Proceedings of COBEM 2005
Copyright © 2005 by ABCM

18th International Congress of Mechanical Engineering
November 6-11, 2005, Ouro Preto, MG

Mobile Robot Path Control Using Landmark Information

Gabriela Werner Gabriel
Instituto Tecnológico de Aeronáutica, Divisão de Engenharia Eletrônica, 12228-900 – São José dos Campos – SP, Brazil
gabriela@ita .br

Cairo Lúcio Nascimento Júnior
Instituto Tecnológico de Aeronáutica, Divisão de Engenharia Eletrônica, 12228-900 – São José dos Campos – SP, Brazil
cairo@ita.br

Eduardo Hisasi Yagyu
Instituto Tecnológico de Aeronáutica, Divisão de Engenharia Eletrônica, 12228-900 – São José dos Campos – SP, Brazil
yagyu@ita.br

Abstract. *This paper is concerned with the design, assembly and testing of an autonomous mobile robot named ROMEO III. This robot was designed to follow a planned trajectory by combining data from odometry sensors with landmark information. The odometry sensors are optical encoders built in auxiliary wheels. The landmark information is acquired by 5 infrared sensors that are used to detect white line segments on a black floor. A path control algorithm is then proposed and evaluated. The experimental results show that the proposed path control algorithm successfully avoids the accumulation of the odometry errors and the autonomous mobile robot follows the planned trajectory with small errors.*

Keywords: *mobile autonomous robots, trajectory control, robot navigation, infrared sensors, embedded computing.*

1. Introduction

Research and development on mobile robotics began on 1950s when W. Grey Walter constructed the first known mobile robot, which was called “tortoise” (Arkin, 1998). This robot was constructed as an analog device and consisted of two sensors, two actuators and two “nerve cells” made of vacuum tubes. Since then, the advances in analog and digital electronics, precision mechanics, embedded computing and artificial intelligence techniques have been applied to the field of mobile robotics. Some examples of state-of-the-art in this field are: 1) the Rhino Project developed by the University of Bonn where autonomous robots are used as museum tourguides (http://www.cs.uni-bonn.de/~rhino/), 2) the Roomba Robotic FloorVac (http://www.roombavac.com), an intelligent robotic vacuum cleaner built by the iRobot Co., 3) the ASIMO robot (http://www.world.honda.com/ASIMO/), a two-legged humanoid robot built by Honda Co., and 4) the AIBO robots (http://www.sony.net/Products/aibo/), personal entertainment robotic dogs build by the Sony Co.

One crucial problem in the field of mobile robots is navigation. The robot has to deal with three basic questions (Leonard and Durrant-White, 1991): “Where am I?”, “Where am I going?” and “How should I get there?”. Several researchers have proposed different approaches to answer these questions. A possible approach is known as SLAM, simultaneous localization and mapping (Csorba, 1997).

This paper is concerned with the design, assembly and testing of an autonomous mobile robot named ROMEO III shown in Fig. 1. This robot was designed to follow a planned trajectory in a perfectly known and structured environment using information provided by odometry and infrared sensors. The odometry sensors are optical encoders (one for each wheel) that are used to estimate the robot pose (its x-y location and its heading). The infrared sensors are used to detect the landmarks in the structured environment. The landmarks are line segments and node points on a two-dimensional grid constructed with white reflective tape on a black floor. Therefore the grid specifies the robot operating area. Fig. 2 shows an example of environment where the robot can navigate. The robot trajectory is planned using the A* algorithm (Nascimento Jr. and Yoneyama, 2000), given the supposed known robot initial pose, the desired robot final position, the grid parameters and the assumed known obstacle positions.

Several authors have proposed different approaches for combining the data from the odometry sensor with data from other sources to estimate the robot pose (Bezerra, Alsina and Medeiros, 2004, Betke and Gurvits, 1997, or Briechle and Hanebeck, 2004). By combining the information from the infrared sensors (used to correct the robot orientation and to detect when the robot is over a node points) with the data provided by the odometry sensors, our tests show that the robot pose estimates have much greater accurary than those obtained when only the odometry data was used.

The ROMEO III autonomous mobile robot was developed using the experience and knowledge acquired by constructing the ROMEO I and ROMEO II mobile robots (Yoshitome, 1997, Gabriel, 2003). The ROMEO III dimensions are approximately 35 cm x 20cm (diameter x height). Its weight is about 3.5 kg.

This article is organized as follows. Section 2 presents the ROMEO III system architecture and in section 3 the hardware for each robot system is described. Section 4 presents the strategy programmed on the onboard computing system such that the robot precisely follows the planned trajectory. Section 5 describes experimental results. Concluding remarks and some directions for future research are given in section 6.

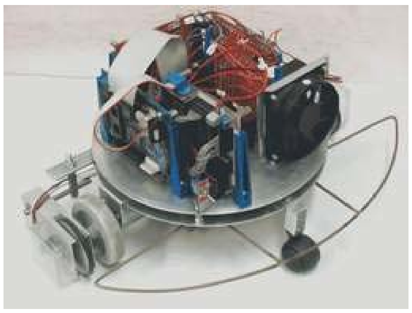


Figure 1 – The autonomous mobile robot ROMEO III.

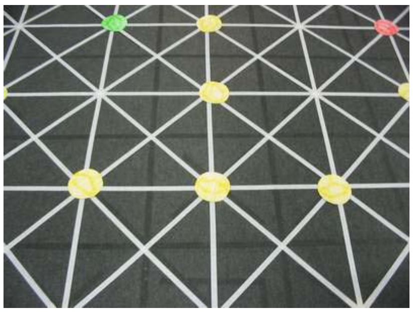


Figure 2 – The well structured environment where the mobile robot operates.

2. The ROMEO III robot system architecture

The autonomous mobile robot ROMEO III is comprised of 5 systems: external communication system, onboard computing system, propulsion system, sensory system and energy system. Figure 2 shows how these systems are interconnected.

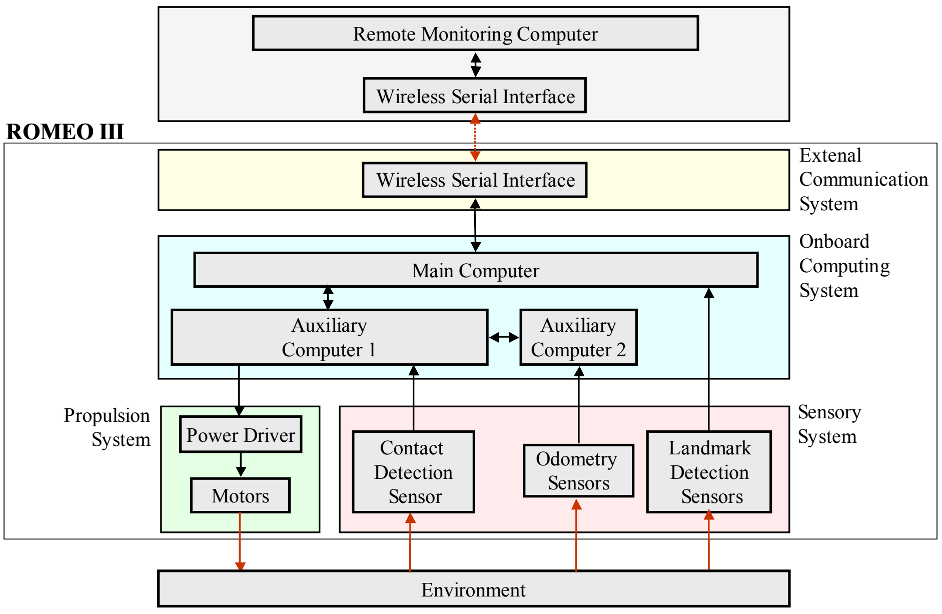


Figure 3 – Block diagram of the ROMEO III robot system architecture.

The remote monitoring computer (a typical desktop computer running MS-Windows® and MATLAB®) performs the following tasks sequentially:

1) plans the robot trajectory using the A* algorithm (Nascimento Jr. and Yoneyama, 2000), given a detailed description of the environment where the mobile robot operates (the grid parameters and the assumed known obstacle positions), the desired travelling speed, the supposed known robot initial pose and the desired robot final position,
2) transfers the planned robot trajectory to the robot's onboard computing system,
3) commands the robot to start to follow the planned trajectory,
4) receives continual updates of the robot pose estimate and shows graphically and (in near real-time) this information on its computer screen.

The planned trajectory is encoded in a file and the desired robot movements are coded as high-level instructions (speed, distance if translational movement, angle if rotational movement). The A* algorithm was implemented using the C language. A graphical user interface with the following features was written using MATLAB®: 1) it communicates with the robot's onboard computer system by sending and receiving text messages (it writes to and reads from the remote monitoring computer standard serial port emulating a serial terminal program), 2) when it receives a message containg the current robot pose estimate, this information is shown graphically on the computer screen.

The external communication system is responsible for establishing the communication between the remote monitoring computer and the robot's onboard computing system. This is implemented using a wireless serial link.

The onboard computing system (hardware and software) is responsible for defining the robot's embedded intelligence, which means its ability to take decisions given its mission, its perception of the environment and its own internal state. The onboard computing system performs the following tasks:

1) exchanges data with the remote monitoring computer by receiving the planned trajectory and by continually sending information about the mission status (defined as the robot pose estimate and the occurrence of several normal and abnormal events such as detection of collision and low-battery state),
2) executes all tasks related to the robot navigation by processing the incoming information from the sensory system, estimating the current robot pose, comparing it to the desired robot pose, and sending commands to the propulsion system, and
3) checks the robot's health by using the information gathered by the sensory system about some of the robot internal components, which for now means just reading the low-battery detection sensor.

The propulsion system receives the low-power digital signals from the onboard computing system and generates signals that are powerful enough to be applied to the motors such that the robot executes the movement selected by the onboard computing system.

The sensory system must interact with the environment where the robot operates, gather the relevant information and present it to the onboard computing system as a digitally encoded signal. Additionally, the sensory system can also gather information about the robot's health. The ROMEO III sensory system is composed of: 1) the contact detection sensor: used to detect frontal collisions (Fig. 1), which are defined as an abnormal events that abort the navigation program and stop the robot, 2) the odometry sensors: one for each wheel, used to estimate the robot pose at any instant, 3) the landmark detection sensors: 5 infra-red sensors located under the robot used to detect the line segments and node points on a two-dimensional grid constructed with white reflective tape on a black floor, 4) the low-battery detection sensor: detects when the onboard battery voltage is below a certain threshold level (not shown in Fig. 3).

The energy system is responsible for electrically powering all the other robot systems and is not shown in Fig. 3.

3. Hardware description

3.1. External communication system

The external communication system employs a wireless serial link to connect the onboard computing system with the remote monitoring computer. This wireless link is implemented using 2 Bluetooth modules from Wireless Futures Co (http://www.wirelessfutures.co.uk). These modules use the Bluetooth Serial Port Profile (SPP) to provide a "virtual wireless cable". This avoids the need for a physical cable linking the remote monitoring computer and the ROMEO III mobile robot with the advantage that the communication software routines on both computers treat these Bluetooth modules as standard RS-232 serial ports. Accordingly to the manufacturer, the operating range of these Bluetooth modules is 100 m.

3.2. Onboard computing system

The onboard computing system is composed of three different embedded computer boards, which exchange information using RS-232 serial ports and their data buses. This structure is preferable than using just one computer for the following reasons: 1) more I/O pins are available for communication with the other robot systems, 2) since there is a fewer number of tasks being executed on each computer, then the overall computing system will respond faster to critical events (such as reading the infrared sensors), and 3) it is easier to write and debug the onboard software, as previously observed by Borenstein (1987).

The three embedded computers are: 1) Flashlite 386Ex: a DOS-based single board computer (SBC) with an Intel 386Ex microprocessor running at 25 MHz, 512 kB SRAM, 512 kB Flash memory, 2 RS-232 serial ports, 36 digital I/O ports, manufactured by JKmicrosystems Inc. (http://www.jkmicro.com/), 2) an Infineon C515 microcontroller board running at 11.0592 MHz with 32 kB external RAM, 32 kB external EPROM, 1 RS-232 serial port, 28 digital I/O ports, and 3) an Atmel AT89C52 microcontroller board running at 11.0592 MHz with 256 B internal RAM, 8 kB internal Flash memory, no serial port, 30 digital I/O ports. The C515 and the AT89C52 microcontroller boards were designed and assembled at our laboratory.

These three computers exchange digital data as follows: a) the Flashlite 386Ex board uses one serial port to communicate with the wireless communication system and the other serial port to communicate with the C515 microcontroller board, b) the C515 microcontroller board communicates with the AT89C52 board through the I/O digital ports. The Flashlite 386Ex board is the main computer and the other boards act as auxiliary computers (Fig. 3).

The Flashlite 386Ex board is responsible for all high-level tasks related to the robot navigation. It operates at the tactical level as follows: firstly it receives from the remote monitoring computer a file that encodes the planned trajectory as high-level movements (speed, translation/distance, rotation/angle), then continually performs the following actions in sequence: 1) decides the next low-level movement, that is, it computes for each motor in which direction (clockwise/counterclockwise) and for how long the motor should rotate, 2) instructs the C515 microcontroller board to execute this low-level movement, 3) waits until the C515 microcontroller board reports back that the low-level movement has been completed, 4) asks the C515 microcontroller board for the odometry sensors readings, 5) computes the current robot pose estimate and sends this information to the remote monitoring computer, 6) reads the landmark detection sensors (connected to 5 of its digital I/O ports) and decides if the robot orientation is correct or wrong: if wrong but correctable then a correction low-level movement is executed and this action is repeated until the robot orientation is considered correct; if wrong but not correctable then the navigation program is aborted; if correct then the program goes back to action 1.

When the C515 microcontroller board receives a message from the Flashlite 386Ex board through its RS-232 serial port with instructions about the next low-level movement, it configures the propulsion system such that this low-level movement is executed and notifies back when this movement is completed. The C515 microcontroller board is also responsible for sending messages to the Flashlite 386Ex board: a) with the readings for the odometry sensors, which are acquired from the AT89C52 board using its I/O pins, and b) when the contact detection sensor is triggered.

The AT89C52 board is responsible for: a) continuously sending to the propulsion system a signal that encodes the speed that the motors should rotates, and b) continually counting the number of pulses generated by the odometry sensors.

3.3. Propulsion system

The propulsion system is composed of a power driver electronic circuit and 2 permanent magnet DC motors that are driven independently of each other. The motors are rated as 24 V and 0.14 A (but are powered by a 12 V battery in this case), model CN35-09720 Canon, and are coupled by their own internal gearbox to the wheel axes.

Figure 4 shows a block diagram of the power driver circuitry. Its main parts are: 1) a dual full bridge driver implemented with a L298 chip that is directly connected to the motors windings, 2) 6 optocouplers implemented with 6 TIL111 chips that are used to optically isolate the onboard computing system from the circuitry that electrically drives the DC motors, 3) a "buffer" implemented with a SN74LS244 chip to act as a current amplifier such that the 6 low-power signals from the onboard computing system can drive the optocoupler chips. The full bridge driver chip performs bidirectional variable speed motor control using the PWM (Pulse Width Modulation) technique (Gabriel, 2005).

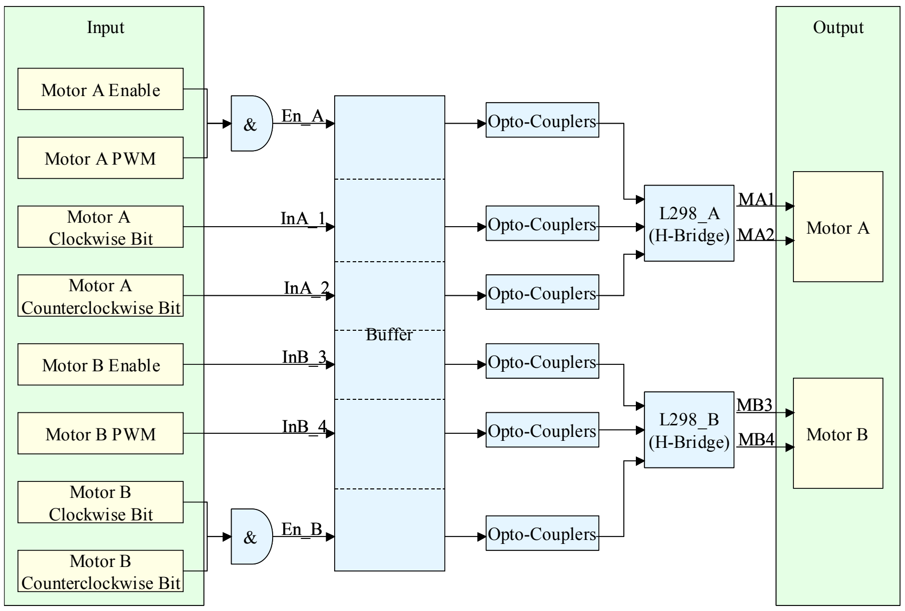


Figure 4 – Block diagram of the power driver circuitry.

3.4. Sensory system

As indicated in Section 2, this system is composed of: 1 contact detection sensor, 2 odometry sensors, 5 landmark detection sensors, and 1 low-battery detection sensor.

The contact detection sensor is implemented by placing a long curved wire in front of the robot to act as a bumper switch. This wire is mechanically attached to 2 snap-action microswitches with levers that are normally open. The microswitches are connected to "pull-up" resistors and to a AND gate. If both microswithes are open the output of the

AND gate is a logical "1". If any of the microswitches is closed than the output of the AND gate is a logical "0" activating an interrupt routine on the C515 microcontroller board.

The 2 odometry sensors are incremental optical encoders attached to 2 auxiliary wheels (one sensor for each wheel). These auxiliary wheels are aligned with the main robot wheels (the wheels attached to the axes of the DC motors) as shown in Fig. 5. The optical encoder is implemented by attaching a disk with slots cut in it to the axes of a auxiliary wheel and a near-infrared led is placed on one side of the disk's slots and a phototransistor is placed on the other side. As the disk spins, the light passing through the disk is interrupted by the moving slots, and a signal in the form of a pulse train is produced at the output of the phototransistor that is connected to an interrupt pin at the AT89C52 microcontroller board. Therefore each pulse activates an interrupt routine that counts the number of pulses. The Flashlite 386Ex periodically requests this information (using the C515 microcontroller board), combines it with data from other sources and computes the robot pose estimate. The data that comes from the odometry sensors is known to be noisy and biased due to several reasons, for instance, unequal wheel diameters and wheel slippage (Gabriel, 2005, Borestein, Everett and Feng, 1996).

Each of the 5 landmark detection sensors is implemented as a photodiode coupled with a phototransistor. Figure 6 shows the location of the 5 landmark detection sensors attached to the bottom of the robot chassis. The L2, L3 and L4 are used to detect the white line that should be parallel to the robot center line. The L1 and L5 sensors are used to detect lines that should be parallel to the driven wheels axis. Experimental tests for our setup showed that: a) if the photodiode-phototransistor pair is over a white line the phototransistor output is between 1.3 V and 2.6 V, b) if the photodiode-phototransistor pair is over the black floor the phototransistor output is between 4.4 V and 4.6 V, depending on which pair was tested and the intensity of the ambient light. To distinguish between the two situations, in each sensor the phototransistor output was compared with the output of a 3.9 V zener diode. This comparison was implemented for the 5 sensor using 3 LM393 chips (a low power dual voltage comparator).

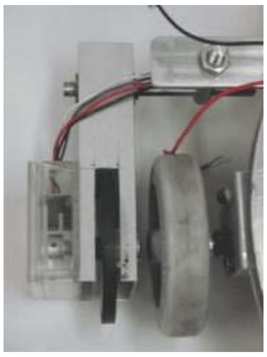


Figure 5 - Physical mounting of one of the encoders used in ROMEO III.

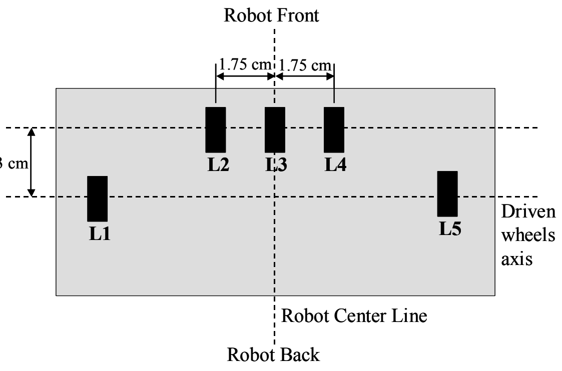


Figure 6 – Location of the 5 landmark detection sensors (top view).

The nominal output voltage for the robot onboard battery is 12 V. The low-battery detection sensor is used to detect when the battery voltage is equal or below to 10.1 V. This sensor is implemented by the circuit shown in Fig. 7 and uses a LM393 chip, two 4.7 V zener diode and some resistors. If the battery voltage is above 10.1 V, then *Vout* is 0 V. As the battery is discharged its voltage slowly decreases and when it reaches 10.1 V, *Vout* changes to 4.7 V (a logical "1"), the main computer stops sending commands to the propulsion system and the robot stops.

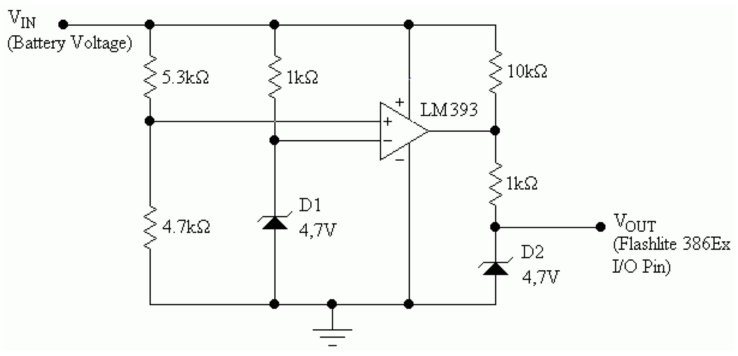


Figure 7 – Circuit for the low-battery detection sensor.

3.5. Energy system

The ROMEO III robot uses a 12 V 3.2 Ah battery. Since for the input (Vcc) voltage for some of the robot systems is 5 V, a standard 7805 regulator chip is used to create a 5 V energy source.

More details about the ROMEO III hardware and its electronic circuits are given in Gabriel, 2005.

4. Path control algorithm using landmark information

The robot must follow white lines constructed with white reflective tape on a black floor. Consider the case when the white lines are arranged as a two-dimensional rectangular grid without diagonal lines. Thus the node points are given by the intersection between the vertical and horizontal lines.

The onboard computing system must use the white lines to correct the robot orientation. These corrections occur when the robot is in one of the following situations: a) traveling between 2 node points (always in a straight line) or approaching a node point or leaving a node point, and b) executing a 90^o rotational movement over a node point. Table 1 shows the correction low-level movements as a function of the L2, L3 and L4 infrared sensors readings for the case when the robot is in the first situation.

Table 1. Correction movements when the robot is traveling between 2 node points or approaching a node point or leaving a node point as a function of the L2, L3 and L4 infrared sensors readings.

L2	L3	L4	Correction movement
black	black	black	Aborts the program – Abnormal situation
black	black	white	Clockwise rotation
black	white	black	none
black	white	white	Clockwise rotation (approaching a node point)
white	black	black	Counterclockwise rotation
white	black	white	Aborts the program – Abnormal situation
white	white	black	Counterclockwise rotation (approaching a node point)
white	white	white	none (approaching a node point)

A translational movement between node points should begin and end when the readings for the L1 and L5 sensors are simultaneously “white” and “white”. Then if a 90^o counterclockwise rotational movement is planned, table 2 shows the correction low-level movements as a function of the L1 and L5 infrared sensors readings. For a 90^o clockwise rotational movement a similar strategy is followed. Note that for a rotational movement the L1 and L5 sensors readings should go through the following transition stages: (“white”,”white”) → (“black”,”black”) → (“white”,”white”).

Table 2. Correction movements when the robot is performing a 90^o counterclockwise rotational movement over a node point as a function of the L1 and L5 infrared sensors readings.

Previous Readings		Current Reading		Correction movement (CC = Counterclockwise)
L1	L5	L1	L5	
white	white	white	white	none
white	white	white	black	CC movement with LEFT motor until L1=”black”, L5=”black”
white	white	black	white	CC movement with RIGHT motor until L1=”black”, L5=”black”
white	white	black	black	none
black	black	white	white	none
black	black	white	black	CC movement with RIGHT motor until L1=”white”, L5=”white”
black	black	black	white	CC movement with LEFT motor until L1=”white”, L5=”white”
black	black	black	black	none

When the white lines are arranged as a two-dimensional rectangular grid with diagonal lines as shown in Fig. 2, a more complicated set of correction movements should be programmed in the onboard computing system. Gabriel, 2005 discusses this case and implements a solution to solve it.

The proposed path control algorithm using landmark information is as follows:

1) whenever the infrared sensors detect that a translational or rotational movement was completed, it is assumed that the robot is over a node point on the two-dimensional grid with a known orientation and a known position. Therefore the robot pose can be inferred with great accuracy. It is also assumed that the initial robot pose (starting point of the planned trajectory) is known with great accuracy.

2) when the robot is executing translational or rotational movements, the infrared sensors readings are used to correct the robot orientation. At the same time only the odometry sensors readings are used to compute the robot pose estimate by applying well-known odometry equations (Borestein, Everett and Feng, 1996). This means that the robot's incremental change of orientation and the incremental linear displacement of the robot's centerpoint are computed using the incremental travel distance for the left and right wheels (which in turn are computed using the odometry sensors readings) given that the robot pose is known with great accuracy at the last node point. These computations are done assuming that angle of orientation is constant during a translational movement and that the robot's centerpoint is fixed during a rotational movement.

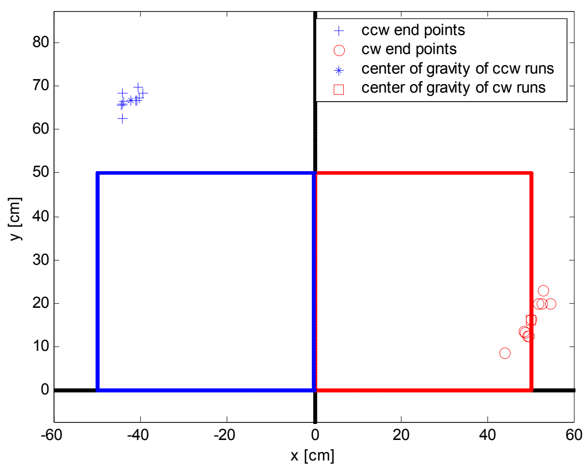
The proposed path control algorithm works by avoiding the accumulation of trajectory errors, a impossible task if one's is using only the odometry sensors readings (Borenstein, Everett, Feng, 1996).

5. Experimental results

In order to measure the accuracy of the proposed approach for path control, the bidirectional square path test (also known as UMBmark, University of Michigan Benchmark) was performed. This test, which was firstly proposed by Borenstein and Feng (1996) requires that the robot follow a square path in both clockwise and counterclockwise direction. In our tests a 50 cm x 50 cm square path was used and the initial and final robot pose should be the same.

Two cases were investigated and for each case 10 runs were executed in each direction. In the first case only odometry equations were applied to navigate the robot without using any sensor reading. It was assumed that both driven wheels have the same perfectly known diameter and that their speeds can be perfectly adjusted. Therefore one can compute for how long each wheel should be driven in order to execute the desired movements. In the second case the path control algorithm explained in section four was employed.

Figure 8 shows the test results for both cases. In each case it shows the results of each run and their average in both directions. In the first case the distance between the desired and the averaged actual final robot's centerpoint was 79 cm (± 1 cm) for the tests in the clockwise direction and 53 cm (± 4 cm) for the tests in the counterclockwise. In the second case, these distances were respectively 1,3 cm (± 0,3 cm) and 1,1 cm (± 0,2 cm).

a.


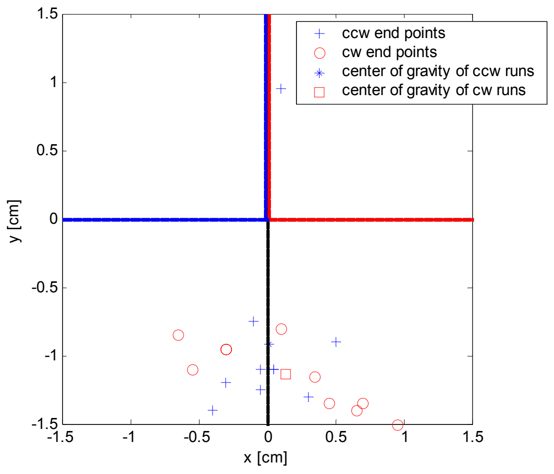
b.


Figure 8 – Results for the bidirectional square path test: a) using only odometry equations and measured robot parameters, b) using the proposed path control algorithm with landmark information.

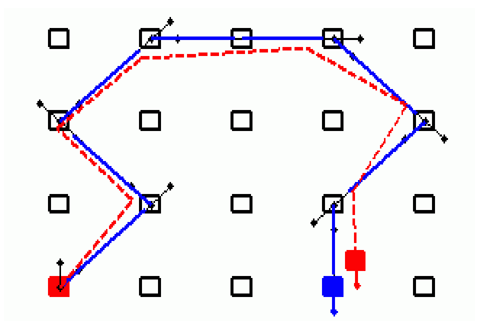


Figure 9 - Robot pose estimate using the proposed path control algorithm (blue solid line) and without forcing the robot pose to be the expected pose when the robot finishes a translational or rotational movement (red dashed line).

Figure 9 shows a test run for a square grid with diagonal lines. The distance between node points is 50 cm. The desired trajectory is composed of 5 diagonal movements, 3 side movements (shown as horizontal or vertical lines), 3 45^o clockwise rotations, a 45^o counterclockwise rotation, 2 90^o clockwise rotations, and a 90^o counterclockwise rotation. The total linear displacement is about 504 cm and the sum of the absolute values of the orientation changes is 450^o. Figure 9 shows the robot pose estimate during the execution of the desired trajectory using the path control algorithm proposed in section 4 (blue solid line). Visual observation of the mobile robot shows that the robot pose estimates are correct. Figure 9 also shows the robot pose estimate without forcing the robot pose estimate to be the expected pose when the robot finishes a translational or a rotational movement (red dashed line).

6. Conclusions and Future Research

This paper deals with the design and assembly of ROMEO III, an autonomous mobile robot that is capable of navigating in a well structured environment (a two-dimensional grid with or without diagonal lines) by combining the readings from odometry sensors with data from infrared sensors (landmark based navigation). The experimental results showed that the proposed path control algorithm overcomes the accumulation of trajectory errors, something that is impossible when only the odometry sensors are used ("dead reckoning" navigation).

Possible directions for future research are: 1) improvement of the sensory system, for instance, by adding sensors for noncontact near- and far-object detection, such as a camera or a sonar device, 2) investigation and implementation of SLAM (simultaneous localization and mapping) techniques, 3) evaluation of alternatives for the onboard computer system, 4) implementation of cheaper alternatives for the wireless communication system.

7. Acknowledgements

The authors wish to acknowledge the support for this research provided by FAPESP (grant 02/01583-5), CNPq and the Division on Electronics Engineering at the Aeronautical Institute of Technology (ITA).

8. References

Arkin, R. C., 1998, *Behaviour-Based Robotics*, MIT Press, Cambridge, USA.

Betke, M., Gurvits, L., 1997, "Mobile Robot Localization Using Landmarks", *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 2, pp. 251-263.

Bezerra, C. G., Alsina, P. J., Medeiros, A. A. D. de, 2004, "Localização de um Robô Móvel Usando Estimativa do Erro de Odometria e Transformada de Hough", *Proc. of the 15th Brazilian Conference on Automatics*, Gramado, Brazil, pp. 680-685 (in portuguese).

Borenstein, J., 1987, *Development of a Nursing Robot System*, PhD Thesis, Technion-Israel Institute of Technology, Haifa.

Borenstein, J., Feng, L., 1996, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots", *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 6, pp. 869-880.

Borenestein, J., Everett, H. R., Feng, L., 1996, *Navigating Mobile Robots: Systems and Techniques*, A K Peters, Wellesley, MA, pp. 103-108.

Briechle, K., Hanebeck, U. D., 2004, "Localization of a Mobile Robot Using Relative Bearing Measurements", *IEEE Trans. on Robotics and Automation*, Vol. 20, No.1, pp. 36-44.

Csorba, M., 1997, *Simultaneous Localisation and Mapping*, PhD Thesis, Department of Engineering Science, University of Oxford, Oxford, UK..

Gabriel, G. W., 2003, "Desenvolvimento de Software para Acionamento de uma Plataforma Móvel Autônoma com Microcontrolador", *Undergraduate Final Project*, Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil (in portuguese).

Gabriel, G. W., 2005, *Construção de uma Plataforma Móvel Contendo Múltiplos Computadores Embarcados e Utilizando Sensores de IR para Movimentação em Ambientes Estruturados e Pré-Definidos*, MSc. Thesis, Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil (in portuguese, in preparation).

Leonard, J. J., Durrant-White, H. F., 1991, Mobile Robot Localization by Tracking Geometric Beacons, *IEEE Trans. on Robotics and Automation*, Vol. 7, no. 3, pp. 376-382.

Nascimento Jr., C. L., Yoneyama, T., 2000, *Inteligência Artificial em Controle e Automação*, Ed. Edgard Blücher, São Paulo, http://www.ele.ita.br/ia_contaut/ (in portuguese).

Yoshitome, F. E., 1997, "Planejamento de Trajetórias para Robôs Móveis", *Undergraduate Final Project*, Aeronautics Institute of Technology (ITA), São José dos Campos, Brazil (in portuguese).

8. Responsibility notice