Autopilot – An Autonomous Navigation System

David J. F. Vaz, Filipe A. V. João, António J. Serralheiro
Academia Militar
Lisboa, Portugal

António J. Serralheiro, José A. B. Gerald
Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento (INESC-ID)
Lisboa, Portugal

José A. B. Gerald
Instituto Superior Técnico (IST)
Universidade de Lisboa (UL)
Lisboa, Portugal

***Abstract* – In this work an autonomous navigation system to empower a ground mobile robot to drive itself autonomously on a predefined course was developed. The navigation system implementation is described and relevant testing experiments are presented. Furthermore, graphical users interface (GUI), comprising the entire system positioning, navigation and control systems was developed for the commodity of the users. This GUI also provides an interface to allow the robot movement to be tracked on a map. The motion control is achieved using several 1D and 2D Kalman filters, where the different IMU sensors signals are combined and filtered, in order to obtain a probabilistic evaluation of the position of the robot platform. Evaluation tests confirming the ability of the robot to autonomously reach the pre-designated points are presented, showing that the validity of the approach.**

Keywords - autonomous navigation; estimated position; Kalman filters; noise filtering; motion sensors.

I. INTRODUCTION

Ever since the beginning of the growing interest shown in robotics, autonomous navigation presented itself as an area of great deal of interest among the scientific community. An autonomous system is a system that can operate for long periods of time without any external control [1]. To accomplish autonomous navigation one needs to know where to go (navigation), from where (location) and how (control). In order to provide the required navigation data, sensors as the Inertial Measurement Unit (IMU) sensors, GPS receptor and encoders, are commonly used. As these signals are corrupted with noise, their post-processing is a main issue.

In Europe, advances in this area were stimulated by the EUREKA Prometheus Project (PROgramME for a European Traffic of Highest Efficiency and Unprecedented Safety), in the end 90s, which produced three twin autonomous vehicles – the VaMP, the Vita and the ARGO [2]. Also in the USA several autonomous vehicles have been produced, as for instance the MDARS project, presently in use in the US army [3]. Among the most sophisticated studies known today, lays the vSLAM (Visual Simultaneous Localization and Mapping) system [4] from ActivMedia Robotics and Evolution Robotics. With the inclusion of the GPS, motion sensors and high signal processing capacity, today autonomous navigation systems can be made very complex, as those of iRobot [5] and NASA [6]. Many of the developed autonomous robots today are for military and security purposes, transferring to a machine (robot) the goal of acquire/process required data/actions without the risk of losing Human lives.

The robot used in this study was the DrRobot Jaguar 4x4 Wheel, belonging to the Portuguese Military Academy[1]. This robot is shown in Fig. 1. It is powered up by four 80W motors, one for each wheel, and has an outside rigid deck, weighting less than 20 kg with a maximum speed of 15 km/h. It has a ground clearance of 88 mm, being compact and waterproof. The Jaguar 4x4 Wheel has the ability to climb elevations not exceeding 155 mm and can climb stairs with an elevation less than 110 mm. It has an integrated GPS system, outdoor sensors and 9 DOF (degrees of freedom) IMU (Gyro / Accelerometer / Compass magnetic). It also has a built-in video system and high resolution audio. The communication with the robot is handled by wireless technology (IEEE 802.11G). Although providing real time information of its movement, these signals are highly corrupted by noise, making difficult to assess its actual location. The robot's estimated position at each time is calculated using four two dimensional Kalman filters (2D KF). However, before this implementation, both the IMU sensor data and the encoder data were filtered and calibrated. This calibration allows the reduction of the reading errors, thus leading to the decrease of the existing noise, that was filtered using three one-dimensional (1D) KFs. To control the movement of the robot it was necessary to process the angles of the current direction and of the desired direction, obtained by gyro and magnetic compass. Depending on the error between these two angles, the correct values to send to the robot motors were calculated. The developed algorithm uses the general model of Siegwart [7].

[1] This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

978-1-5090-4565-5/16/$31.00 ©2016 IEEE

Fig. 1. Jaguar 4x4 Wheel Robot.

This paper is structured as follows: in sec. II the navigation system is addressed, presenting the GUI and the navigation procedure. In sec. III experimental results are shown, confirming the good performance of the autonomous navigation system developed. Finally, in sec. IV, conclusions and comments on the obtained results are addressed.

II. NAVIGATION SYSTEM

A. Graphical User Interface

The navigation and control application were programmed in C# using Microsoft Visual Studio 2010 with .Net framework 3.5. Graphical construction, research and data analysis were performed with Matlab R2012b. The program is divided into two modules, the first one relates to connections and control and the second one relates to the navigation. The first program module was used as a base system for the manual control of the robot. This interface allows to create a virtual command and a connection with the robot, and enables a real-time change in the calibration variables of the KFs. This interface was also used in control tests, such as testing the relationship between angles and engine power. The second program module is the navigation module. This interface (Fig. 2) allows a graphically view of relevant information, to change navigation variables, to establish connections with independent modules on the robot and to support all internal algorithms needed for autonomous navigation. It also establishes the connection and the reading of the IMU sensors and GPS. Those values are calibrated and filtered, which allows obtaining the estimated location of the robot in real time. All calculations for autonomous navigation are still performed, concerning angles, distances, the motors' power, etc. All data are stored for later study and processing. The GUI allows the users to see almost all relevant motion parameters, such as: the motor power levels sent at every correction; the battery voltage level; the motor temperature; the present direction angle; the direction error, obtained between the present direction and the desired one; the present distance to the ideal trajectory; the distance between the estimated position and the final position to be attained and the GPS output for the latitude and longitude.

Fig. 2. Graphical interface for navigation.

This GUI and the stated Matlab analysis must be performed in real time, so, a PC with capability to run Microsoft Visual Studio 2010 with .Net framework and Matlab is needed to do the control of the robot. This computer could be attached to it making the robot a completely autonomous system. After further development, it would be even possible to adapt the Matlab algorithm to run in an executable format and implement the guidance system into a SoC, and then include it into the robot.

B. Navigation System

The navigation system comprises the following modules: the Kalman filter, the navigation algorithm, the attitude estimation and the motion control.

The Kalman Filter

As mentioned before, the data filtering for position estimation and noise reduction is performed with linear Kalman filters. An overview of the Kalman filtering process is depicted in Fig. 3 [8].

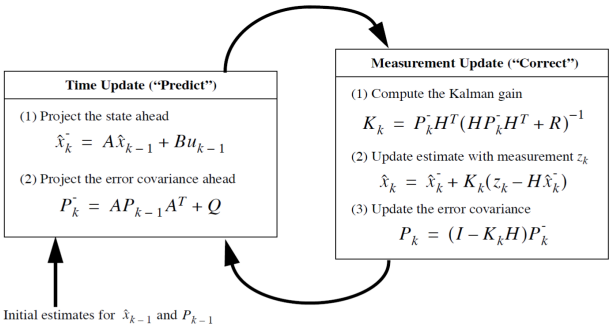

Fig. 3. Kalman filter algorithm. [8]

The choice of using this filter is justified mainly by its both simplicity of implementation and good performance in process prediction and noise reduction. Actually, the ability of this filter to predict future values makes it the adequate tool to use in this real time controlling system, avoiding the characteristic delays introduced by other filtering methods that are harmful to the navigation performances.

It is well known that the best way to consider all variables and the nonlinearities of the system would be using a nonlinear filter such as the Extended Kalman Filter (EKF). However, in our implementation, this was avoided due to several well-

known issues [9], namely: nonlinear filtering can be difficult and complex and it is not as well-understood as linear filtering; to use an EKF one needs a mathematical system representation model; accurate system noise models are needed and, at last but not least, more computational resources are necessary. In fact, although a mobile robot is not a linear system, it can be fairly approximated by a linear one and, therefore it is expected that linear estimation approaches would give good results.

Navigation Algorithm

This module is divided into three main parts. The first one addresses the connection and control mechanisms. After a successful connection to the robot the user can manually control the robot, either with the original application and newly implemented one, using a joystick. In this control part it is possible to change the variance (Q) and minimum noise matrices or values (R) used in the KF, in real time.

The second part addresses the connection of the sensors and the processing of their data. The routine for the sensors connection was provided from the original API and it creates two *threads* that run concurrently, one for GPS and one for the IMU sensors. The embedded IMU provides data from three different sensors: accelerometer, compass and gyroscope. The gyroscope and accelerometer are initially calibrated and in the case of the accelerometer its data is filtered using three 1D KFs. Using the data from the magnetic compass and the gyroscope, the steering angle is obtained, based on magnetic north and gyro information. A system based on *dead reckoning* is then applied to obtain the estimated position. The GPS values of latitude and longitude are converted to a XY referential. Eventually, the data can be recorded and then we can return to the starting point, waiting for new data from all these sensors.

The last part, the autonomous navigation operation and control, is exclusively dedicated to the autonomous navigation. The KFs receive as inputs the acceleration, velocity and position samples, outputting the estimated position of the robot. After this estimated position, the distance to the desired destination and the distance to the desired path are then evaluated. The direction error, i.e. the difference between the robot orientation and direction which it should go, is also evaluated. This error allows us to control the robot motors, moving it through the programmed path.

Attitude Estimation

An initial calibration of the accelerometer sensor is required in order to produce accurate values, performed with the robot stopped. The nonzero initial mean values are thus subtracted from the sensors output, correcting the accelerometer output. As expected, the accelerometer signal is quite noisy, so it should be processed by a 3D KF. However, to simplify the calculations (thus lowering the computing requirements), three independent unidimensional KF are used for each one of the three dimensions. Fig. 4 depicts instantaneous acceleration (in red) and the filtered data (in blue).

After performing the sensors calibration and after filtering the accelerometer data, the estimation of the instantaneous velocity is done. Since this velocity data, estimated from the accelerometer, has a high inherent error and also because the velocity data obtained from the GPS is noisy, it was decided to use the velocity calculated from the wheel encoders and filtered by another dedicated KF. The estimated velocity takes into account a weighted average of the number of turns of the left and right wheels, multiplied by their perimeter, performed at each second.

Fig. 5 depicts velocity estimation obtained by the three above mentioned methods, in a circular path: in red from the accelerometers, in green using the GPS data and in blue the one using the wheel encoders as detailed above.

The estimated position, calculated from the velocity and direction estimations, will drift from the actual position over time. But combining it with the measurement of the GPS will bring the estimate closer to the actual value without becoming noisy and without sudden jumps in each measurement. To perform this, a 3D KF (position, velocity, acceleration) could have been used. However, the level of complexity and the required computing time would be very penalizing, so it was decided to use two 2D filters and optimize each one individually. The first filter calculates the velocity using a 2D KF (velocity and acceleration). This filter uses the original velocity obtained by the data encoder (blue line in Fig. 6) and the acceleration given by the accelerometer (red line in Fig. 6). The estimated velocity (black curve in Fig. 6) is obtained as output. The second filter uses this more accurate velocity information to estimate the position data. In order to work on a horizontal plane, two filters are required for the X axis and two filters for the Y axis (velocity and position), since the robot is not supposed to laterally slide.

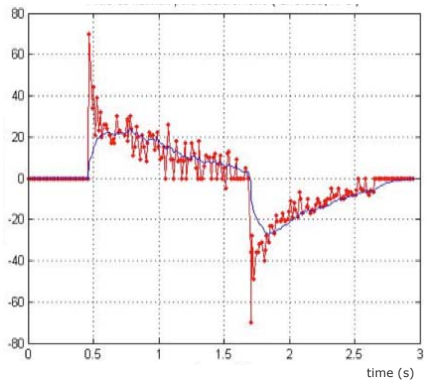

Fig. 4. Kalman filter results with (Q=0.065 and R=5). Original data (red) and filtered data (blue).

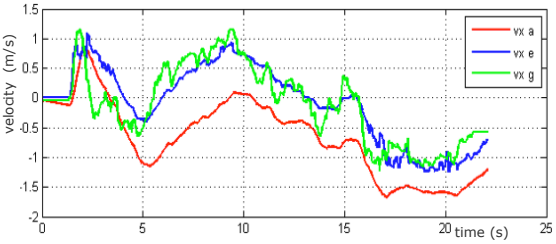

Fig. 5. Velocity obtained by means of the accelerometer (red), the GPS (green) and encoders (blue).

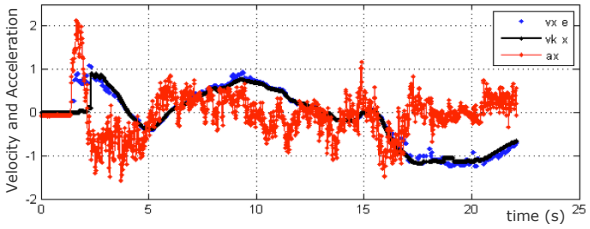

Fig. 6. Estimated velocity filtered by the 2D Kalman filter.

As shown in Fig. 7, using the previously calculated velocity and the GPS signal with a high induced noise (red) an estimate of the instantaneous position of the robot (black) was obtained. Also in Fig. 7, we can compare the estimated position (blue) with the desired position (green).

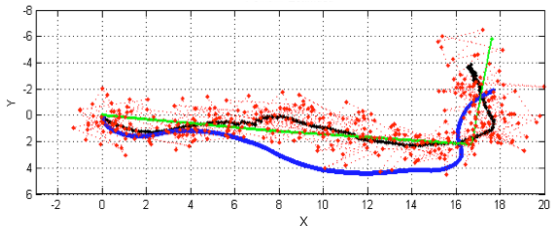

Fig. 7. Kalman filter estimated position.

Robot Motion control

To control the robot movement, the values of the magnetic compass and gyroscope are used in order to obtain the angle between the magnetic north and the direction in which the robot is oriented (*yaw*). It is also necessary to calculate the driving angle which represents the intended direction. A heuristic process analyses these angles, together with the distance from the local position to the desired path or final position and establishes the actual course to be pursued.

The implemented algorithm communicates and controls the robot in real time, receiving the sensors data, processing them and sends back motion data to control the robot electrical motors, using these time slots:

1. Update of the motor control information – 100 ms;
2. Sensors waiting response time:
    a. GPS – 200 ms;
    b. Accelerometer and gyroscope – 20 ms;
    c. Magnetic compass – 220 ms;
3. Position estimation periodicity - 56 ms (average time);

It is worth noting that, the communication delay time (wireless 802.11) with the robot is negligible.

III. Experimental Results

Several experiments were performed to evaluate the implemented robot autonomous navigation system. In the following pictures, we present the desired path (in green) and, the travelled path (in red), in blue is shown the path obtained by the nonfiltered GPS data. The parameters under evaluation were the estimated positions (generated by the default filter in the original robot control program and the KF), the error direction and the robot velocity / acceleration at each moment. The navigation was performed at approximately 60% of the robot's total power.

In the first test, a rectangular path with 15 m of width and 25 m of length was defined as desired trajectory (Fig. 8). This test was important because it allows studying the robot autonomy for long journeys with abrupt changes of direction along with the ability of the algorithm to estimate a good location, thereby obtaining the smallest possible error along the path. It can be concluded from Fig. 8 that the navigation performance is of very good accuracy in both our and original navigation systems. In Fig. 9 it can be seen in detail the actual and desired path in the implemented algorithm. With these two tests is possible to recognize that both methods are accurate despite the GPS deviation.

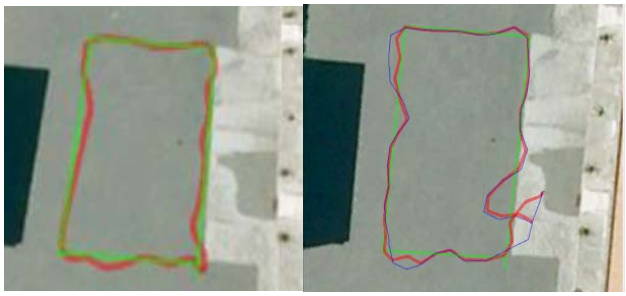

Fig. 8. Rectangular path with ideal conditions (on the left with our navigation system and on the right with the robot's original program).

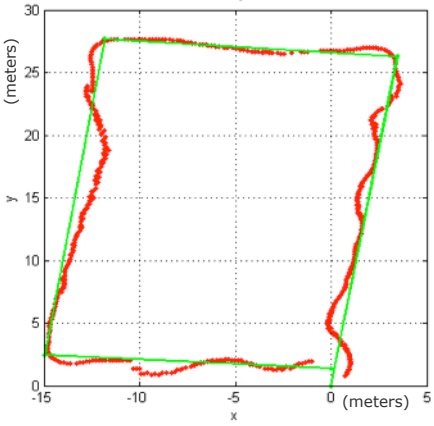

Fig. 9. Rectangular path with ideal conditions zoom. (KF algorithm)

In Fig. 10, the angular path direction (ranging from -π to +π) and its error are depicted and the large variations occurring in 17, 25 and 40 seconds are explained by the sudden change of direction, imposed by the programmed desired path. In Fig. 11, the absolute error distance along the programmed path is shown and it can be seen that it is always lower than 1.3 m. From these results one can conclude that the robot executes its mission in a highly satisfactory way, although some improvements can still be done, namely in the sudden changes of direction. This indicates that (i) the KF may be restricting the sudden change of position values and (ii) there is a delay in the

motor control, between the calculations of the estimated position and the errors of the angles and the values that are sent to the engines. Issue (i) can be attenuated by a more suitable tuning of the KF parameters and issue (ii) can be minimized with a faster processing unit, for instance, an on-board digital signal processor (DSP) instead of the general purpose PC used.

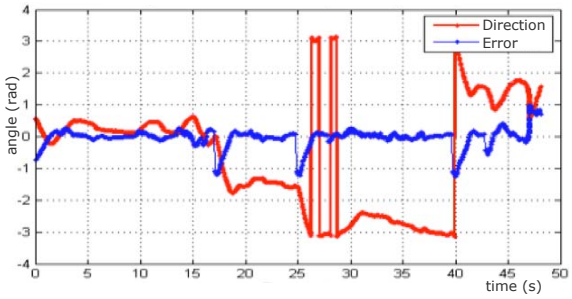

Fig. 10. Robot path angular direction (red) and its error (blue).

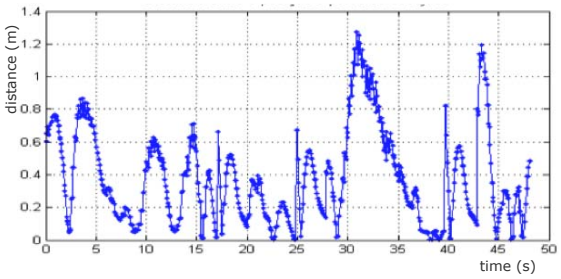

Fig. 11. Absolute distance between the ideal and the actual path.

In another experiment, a 10 point polygonal path was predefined, in order to build up a circular trajectory of approximately 5 meter radius (Fig. 12, green line). As can be seen in Fig. 12 (left, red line) the robot executed a path quite close to the desired one, which was much better than the original program (on the right).

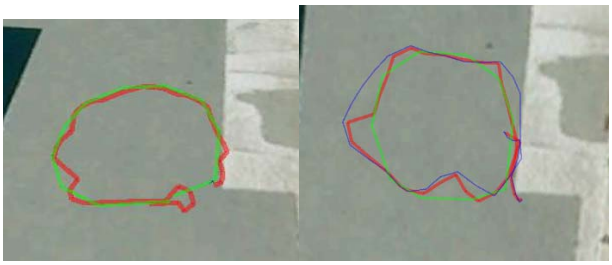

Fig. 12. Circular path with ideal conditions (on the left test with KF, on the right test with the robot's original program).

Analyzing the GPS signal and the supplied position by the KF, one can see in Fig. 13 that the KF method is much smoother, effectively filtering the GPS data discontinuities. This is due to the filter time integration of the accelerometer and encoder data (acceleration and velocity) and its prevention that successive positions may present high discrepancies, and so allowing a better approximation of reality. In all the various tests performed, it was observed that the robot accurately followed all the previously defined paths.

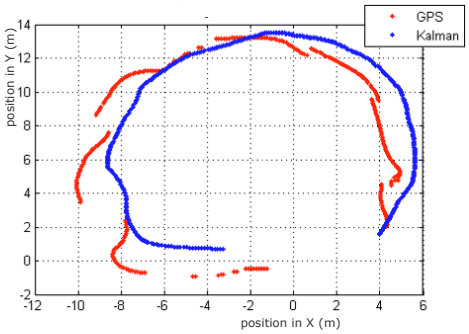

Fig. 13. GPS and Kalman filter supplied positions.

IV. CONCLUSIONS

An autonomous navigation system for the Jaguar 4x4 wheel robot was developed and tested. Besides the motion control algorithm, a convenient GUI was developed. The algorithm receives information from the IMU sensors and the GPS, performs the required signal processing (using several linear Kalman filters) and feeds back the motor control data. Performed field trials showed the need of a faster digital signal processing tool (other than the used general purpose PC) and further research in the Kalman filters and sensors tuning. For these reasons and because the motion control procedure is always an open research subject, this work is not closed and will be continued. Nevertheless, in all experiments, the robot followed quite well the predefined path, and therefore, it can be anticipated future promising results.

REFERENCES

[1] G. Bekey, Autonomous robots : from biological inspiration to implementation and control, Massachusetts: The Mit Press, 2005.

[2] "EUREKA Prometheus Project:all facts at a glance," Maio 2010. [Online]. Available: http://vionto.com/show/info/en/EUREKA%20Prometheus%20Project/EUREKA. [April 2013].

[3] "Autonomous Robotics Programs," General Dynamics - Robotic Systems, [Online]. Available: http://www.gdrs.com/robotics/programs/program.asp?UniqueID=27. [April 2013].

[4] N. Karlsson, L. Gonçalves e M. Munich, The vSLAM Algorithm for Navigationin Natural Environments, Evolution Robotics, Inc., 2012.

[5] iRobot, "IRobot," Irobot Corporation, 2013. [Online]. Available: http://www.irobot.com/en/us/learn/defense.aspx. [May 2013].

[6] B. D. -. NASA, "NASA," 28 Julho 2013. [Online]. Available: http://spaceref.com/nasa-hack-space/2013-sample-return-robot-challenge.html. [2013].

[7] R. Siegwart and I. R. Nourbakhsh, Introduction to Autonomous Mobile Robots, London: Massachusetts Institute of Technology, 2004.

[8] G. Welch and G. Bishop, "An introduction to the kalman filter. 2006," University of North Carolina: Chapel Hill, North Carolina, US, 2006.

[9] D. Simon, "Using nonlinear kalman filtering to estimate signals," Embedded Systems Design, vol. 19, no. 7, p. 38, 2006.