmark Recognition method.

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

Figure 9 shows the achieved mapping accuracy when the Visual Landmark Recognition is executed on the environment presented in Figure 7. The grid presents the obstacles as black colored cells, obstacle estimates calculated in blue colored cells, and the path taken by the mobile robot is presented in orange.

Although the implemented recognition makes use of a simple approach to computer vision, and considering good lighting conditions, the identification of the landmarks were reasonably accurate. In field testing, the method revealed less accurate. The robot's movement and variable lighting conditions prevent the method from achieving its best results. Thus, this solution cannot be considered a very reliable method for accurate mapping purposes in real-time mobile robot navigation.

4.2. Localization

Experimental results for Localization were conducted considering only the global localization approach.

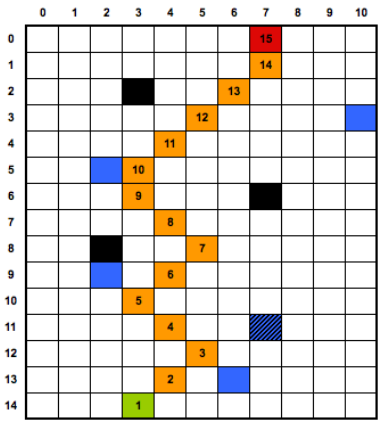

Figure 9: Mapping results with landmark estimated positions.

For profiling the Particle Filter implementation, we considered one execution of the method with a total number of 1000 particles and using the environment in Figure 10. The robot pose estimate is only performed at the end of the mobile robot's movement. Figure 11 presents the percentages of execution time of the main phases of the Particle Filter method. Table 3 presents the execution time and memory used comparison between running the implemented localization application on a standard desktop PC and on the smartphone models Nokia N80 and Nokia N95.

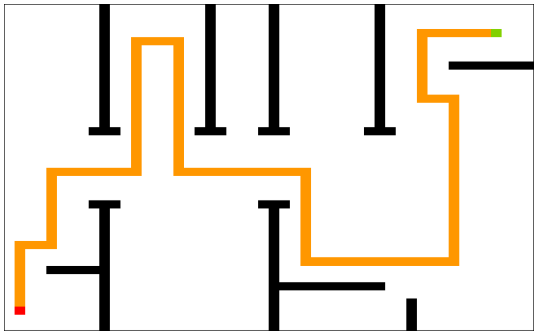

Figure 10: Occupancy grid map for Particle Filter.

	PC	Nokia N80	Nokia N95
Time (ms)	78.00	3618.00	1725.00
Memory $(bytes)$	139200.00	138060.00	78212.00

Table 3: Time and Memory measurements for the Particle Filter method.

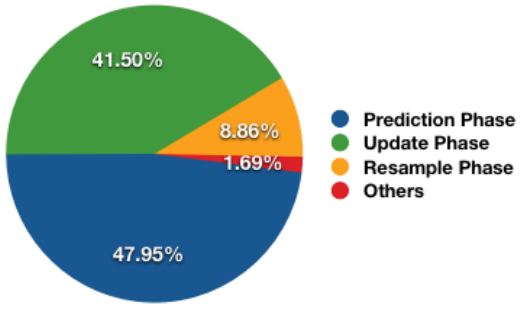

Figure 11: Contribution to the overall execution time of each phase of the Particle Filter method.

According to the experiments, the phase which was responsible for the highest percentage of execution time was the Prediction Phase with almost 48%. The Update phase followed with more than 41%. Finally and considering the number of particles used and their distribution within the environment, the Resample Phase took approximately 9% of execution time.

In the Prediction Phase the particle's movement and noise addition are the main consuming areas. Within the Update Phase, the execution time is dominated by the measurement of the particles. For the Resample Phase, the execution time is dominated with the selection of particles to remove and particles to duplicate. The last 1.69% of execution time is spent by auxiliary tasks and by the attainment of the pose estimate.

Now let's consider testing the Localization using the Particle Filter on the field. Experiments on the field will try to localize the mobile robot in the environment presented in Figure 7. The Localization approach is implemented as a distributed system, were the Particle Filter approach is executed on the Nokia's N95 smartphone, considering 1000 particles, the same amount of particles as the profiling experiment; and the measurement model is a visual sensor performed with the Visual Landmark Recognition method running on the Nokia N80 model.

Results for five executions of this field experiment are presented in Table 4. 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 4, 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.

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

Table 4: Pose estimate for field testing using the Particle Filter method.

The random initialization of the particles makes the method difficult to predict, by providing very different results on different runs of the algorithm (see experiments #1 to #5 in Table 4). For the same map, same measurement model and motion model, the algorithm can provide a very accurate estimate of the mobile robot position or a very inaccurate one. One possible solution to this problem is the increase of the number of particles, but with high additional computational costs.

The solution implemented was feasible to be executed in real-environments, with the limitation of slow movement by the mobile robot. The visual sensor, the particle filter execution and the transmission of information using Bluetooth are 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 12 illustrates the effect of the lookahead function in the overall execution of the method. A lookahead of 5 positions was used to detect the local minima locations. This preemptive detection consumed the greatest amount of execution time.

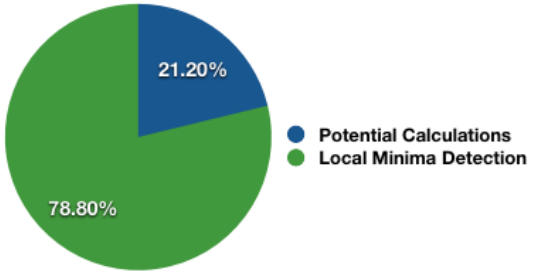

Figure 12: Contribution to the overall execution time of each stage of the Potential Fields.

Table 5 shows the execution times for the simulation in Figure 13. The PC presents the lowest execution time, and on the smartphones, when considering algorithms that simply need to accomplish fast calculations, the Nokia N95 has faster execution times than the Nokia N80 model.

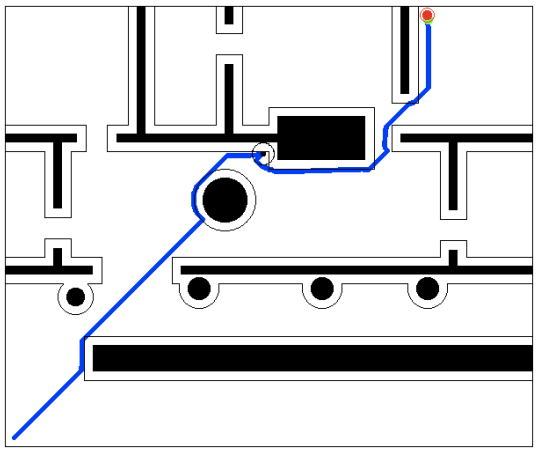

Figure 13: Complex environment used for profiling.

	PC	Nokia N80	Nokia N95
Average Step Time (ms)	11.00	377.00	278.75
Total Time (ms)	7664.60	253442.75	187882.75
Total Memory $(bytes)$	705572.00	108412.00	109437.00

Table 5: Time and Memory measurements for the Potential Fields algorithm.

A simulation of the execution of the Potential Fields method on the environment presented in Figure 7 is shown in Figure 14. The illustrated movement represents the real movement taken by the mobile robot, successfully escaping the obstacles and reaching the target goal.

In the field test, due to the nature of the Potential Fields method, the robot revealed some strange orientation changes when avoiding obstacles. This fact was never very noticeable in simulation testing. 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. For

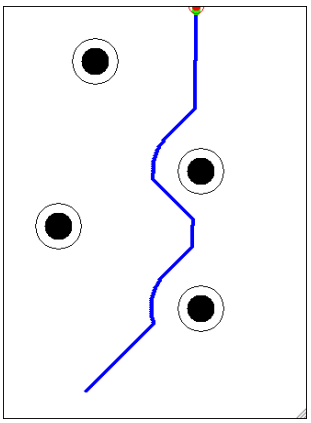

Figure 14: Path Planning simulation test.

field testing, a preprocessing translation model should be applied in order to better convert the output from the potential fields to the mobile robot's movements.

5. Conclusions

Mobile robot navigation is today a main area in the robotics field. It approaches three of the most important problems for autonomous mobile robots: Mapping, Localization and Path Planning. These problems were addressed by this work to research the use and feasibility of a smartphone for mobile robot navigation.

Mapping using artificial landmarks revealed easy to detect with the smartphone's built-in camera. Unfortunately, we noticed that the capture time for a smartphone is time consuming. Distance and orientation calculation are very dependent of the quality of the color segmentation stage output. Therefore in conditions where illumination changes often, incorrect measurements of distance and orientation can occur. Overall, the solution presented obtained acceptable results for such a simple method, having its quality bottleneck on the color segmentation process.

Localization using Particle Filters had its result outcome worse than expected due to the limitations encountered in the measurement model used. With a more correct and precise measurement model, surely the results would have been greatly improved. Nevertheless, in the experimental results conducted, in simulation and on the field, the outcome was always in positions similar to the actual mobile robot location.

Path Planning with Potential Fields, although not always ensuring optimal paths for robot motion, it still provides possible obstacle-free paths with low execution times and memory usage.

Experiments on the field allowed us also to perceive the odometry errors caused by the mobile robotics kit used. The mobile implementation of the algorithms, revealed that although all the constraints and migration rules applied, algorithm consistency was kept. Experimental results also show that it is feasible to execute some navigation algorithms in real-time for robot motion, under certain limitations. This feasibility for real-time applications testifies the current processing capacity's state in high-end mobile phone devices.

Future improvements for this work include:

- Further optimization of the implemented algorithms for faster execution;
- Implementation of other navigation algorithms;
- Further experimental testing on the field considering more complex environments;
- Identification of more than one landmark per captured image;
- Study and testing of more Particle Filter measurement sensors;
- Potential Fields local minima lookahead study.

References

[1] R. Baczyk, A. Kasinski, and P. Skrzypczynski. Vision-based mobile robot localization with simple artificial landmarks. *Prepr. 7th IFAC Symp. on Robot Control*, pages 217–222, 2003.

[2] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. *In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, 3:2061–2066, October 2000.

[3] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[4] D. Fox. *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] M. A. Goodrich. Potential fields tutorial. *Class Notes*, 2002.

[6] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings for Radar and Signal Processing*, 140(2):107–113, April 1993.

[7] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[8] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *The American Association for the Advancement of Science*, 220(4598):671–680, May 1983.

[10] L.-F. Lee. 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] Lego. http://mindstorms.lego.com/, last visited in February 2008.

[12] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[13] D. G. Lowe. Object recognition from local scale-invariant features. *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157, 1999.

[14] M. J. Matarić. *The Robotics Primer*. The MIT Press, 1st edition, 2007.

[15] R. Negenborn. Robot localization and kalman filters. Master's thesis, Utrecht University, Netherlands, September 2003.

[16] Nokia. Nokia nseries n95 smartphone. http://www.nseries.com/products/n95/, last visited in February 2008.

[17] Nokia. Nokia nseries n80 smartphone. http://www.nseries.com/products/n80/, last visited in June 2008.

[18] M. G. Park and M. C. Lee. Artificial potential field based path planning for mobile robots using a virtual obstacle concept. *Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, 2:735–740, July 2003.

[19] D. Prasser and G. Wyeth. Probabilistic visual recognition of artificial landmarks for simultaneous localization and mapping. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, 1:1291–296, September 2003.

[20] I. Rekleitis. *Cooperative Localization and Multi-Robot Exploration*. PhD thesis, School of Computer Science, McGill University, Montréal, January 2003.

[21] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167 – 193. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[22] J. Solórzano, B. Bagnall, J. Stuber, and P. Andrews. lejos: Java for lego mindstorms. http://lejos.sourceforge.net/, last visited in June 2008.

[23] SunMicrosystems. Java me technology. http://java.sun.com/javame/technology/, last visited in May 2008.

[24] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, September 2005.

[25] P. Veelaert and W. Bogaerts. Ultrasonic potential field sensor for obstacle avoidance. *IEEE Transactions on Robotics and Automation*, 15(4):774–779, 1999.

[26] K.-J. Yoon, G.-J. Jang, S.-H. Kim, and I. S. Kweon. Fast landmark tracking and localization algorithm for the mobile robot self-localization. *in IFAC Workshop on Mobile Robot Technology*, pages 190–195, 2001.