ANALYSIS AND IMPLEMENTATION OF LOCALIZATION AND MAPPING ALGORITHMS FOR MOBILE ROBOTS BASED ON RECONFIGURABLE COMPUTING

M. C. SACCHETIN, J. J. LOPES, D. F. WOLF, J. L. SILVA and E. MARQUES

Computer Systems Department, University of São Paulo
Av. Trabalhador São-carlense, 400 – centro - CEP: 13566-970, São Carlos - SP, Brazil
e-mail:{sacchet.joelmir.denis.jsilva.emarques}@icmc.usp.br
http://www.icmc.usp.br

Abstract— **Localization and Mapping are fundamental problems in the field of mobile robotics that have been receiving considerable attention of the scientific community in the last ten years. Most of the work in this area is developed using personal computers and it still a challenge to execute these algorithms on embedded systems. This paper describes the analysis and embedded implementation of particle filter and occupancy grid algorithms, used for localization and mapping respectively. Experimental results and performance analysis were obtained using the softcore Altera Nios II running on Stratix II FPGA devices.**

Keywords — **Embedded Systems, Mobile Robotics, Localization, Mapping.**

I. INTRODUCTION

In the last years a considerable amount of research in mobile robotics has been focusing on localization and mapping. Particularly, the problem of simultaneous localization and mapping (SLAM) has been receiving attention from the researchers in the field. A large part of the approaches for these problems are based on probabilistic theory.

Some works describe theoretical proofs of the convergence of the problem (Dissanayake *et al.*, 1999); experimental verification of the adaptation of the robot behavior to improve the precision (Leonard and Feder, 1999) and different implementations for robots in external and internal environment (Leonard and Feder, 1999) (Castellanos *et al.*, 1999). However, those solutions are developed to solve a conceptual problem that in most cases requires a large amount of computation to be executed in real time. As most of these implementations are design to be executed on personal computers, embedded solutions for these problems still pose challenges due to their computation limitations.

This paper presents an embedded implementation for particle filter and occupancy grid algorithms, used respectively for mobile robot localization and mapping. Our approach is based on code originally developed in C language, which was analyzed and modified to be executed on an embedded system based on a field programmable gate array (FPGA) device.

The rest of the paper is organized as follows. Section II presents an introduction for the FPGA technology. Section III describes the particle filter algorithm, used for robot localization. A brief description of the occupancy grid algorithm used for the mapping is presented in the Section IV. Section V describes the implementation of the algorithms in FPGA. Section VI presents the experimental results and Section VII shows the conclusions and proposes some future work.

II. FPGA TECHNOLOGY

With the evolution of the micro-electronics, FPGA becomes an intermediate element among the General Purpose Processors and Application Specific Integrated Circuit (ASIC) (Dehon,1996).

A specific application project, in general, is very inefficient or inappropriate for others applications. Due to the possibility of reconfiguration of the circuits, a FPGA can operate as a variety of specific architectures (Cappelatti, 2001).

A FPGA is a digital integrated circuit that contains a regular structure of configurable cells and a programmable interconnection, and can be used to implement arbitrary digital systems, limited for the number of cells and available interconnections. When a FPGAs is configured for some application, it can be viewed like ASICs (Hauck, 2000).

As in general-purpose processors, FPGAs are programmed after the production to solve many computational tasks. However it is possible to explore the parallelism of the application program implementing different parts of the program, inside of the FPGA. As an example, some parts of the program can be executed in a regular general-purpose processor (which can be placed inside the FPGA) and the intensive computational part of the code can be executed on dedicated hardware parts of the FPGA. This can result on a considerable performance gain when compared to traditional software implementations executed on general-purpose processors.

III. PARTICLE FILTER

Particle filter is a sampling-based estimated method derived from the Bayes filter (Dellaert *et al.*, 1999). In the mobile robotics localization context, each particle corresponds to the possibility of the robot being at a specific position. This localization method requires some previous knowledge about the environment, usually represented as a map. Particles propagation (action model) is based on odometric information plus an added Gaussian noise to compensate for possible odometry errors. As robot's sensors acquire information about the environment, that information along with the map data is used to weight the particles according to their likelihood to be in the correct position (observation model). After the particles being weighted, a resample step takes place. All the particles are sampled based on their weight. High weighted particles have more chanced to be chosen for the resampling than the low weighted ones. In this manner, only height weighted particles (high likelihood to represent the correct) will remain and given enough number of particles, this technique is proved to converge.

IV. OCCUPANCY GRID

Occupancy grid mapping is a very popular technique introduced by Elfes (1986). In this algorithm, the environment is represented by a two-dimensional grid of small cells. Each cell can assume one of the 3 possible states: occupied, free, or unknown. The state of each cell is determined based on the probability of it being occupied or not. The occupancy probability of each cell is updated as range information is obtained from the sensors. For each range bean, a line is traced between the robot's position and the next obstacle on that particular direction. Every cell crossed by the line is considered empty and the cell in which the line ends is considered occupied.

Due to its probabilistic formulation, the occupancy grid algorithm can successfully handle range sensor noise. A disadvantage of this approach is the scalability. In order to have a good representation for the environment the size of the grid cells has to be small. Large environments require a very high number of grid cells, which imply in a huge amount of memory. This algorithm can handle indoor environments well but it is not suitable for large outdoors or 3D maps. Another disadvantage of the occupancy grid mapping is the incapability to manage uncertainty in the robot's pose. This algorithm assumes that the robot's position is known during the mapping task, which is not true in most real situations. The environment representation by grid cells still widely used by the mobile research community, mostly in conjunct with other mapping techniques. The occupancy grid implementation presented in this paper is based on Howard (2004).

V. EMBEDDED IMPLEMENTATION

The original software implementations for localization and mapping described in the previous section were analyzed to accomplish the embedded implementation. For that, we analyzed the codes to identify intensive computational functions that could be optimized through reconfigurable computing.

The Altera gprof tool (Gprof, 2005) was used to identify the most computing intensive functions and to modify them for partial execution in hardware. Based on the results generated by gprof, it was possible to identify very intensive computational functions, such as particle filter sampling, normal distribution calculation, and mean calculation used through the algorithms. Figure 1 illustrates the processing percentage of the most relevant functions of the software.

Data Flow Graphs (DFGs) of the mean and normal functions were generated for better understanding and analysis of the execution of the most intensive computational functions. The original code of the filter of particles developed by (Wolf *et al.*, 2005) in C was modified suppressing the threads of the function, so that could be analyzed correctly with the Gprof (Gprof, 2005) of Linux.

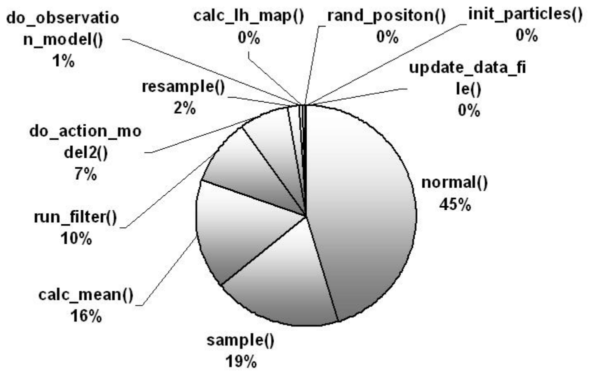

Fig. 1. Percentage of the execution time for each function of the particle filter algorithm

The three most intensive computational functions in the original implementation are responsible for the execution of several operations in flotation point. To improve the execution of the algorithm in FPGA for an embedded solution the unit of flotation point, developed in the Reconfigurable Computing Laboratory (LCR) of the University of Sao Paulo denominated FPMU (Rodrigues, 2002), was added as a custom instruction to the Nios II (Fig 3).

The FPMU is a unit of flotation point developed in VHDL (IEEE-754 standard) capable to execute the following instructions directly in hardware: addition, subtraction, division and multiplication with real numbers of 32 bits. In the implementation in VDHL, the signs dataa[31 ..0] and datab[31 ..0] receives the entrance operands, result[31 ..0] receives the result and n[7 ..0] is the code of the instruction to be executed. The main component in the FPMU also presents control signs, such as clk (clock), clk_en (clock enable) start, reset and done. The specification of the FPMU is compatible with the extended architecture of personalized instructions of the Nios II, and described in Fig. 2.

VI. EXPERIMENTAL RESULTS

The gprof tool was used to compare the original software to the embedded implementation. The original code was compiled and executed on a PC loaded with a AMD Athlon (TM) XP 2000+ (1660MHz) processor, 256 KB of cache, 1G of RAM, and Linux operating system. The embedded implementation was tested in Altera stratix FPGA device. The robot and sensors input data for the experiments was obtained at the USC campus

(Wolf *et al*., 2005), using a segway RMP robot and laser range finders. The trajectory followed by the robot during the data collection consists of a 2 km run, which included 3 complete loops.

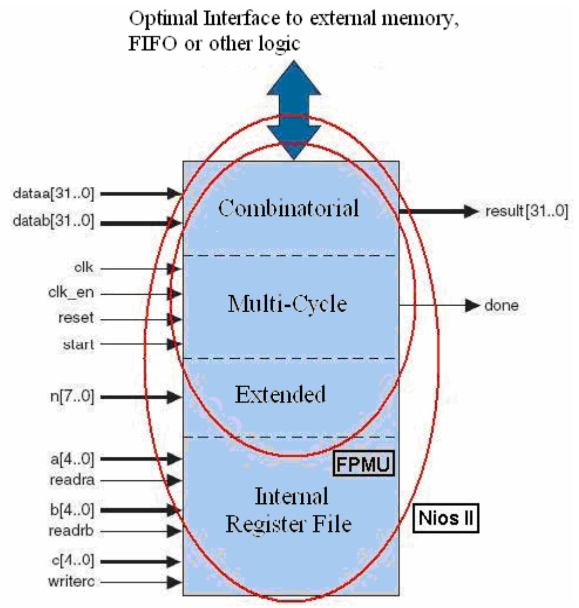

Fig. 2. The block diagram of hardware FPMU extended to the Altera Nios II processor

Although the execution time of the functions executed in FPGA was longer, the results were satisfactory considering that the clock of the processor Nios II used to execute the embedded implementation was considerably slower than the one in the PC (50MHz and 1666MHz respectively). Table 3 shows the processing time of the three embedded functions that were modified to execute instructions of flotation point in hardware. Other important point to be taken to account is the energy consumption, which is much smaller in the Nios II than in the PC processor. Similar analysis was generated for the algorithm occupancy grid. The three more intensive computational functions were identified and analyzed in the Table 4.

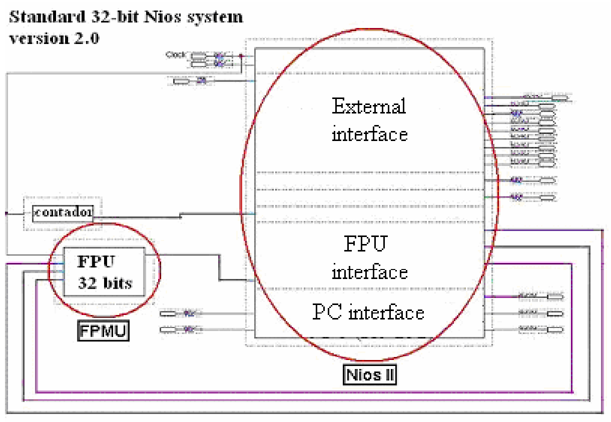

Fig. 3. NIOS II connected to the FPU.

Research projects related to this subject are still in process in the Laboratory of Reconfigurable Computing at USP (Gonçalves *et al*., 2003). The use of the unit of flotation point FPU was not used as personalized instruction of Nios II as shown in the Fig 3 but as an external component to the processor.

Our laboratory very recently obtained a Pioneer robot with a laser range finder (Fig. 4). Soon the localization and mapping experiments will be performed at USP campus, allowing experiments on different types of environment and with different parameter combinations (Figs. 5).

Table 1. Profile generate by nios2-elf-gprof through of execution of the particle filter on a PC

Function	% Time	Time(sec.)	N Call
rand_position()	0.41	0.24	100
rand()	0.26	0.15	
normal()	0.24	0.14	9900
calc_lh_map()	0.21	0.12	1
floor()	0.20	0.12	
do_action_model2()	0.19	0.11	9900
sample()	0.08	0.05	9900
run_filter()	0.06	0.03	1
calc_mean()	0.04	0.02	99

Table 2. Profile generate by gprof through of execution in FPGA

Function	% Time	Time(sec.)	N Call
rand_position()	50.00	0.11	100
normal()	18.18	0.04	9900
run_filter()	9.09	0.02	1
sample()	4.55	0.01	9137
do_action_model2()	0.00	0.00	9900
arred()	0.00	0.00	1191
round_angle()	0.00	0.00	198
update_data_file()	0.00	0.00	100
calc_mean()	0.00	0.00	99

Table 3. Performance comparison of the particle filter executed on a PC and in the Altera NIOS II processor

Function	**(Wolf, *et al*., 2005)**		**Embedded**	
	% of time	***Time (s)***	***% of time***	***Times (s)***
normal()	45.16%	0.04 s	0.24%	0.14 s
sample()	18.89%	0.01 s	0.08%	0.05 s
calc_mean()	16.23%	< 0.01 s	0.04%	0.02 s

VII. CONCLUSIONS

In this paper we presented embedded implementations for localization and mapping algorithms (particle filter and occupancy grid respectively) for mobile robots. FPGA technology was used in the embedded implementations to validate the experiments. Analyses of the original code (designed to run on a PC) were generated to identify intensive computational functions of the program to be executed in reconfigurable hardware. The execution in hardware obtained considerable performance gain. It is also important to mention that the embedded implementation of the localization and mapping algorithms present much smaller energy consumption when compared to the PC equivalent ones. It is a very important point since energy is obtained from batteries on most autonomous mobile robots.

As future work, we plan to improve the performance of the embedded algorithms combining pure hardware to sets of general-purpose processors on the same FPGA chip. Other mobile robotic algorithms such as potential fields and topological mapping are currently being analyzed to have their implementation embedded on FPGA devices.

Table 4. Performance comparison of the occupancy grid executed on a PC and in the Altera NIOS II processor

Function	**(Howard, 2004)**		**Embedded**	
	% of time	***Time (s)***	***% of time***	***Time (s)***
omap_add()	84.62%	0.11 s	0.04%	0.20 s
pose2_add_pos()	15.38%	0.02 s	0.01%	0.10 s
omap_alloc ()	< 0.01%	< 0.01 s	< 0.01%	0.01 s

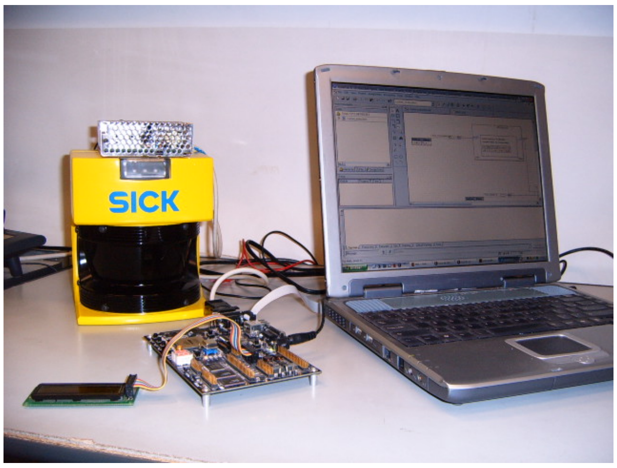
Fig. 4. Laser unit connected to the Stratix FPGA board.

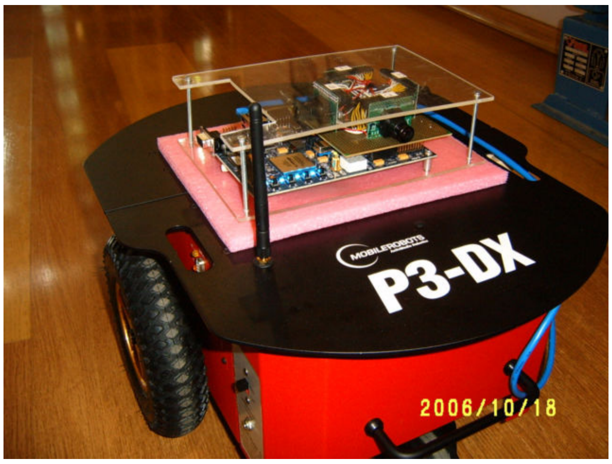
Fig. 5. Stratix FPGA board connected to a Pioneer 3 - DX robot

REFERENCES

Cappelatti, E.A., "Implementação do Padrão de Barramento PCI para Interação Hardware/Software em Dispositivos Reconfiguráveis", *Master Thesis, Pontifícia Universidade Católica do Rio Grande do Sul* (2001).

Castellanos, J.A., J.M.M. Montiel, J. Neira and J.D. Tardos, "Sensor influence in the performance of simultaneous mobile robot localization and map building", *In Proc. 6th International Symposium on Experimental Robotics*, Sydney, Australia, 203-212 (1999).

Dehon, A., "Reconfigurable Architecture for General-Purpose Computing", *Ph.D. thesis, Massachusetts Institute of Technology* (1996).

Dellaert, F., D. Fox, W. Burgard and S. Thrun, "Monte Carlo Localization for Mobile Robots", *In proceedings of IEEE International Conference on Robotics and Automation*, 99-114 (1999).

Dissanayake, M.W.M.G, P. Newman, H.F. Durrant-Whyte, S. Clarck and M. Csobra, "An Experimental and Theoretical Investigation Into Simultaneous Localization and Map Building (SLAM)". *In Proc. 6th International Symposium on Experimental Robotics*, Sydney, Australia, 171-180 (1999).

Elfes, A., "Sonar-based real-world mapping and navigation", *IEEE Transactions on Robotics and Automation*, **3**, 249-265 (1986).

Gprof, online documentation http: //www.gnu.org/ software/binutils/manual/gprof-2.9.1/html_mono / gprof. html (2005)

Hauck, S., "Reconfigurable Computing: A Survey of Systems and Software", *ACM Computing Surveys*, **34**, 171-210 (2000).

Howard A., "Simple Mapping Utilities" http://www-robotics.usc.edu/~ahoward/pmap/. (2004).

Leonard, J.J. and H.J.S. Feder, "Experimental Analysis of Adaptive Concurrent Mapping and Localization Using Sonar". *In Proc. 6th International Symposium on Experimental Robotics*, Australia, 213-222 (1999).

Rodrigues, M.I., "Projeto de uma unidade aritmética de ponto flutuante - Padrão IEEE 754 - implementada em computação reconfigurável", *Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação ICMC-USP* (2002).

Wolf, D.F., A. Howard and G. Sukhatme, "Towards Geometric 3D Mapping of Outdoor Environments Using Mobile Robots", In Proceedings of *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1258-1263 (2005).

Gonçalves, R.A., P.A. Moraes, J.M.P. Cardoso, D.F. Wolf, M.M. Fernandes, R.A.F. Romero and E. Marques, "Architect-R: A System for Reconfigurable Robots Design". *In ACM Symposium on Applied Computing-Embedded Systems: Applications, Solutions, and Techniques*, Melbourne, USA, 679-683 (2003).

Received: April 14, 2006.
Accepted: September 8, 2006.
Recommended by Special Issue Editors Hilda Larrondo, Gustavo Sutter.