Code migration from a realistic simulator to a real wheeled mobile robot

José Gonçalves, José Lima, Paulo Malheiros and Paulo Costa

***Abstract*— This paper describes the code migration from a realistic simulator to a real wheeled mobile robot. The robot software consists in the localization and navigation of an omnidirectional robot in a structured environment. The localization estimate is achieved by fusing odometry and infra-red distance sensors data, applying an extended Kalman filter.**

I. Introduction

This paper describes the code migration from a realistic simulator to a real wheeled mobile robot. Code migration from realistic simulators to real world systems is the key for reducing the development time of robot control, localization and navigation software [1]. For this purpose it was developed a realistic simulator (available for download at [2]). Due to the inherent complexity of building realistic models for the robot, its sensors and actuators and their interaction with the world, it is not an easy task to develop such simulators.

The developed robot software consists in the localization and navigation of an omnidirectional robot in a structured environment. The robot is equipped with brushless motors and infra-red distance sensors. The localization estimate is done by fusing odometry and the distance sensors data, applying an extended Kalman filter. In the first place it is presented how to develop robot code and how to migrate it to the real robot, then it are presented the distance sensor modeling, the absolute position estimation and the localization algorithm based on a kalman filter. Finally some conclusions are presented.

II. Code migration to a real robot

A. Code generated with the simulator

Initially, the localization and navigation software is generated with the simulator (Figure 1). The simulator provides to a Remote application the distance sensors data with noise as modeled in Section III [3], the encoders data and the real robot position. The Remote application executes the localization and navigation algorithms and returns the speed references for each wheel to the simulator. The applied communication protocol to exchange data between the Remote application and the simulator is UDP, as shown in Figure 2. As an example, a robot trajectory produced in the simulator is shown in Figure 3. The pose estimate error and its variance are also shown in Figure 4.

José Gonçalves and José Lima are with the Polytechnic Institute of Bragança, Department of Electrical Engineering, Bragança, Portugal {goncalves,jllima}@ipb.pt

Paulo Malheiros and Paulo Costa are with the Faculty of Engineering of University of Porto, DEEC, Porto, Portugal {paulo.malheiros,paco}@fe.up.pt

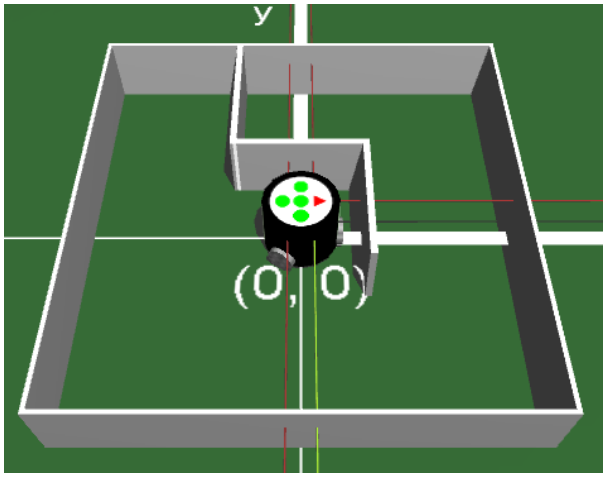

Fig. 1. Robot simulator snapshot.

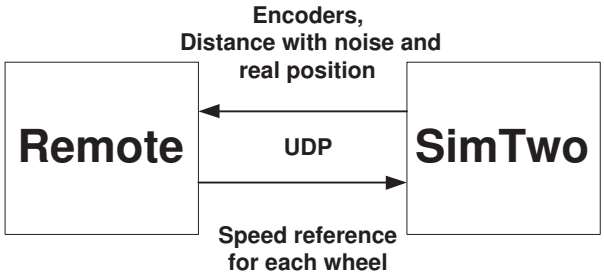

Fig. 2. Simulator communicating with the Remote application

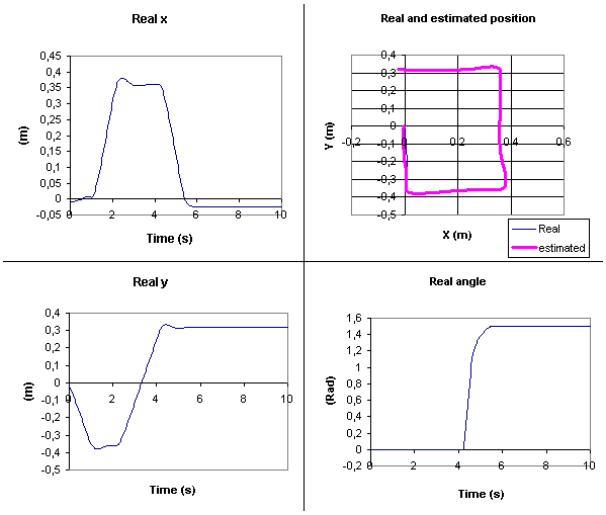

Fig. 3. Simulated robot trajectory

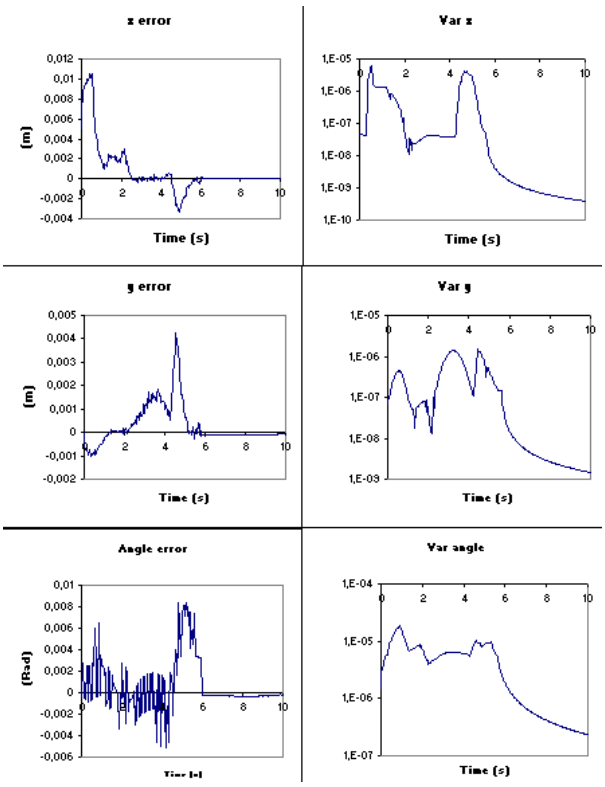

Fig. 4. Simulated robot estimate error and variance

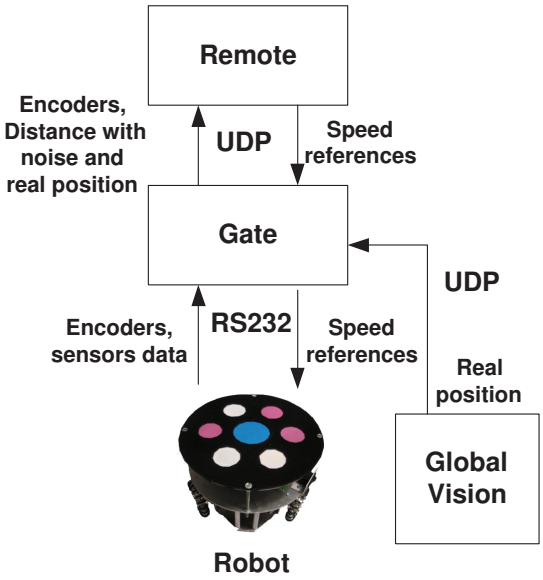

Fig. 5. Real world system architecture

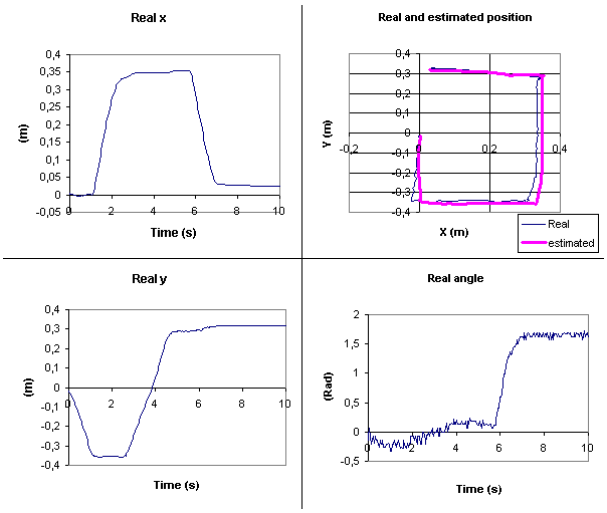

Fig. 6. Real robot trajectory

B. Developed code applied to the real robot

The architecture used to migrate the robot code to the real robot is presented in Figure 5. The Remote Application is shared with the simulation so that the generated code can be applied to the real robot without any changes. The robot real position is provided to the Gate application at a 25 Hz rate, by a global vision system described in [4] (with the difference that was used only the absolute measurements without applying the Kalman filter). The control loop is initiated by the robot when it sends to the Gate application, via RS232, the encoders and the sensor data at a 25 Hz rate. Then, the Gate application provides to the Remote application the distance sensors data, the encoders data and the real robot position via UDP. The Remote application executes the localization and navigation algorithms and returns to the Gate application the speed references of each wheel. Finally, the speed references are sent to the real robot via RS232 protocol. The trajectory executed with the real robot, shown in Figure 6, is similar to the simulated making the simulation very useful, because it allows to reduce considerably the development time of the robot software. The real robot trajectory estimate error and variance are shown in Figure 7.

III. Distance Sensors Modeling

The Sharp family of infra-red range finders is very popular for robotics distance measurement applications. Some drawback of these sensors are their non-linear response and the mandatory minimum distance measurement requisites. The presented study is about the Sharp infra-red distance sensor GP2D120. In order to model the distance sensor it was necessary to collect a considerable amount of data, for this task it was used the industrial robot ABB IRB 1400 to place an obstacle for different distances as shown in Figure 8. Industrial robots allow executing repetitive operations normally performed by human operators, without getting bored and without loosing precision [5]. The introduction of an industrial robot to place the obstacle in different known positions allows to increase the speed, repeatability and reduces errors in the process of distance sensor data collecting.

The sensor data is acquired using the internal analog to digital converter (ADC) of the Atmel AVR ATMega8 with 8 bit precision. At each mobile robot sample time the ADC registers 10 samples for each sensor, which are added and sent to a personal computer. In order to evaluate the sensor noise it are registered 256 mobile robot sample times data for each distance. As the used analog to digital converter was the provided internally with the micro-controller ATMega8, and since there is available an internal reference voltage of 2.56 V (V_{ref}), it is possible to have a precision increase using this reference, when compared to the alternative of using an external reference voltage of 5 V. To use this approach, a voltage divisor must

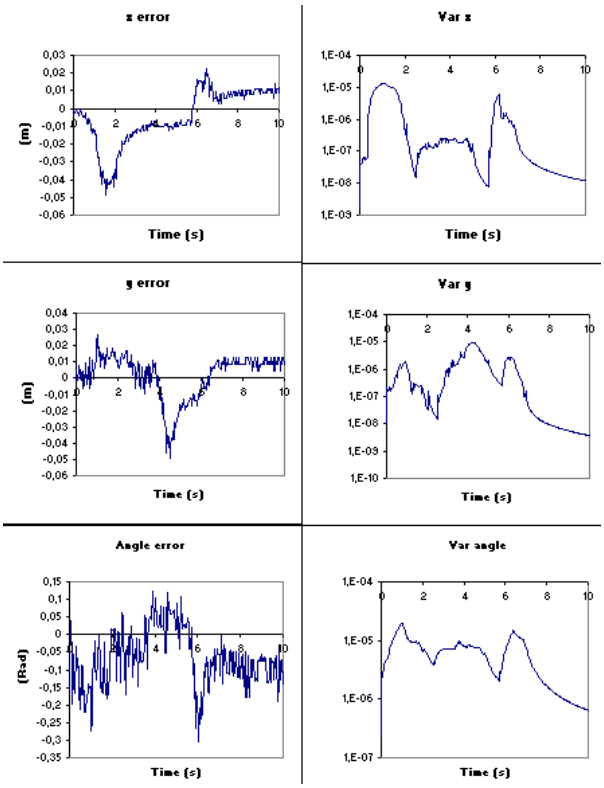

Fig. 7. Real robot estimate error and variance

be applied in order to lower the sensor voltages to values below the converter reference (2.56 V), since its maximum is nearly 3.2 V. This is important if the user wants to use sensor values from 7 to 10 cm. If the user makes the choice of using a minimal distance of 10 cm then it is not necessary to apply a voltage divisor because the value for 10 cm corresponds to nearly 2.33 V, which is below the internal converter reference, and the voltage decreases with the distance. For this application it was considered that it was important to use the sensor range from 7 to 100 cm, thus a voltage divisor was applied. The obtained voltage characteristic of the infra-red distance sensor is presented in Figure 9. It was calculated resorting to equation 1, where v is the voltage, si is the *ith* sample, n is the number of acquired samples and V_{ref} is the micro-controller internal reference voltage.

$$v = V_{ref}(\frac{\sum si}{n.255}) \tag{1}$$

The relation between the inverse voltage and the distance can be approximated to a line as shown in Figure 10, where the real and the approximated curve are presented. The used values to achieve the presented curve were from 7 to 100 cm, taking in account the chosen sensor minimal distance.

In order to obtain the distance in the real robot, having in mind the shown approximation, equation 2 can be applied, where d is the distance expressed in m and v is the sensor voltage, k_1 equals 0.1881 and k_2 equals 4.6779.

$$d = \frac{\frac{1}{v} - k_1}{k_2} \tag{2}$$

In order to obtain the distance variance it was made

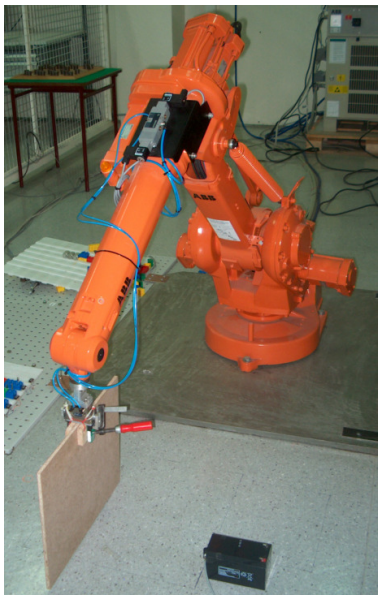

Fig. 8. IRB 1400 placing the obstacle.

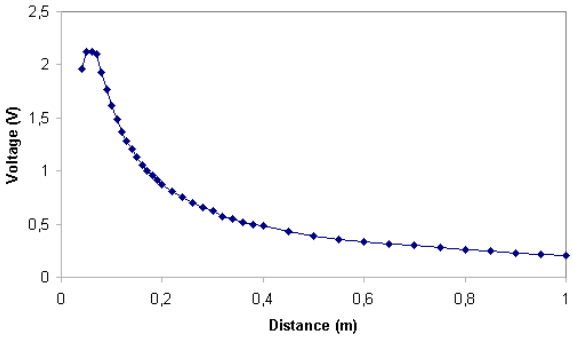

Fig. 9. IR distance sensor characteristic.

the approximation shown in equation 3, with a different derivative for each distance, where m is the voltage derivative presented in equation 4 and in Figure 12. The voltage equation was obtained from the approximation presented in equation 2.

$$v = m.d + b \tag{3}$$

$$m = \frac{-k_2}{(d.k_2 + k_1)^2} \tag{4}$$

This approximation was made in order to obtain the distance variance related with the known voltage variance.The simulated infra-red distance sensors provide the distance with the noise. Its variance is shown in Figure 13, which was obtained resorting to equation 5 applying the approximation of the voltage variance presented in Figure 11.

$$var(d) = \frac{Var(v)}{m^2} \tag{5}$$

An example of two simulated sensors for two different distances is shown in Figure 14. In the example both measures are presented with and without noise.

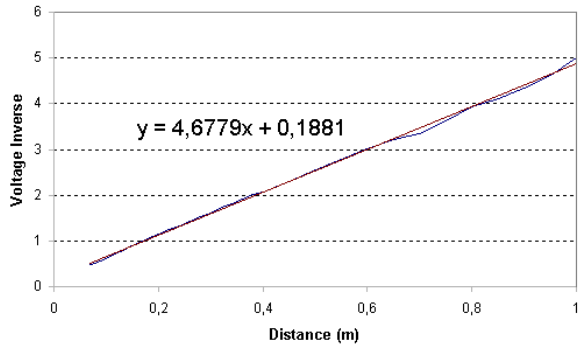

Fig. 10. Voltage Inverse VS Distance.

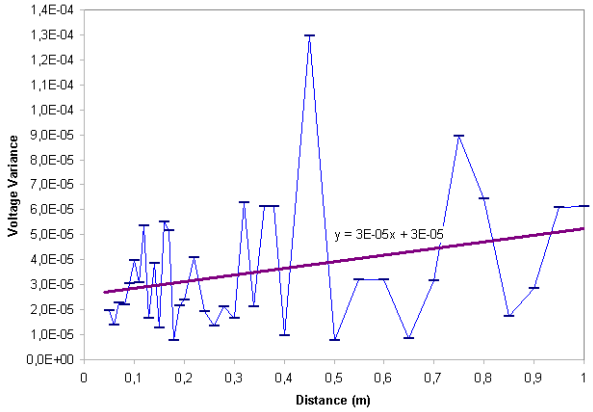

Fig. 11. Voltage variance.

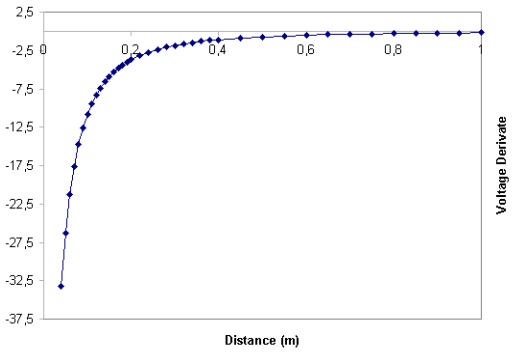

Fig. 12. Voltage derivative.

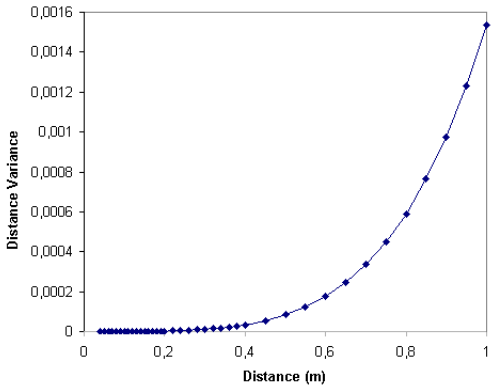

Fig. 13. Simulated sensors variance.

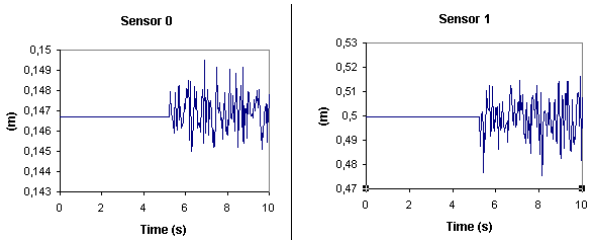

Fig. 14. Simulated sensors with and without noise

IV. Absolute position estimation

In order to extract the absolute position estimation it is necessary to estimate the x, y and θ. The robot disposes of three pairs of Sharp infra-red distance sensors, each par is capable of estimative each absolute position parameters. For this purpose the robot is given, initially, a correct position estimative and a map of its environment.

A. Simulator world construction

The robot, the on-board sensors and the environment are described using the standard eXtensible Markup Language (XML). The robot and its environment are shown in Figure 1. The developed environment has only 90 degrees angles being a possible representation of the real world where our buildings are mostly 90 degrees and usually very artificial [6]. The environment is built using blocks and the robot is a three wheel omnidirectional robot equipped with three pairs of infra-red distance sensors. The intersection of a sensor beam with a block returns a distance with noise.

B. Robot map

The robot knows its environment by knowing a Map, but instead of blocks the Map is composed by lines with the following parameters if it is an horizontal line:

- y position
- minimum x
- maximum x
- VFB2T - visible from bottom to top (boolean)

and by the following attributes if a vertical line:

- x position
- minimum y
- maximum y
- VFL2R - visible from left to right (boolean)

The environment is sensed with distance sensors characterized by the following parameters:

- Angle
- x position
- y position

The sensor parameters are in a local robot referential.

C. Information extracted from the Map

Since the robot is provided with a new position estimate at each sampling time (each 40 ms), it is possible to predict, for each sensor, a distance and the line that the sensors beam is expected to hit. In order to obtain, at each sample, the sensor parameters in the world referential it is necessary, in the first place, to offset the sensor position with the robot position estimate and then rotate the obtained position with the estimated robot angle. Finally the new angle sensor is the angle sensor summed with

the estimated robot angle. To obtain the distance to a line and the expected line that the beam sensor is hitting it is necessary to use equation 6.

$$(x_l, y_l) = (x_s, y_s) + d(u, v) \tag{6}$$

where:

- x_l - x position in the line where the beam is hitting
- y_l - y position in the line where the beam is hitting
- x_s - x position of the sensor in the world referential
- y_s - y position of the sensor in the world referential
- d - measured distance
- ϕ - Angle sensor parameter summed with the estimated robot angle
- u - $cos(\phi)$
- v - $sin(\phi)$

The shown parameters are illustrated in Figure 15.

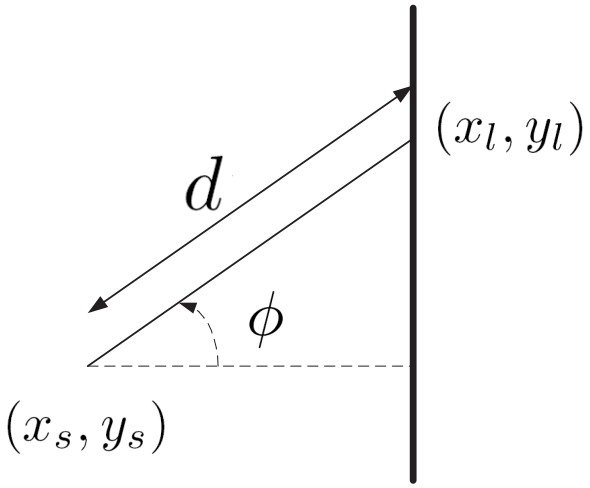

Fig. 15. Sensor hitting a wall

As an example, for a vertical line, as the distance in the x axis is a known constant it is possible to assume that:

$$d = \frac{x_l - x_s}{u} \tag{7}$$

Knowing d, from equation 7, it is possible to obtain y_l by the following equation:

$$y_l = y_s + dv \tag{8}$$

For an horizontal line the calculus process is similar to the example described for an vertical line. Repeating this process for all the robot Map lines (both horizontal and vertical), it is possible to know the distance and the expected wall that the sensor is expected to hit, by choosing the lowest positive distance.

D. Robot position estimation

The used approach to estimate the robot absolute position is valid only if the pair of sensors is expected to hit the same wall. If one of the sensors is expected to hit a different wall then it is considered that the estimative has an variance higher enough so that the filter neglects its contribution. This assumption is valid both for the angle and for x and y estimation. As an example it will be shown the calculation of the robot position estimation of a pair of sensors hitting a vertical wall as shown in Figure 16.

The pair of sensors has the following parameters in the local robot referential:

- x position - 0 for both sensors
- y position - L_s for Sensor 1 and $-L_s$ for Sensor 2
- $angle$ - 0 for both sensors

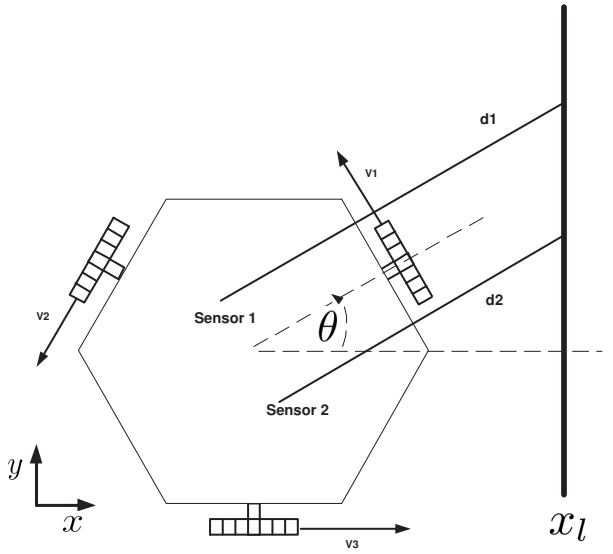

Fig. 16. Pair of sensors hitting a vertical wall

1) θ calculation: As both sensors are seeing the same wall it is possible to estimate the robot angle resorting to equation 9.

$$\theta = \arctan(\frac{d1 - d2}{2L_s}) \tag{9}$$

The horizontal walls also provides angle information. All the available information provided by the three pairs of sensors will be fused in order to obtain an optimal estimation.

If the expected distance of one of the sensors differs of more than 1.2 cm from the real measured distance, then it will be considered that there is no trust in the angle estimation, and its variance will be very high.

2) x and y calculation: As the two sensors are seeing the same wall it is possible to estimate the robot x position resorting to equation 10.

$$x = x_l - \frac{d1 + d2}{2}\cos(\theta) \tag{10}$$

A vertical wall do not provides information for y position so it is considered for this parameter that the variance is high enough that its contribution is negligible when fusing information from the different pairs of sensors.

If the expected distance of one of the sensors differs of more than 5 cm from the real measured distance, then it will be considered that there is no trust in the x or y estimation, and its variance will be very high.

3) Fusion: As there are available three pairs of sensors, there are three estimates for each parameter that must be fused in order to obtain an optimal estimate. To fuse the several estimates it is necessary to apply equation 11.

$$p = \frac{\frac{ps1}{var(ps1)} + \frac{ps2}{var(ps2)} + \frac{ps3}{var(ps3)}}{var(ps1)^{-1} + var(ps2)^{-1} + var(ps3)^{-1}} \quad (11)$$

where $var(psi)$ is the variance for a parameter given by the par of sensors i and p is the resulting optimal parameter estimation. All the robot position parameters depend on the distances $d1$ and $d2$. It is possible to approximate the parameters estimation variance as shown in equation 12 [7].

$$var(p) = \frac{\partial p}{\partial d_1}^2 var(d1) + \frac{\partial p}{\partial d_2}^2 var(d2) \quad (12)$$

V. Odometry and distance sensors data fusion

Odometry and infra-red distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions [8][9].

A. Extended Kalman filter algorithm

With the kinematic dynamic model given by equations system (13) and Figure 16.

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} = \begin{pmatrix} -sin(\theta) & cos(\theta) & L \\ -sin(\frac{\pi}{3} - \theta) & -cos(\frac{\pi}{3} - \theta) & L \\ sin(\frac{\pi}{3} + \theta) & -cos(\frac{\pi}{3} + \theta) & L \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ w \end{pmatrix} \quad (13)$$

and considering that control signals change only at sampling instants, the state equation is:

$$\frac{dX(t)}{dt} = f(X(t), u(t_k), t), t\epsilon[t_k, t_{k+1}] \quad (14)$$

Where $u(t) = [V_1 V_2 V_3]^T$, that is, the odometry measurements are used as kinematic model inputs. This state should be linearized over $t = t_k$, $X(t) = X(t_k)$ and $u(t) = u(t_k)$, resulting in:

$$A^*k = \begin{pmatrix} 0 & 0 & \frac{-\sin(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\cos(\theta)}{2(1+\cos(\frac{\pi}{3}))} \\ 0 & 0 & \frac{\cos(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\sin(\theta)}{2(1+\cos(\frac{\pi}{3}))} \\ 0 & 0 & 0 \end{pmatrix} \quad (15)$$

with state transition matrix:

$$\phi^*(k) = exp(A^*(k)(t_k - t_{k-1})) \quad (16)$$

Resulting in:

$$\phi^*k = \begin{pmatrix} 1 & 0 & (\frac{-\sin(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\cos(\theta)}{2(1+\cos(\frac{\pi}{3}))})T \\ 0 & 1 & (\frac{\cos(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\sin(\theta)}{2(1+\cos(\frac{\pi}{3}))})T \\ 0 & 0 & 1 \end{pmatrix} \quad (17)$$

Where T is the sampling time $(t_k - t_{k-1})$.

Thus the observations are obtained directly, H^* is the identity matrix.

The extended Kalman filter algorithm steps are as follows [10] [11]:

1) State estimation at time $t = t_k, X(k^-)$, knowing the previous estimate at $t = t_{k-1}$, $X(k-1)$ and control $u(t_k)$, calculated by numerical integration.
2) Propagation of the state covariance

$$P(k^-) = \phi^*(k)P(k-1)\phi^*(k)^T + Q(k) \quad (18)$$

Where $Q(k)$ is the noise covariance (14) and also relates to the model accuracy.
As there is a measure, the follow also apply:
3) Kalman gain calculation

$$K(k) = P(k^-)H^*(k)^T(H^*(k)P(k^-)H^*(k)^T + R(k))^{-1} \quad (19)$$

Where $R(k)$ is the covariance matrix of the measurements.
4) State covariation update

$$P(k) = (I - K(k)H^*(k))P(k^-) \quad (20)$$

5) State update

$$X(k) = X(k^-) + K(k)(z(k) - h(X(k^-, 0))) \quad (21)$$

Where z(k) is the measurement vector and $h(X(k^-, 0))$ is $X(k^-)$.

VI. Conclusions

Odometry and distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions.

Code migration from realistic simulators to real world systems is the key for speeding up the developing time in robot software production. The developed code in the simulator was migrated to a real robot, reducing considerably the development time.

References

[1] O. Michel, $Webots^{TM}$: Professional Mobile Robot Simulation, ISSN 1729-8806, International Journal of Advanced Robotic Systems,volume 1,number 1, 2004.
[2] "SimTwo" http://www.fe.up.pt/$\sim$ $paco/wiki$, 2008.
[3] J. Gonçalves, J. Lima, H. Oliveira and P. Costa, *Sensor and actuator modeling of a realistic wheeled mobile robot simulator,* 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, 2008.
[4] J. Gonçalves, J. Lima and P. Costa *Real time localization of an omnidirectional robot resorting to odometry and global vision data fusion: An EKF approach*, IEEE International Symposium on Industrial Electronics, Cambridge, 2008.
[5] M. Groover, M. Weiss, R. Nagel, N. Odrey *Industrial Robotics: Technology, Programming, and Applications* McGraw-Hill, 1995.
[6] Fire Fighting robot competition FAQ *http://www.trincoll.edu/events/robot/FAQ,* Why does the arena have only 90 degree angles?, Trinity College Hartford Conneticut, 2008.
[7] M. I. Ribeiro, *Gaussian Probability Density Functions: Properties and Error Characterization,* Technical Report, IST, 2004.
[8] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics,* MIT Press, 2005.
[9] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, *Principles of Robot Motion : Theory, Algorithms, and Implementations,* MIT Press, 2005.
[10] G. Welch and G. Bishop, *An introduction to the Kalman filter,* Technical Report, University of North Carolina at Chapel Hill, 2001.
[11] R. Negenborn, *Robot Localization and Kalman Filters - On finding your position in a noisy world,* Master Thesis, Utrecht University, 2003.