Online Virtual Control Laboratory of Mobile Robots

D. Galán [*] E. Fabregas [*] G. Garcia [**] J. Sáenz [*] G. Farias [***]
S. Dormido-Canto [*] S. Dormido [*]

[*] *Departamento de Informática y Automática. Universidad Nacional de Educación a Distancia (e-mails: dgalan@dia.uned.es, {efabregas, jacobo.saenz}@bec.uned.es, {sebas, sdormido}@dia.uned.es).*
[**] *Radar Research and Innovations, 11702 W 132nd Terr., Overland Park, KS 66213, USA. (e-mail: garciagarreton@hotmail.com).*
[***] *Pontificia Universidad Católica de Valparaíso, Av. Brasil 2147, Valparaíso, Chile. (e-mail: gonzalo.farias@pucv.cl).*

Abstract: In STEM subjects, interactive laboratories are one of the most widely used tools for students to acquire practical knowledge. These laboratories allow them to modify system parameters and analyze the outputs in real time. In control engineering, these laboratories include different predefined controllers with which the student must experiment to study their operation. However, these laboratories usually do not include functions that allow students themselves to create their own controller. This work presents an interactive virtual laboratory to control mobile robots developed in JavaScript. Mobile robots are an attractive platform for students where they can analyze, test and understand fundamental concepts that are difficult to explain from a theoretical point of view. This environment allows the student to generate their own experiments (general statements, controllers, steps to follow over time) and test them with the simulator. For example, they can design their own position controllers and they can compare different PID-type control strategies in real time. Besides, this environment is open and configurable, so the teacher can decide the available features for each experiment depending of the learning goals.

Keywords: Mobile robots; Simulators; Control engineering.

1. INTRODUCTION

During the last years, new information and communication technologies have had a big impact in the society (public services, communications, social media, e-commerce, information storage, access and exchange, teaching-learning process, etc.). As part of it, the education have been influenced for this impact at all levels. In this scenario, higher education has had to adapt its model to this new challenging paradigm using different tools: videoconferences, simulations, on-line interactive applications, virtual and remote laboratories, robots, etc. Ausín et al. (2016); Broisin et al. (2017); Halimi and Gillet (2018).

Undoubtedly, the use of this kind of tools has greatly benefited both students and teachers, especially in engineering education. The main idea have been to incorporate modern techniques to introduce important concepts in an innovative way from the academic and didactic points of view. In this context, the control engineering education has introduced the robotics in this novel teaching-learning process. Because robots can be seen as attractive toys from a recreational and playful point of view. But they can also be seen as an excellent example of interesting control problems, like in Peralta et al. (2016); Huang et al. (2016); Browne and Conrad (2017); Wang et al. (2017).

In this sense, simulators are very important to testing control designs and observe the results rapidly Fabregas et al. (2014). Using these simulators, students can test and understand, at any time and pace, and from any place, concepts that are not easy to follow in the classroom. For example, on-line tuning of a PID controller in a position control of a mobile robot experiment (Egerstedt et al. (2001)).

Many universities have developed laboratories using these tools. In most cases, to use these applications, students need to install the software in the computer taking into account its specifications and the Operating System. Also, they don't have web interfaces because they are very heavy and consume lot of processing resources in the local machine, even for simple experiments. Often these characteristics are a disadvantage for its correct use and operation.

Besides, this kind of laboratories allow students to compare different control strategies designed by the professor, modify specific parameters, and view the results in running time like in Fabregas et al. (2011); Ionescu et al. (2013). However, interactive laboratories do not usually allow

⋆ This work has been funded by the Spanish Ministry of Economy and Competitiveness under the Project DPI2014-55932-C2-2-R and by the Chilean Ministry of Education under the Project FONDECYT 1161584.

students to create their own algorithms, so they are not entirely aware of their complexity and internal behavior. Moreover, they do not acquire the practical knowledge to implement a controller by themselves.

Therefore, this work proposes the use of an open laboratory in conjunction with an experimentation environment, with which students can design their own code and whose execution modifies the behavior of the interactive laboratory. In this way, the lab is not overloaded with different control methods, parameters to configure and graphs for displaying results. It merely contains the model of the system to study and its visualization. The primary objectives of this simulator are (1) to enable users to design control algorithms that allow the robot to reach a specific position, and (2) to compare the performance of these algorithms. There are several works in which interactive laboratories have been used for these purposes, as Pastor et al. (2003); Vargas et al. (2008); Potkonjak et al. (2016) points out.

The remainder of the paper is organized as follows: Section 2, presents the model of the robot, the control problem and its solution used to teach control; Section 3 presents the implementation of the simulator; Section 4 presents some results obtained with the simulator; and Section 5 presents the main conclusions and the future works of this research.

2. ROBOT MODEL AND POSITION CONTROL

2.1 Kinematic Model of the Robot

A differential wheeled robot is a mobile robot whose movement is based on two separately driven wheels placed on each side of its body. The two drive velocities are always two parallel vectors and perpendicular to the wheels axis. Furthermore, the wheels are assumed to roll without slipping. These conditions impose some restrictions known as non-holonomic constraints (de Wit and Sordalen (1992)). The robot can change its direction by varying the relative rotation between the wheels, so it does not need an additional steering movement to turn. The instant linear velocity (ν) is average of the linear velocities of the two wheels (ν_i, ν_d). While the angular velocity ω respect to the *ICC (Instantaneous Center of Curvature)* must be the same for both wheels. The kinematic model of the robot can be obtained in cartesian coordinates (Chwa et al. (2006); Siegwart et al. (2011)), where θ is the heading direction angle of the robot, it is given by Equations 1.

$$\begin{cases} \dot{x}_c = \nu cos(\theta) \\ \dot{y}_c = \nu sin(\theta) \\ \quad \dot{\theta} = \omega \end{cases} \tag{1}$$

2.2 Position Control or “Point Stabilization”

This problem is titled position control or point stabilization of a differential wheeled mobile robot. Its objective is to calculate the velocities of the robot (ν, ω) to drive the robot from the current position $C(x_c, y_c)$ and orientation (θ) to the target point $P(x_p, y_p)$. Figure 1 shows the variables involved.

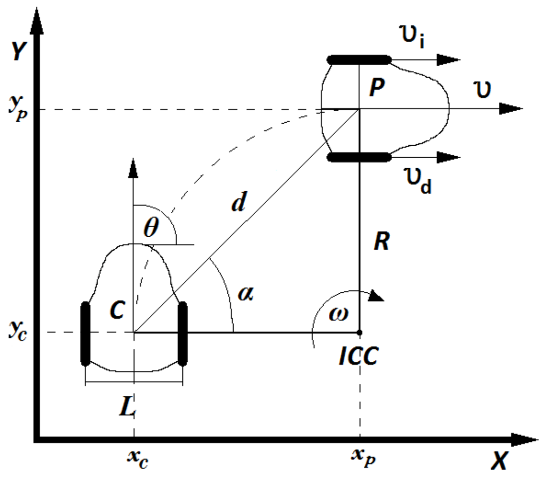


Fig. 1. Position control problem

This problem has been widely studied mainly due to the designing of the control law, under nonholonomic constraints, introducing challenging nonlinear control problems from the academic point of view, as the authors say in Kühne et al. (2005). In order to achieve the control objective, the distance (d) and the angle (α) between the points C and P are obtained using Pythagoras and the arctangent respectively like in Fabregas et al. (2014, 2016).

Figure 2 shows the control blocks diagram of this problem. The robot tries to minimize the orientation error, $\mathrm{e}_\theta = \alpha - \theta$, and at the same time, reducing the distance to the target point ($d = 0$). The values of d and α are calculated in the block *Compute*; by using the target point (P) as reference and the current position of the robot (C). These two values and the orientation θ are used by the *Control Law* block to obtain the control signals (ν and ω).

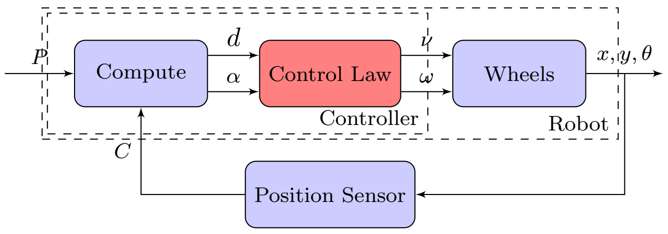


Fig. 2. Diagram of the position control problem

In the bibliography you can find different solutions for this problem. For example: Kanayama et al. (1990); Siegwart et al. (2011); Malu and Majumdar (2014) and Villela et al. (2004). The last one is represented by Equations 2 and 3.

$$\nu(t) = \begin{cases} \nu_{max} & if\, |d| > K_r \\ d\left(\dfrac{\nu_{max}}{K_r}\right) & if\, |d| \le K_r \end{cases} \tag{2}$$

$$\omega(t) = \omega_{max} sin\left(\mathrm{e}_\theta(t)\right) \tag{3}$$

The linear velocity $\nu(t)$, is obtained depending of the distance to the target point. ν_{max} is the maximum linear velocity and K_r is the radius of a docking area (around the target point). When the robot is far from the target point, the ν is saturated to ν_{max}. This velocity is decreased when the robot enter into the docking area. At the same time

ω is obtained as a function of the orientation error to the target point (e_θ).

This control law grants the range of the angular velocity between ($\omega_{max} \leq \omega \leq -\omega_{max}$) due to the sine of the error. But it has an unwanted behavior for values of $e_\theta \leq \frac{-\pi}{2}$ and $e_\theta \geq \frac{\pi}{2}$ (red color in Figure 3), where the relation between the ω and the error angle is inverse (ω decrease when the error increase). The result of this behavior is when the target point is in the back of the robot, ω can be very small like if the robot was almost oriented to the target. To reduce this behavior in this paper it is proposed to add Integral Action to this control law to improve the speed of change of the angular velocity in these situations.

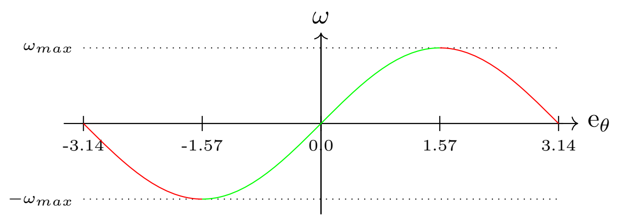


Fig. 3. Value of ω vs. e_θ

2.3 Proposed Control Law

Figure 4 shows the results of the previous control law for different initial conditions. The initial position of the robot is (0;0). The target point is (1;1). Which means that the angle between the initial position and the target point is $\alpha = 45°$. The color lines show the trajectories of the robot from the initial position to the target point for different initial orientation angles ($\theta = 0°$, $45°$, $90°$,..., $-45°$).

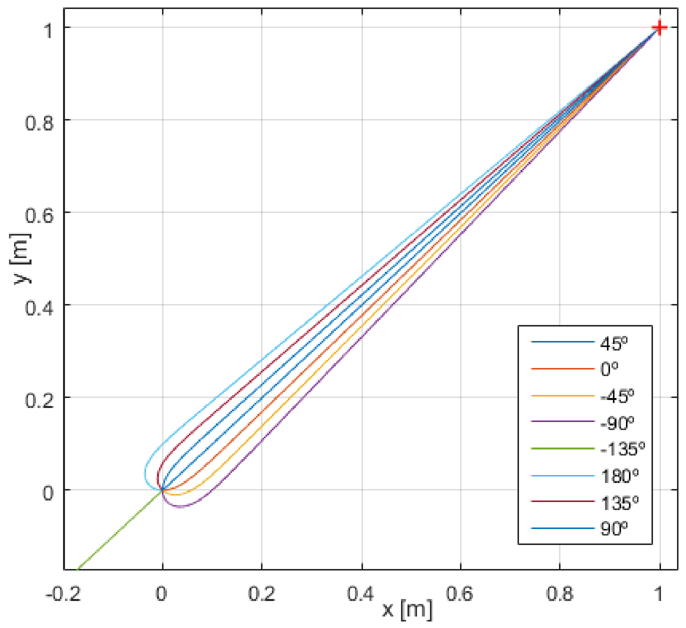


Fig. 4. Result of the position control for different initial orientations.

As can be seen when the angle error is $e_\theta = \pm 180°$ (green line), the $sin(e_\theta) = 0$. Which results in $\omega = 0$ and the robot starts moving away from the target point. This is because for the control law the target point is wrongly aligned in front of the robot, when in fact, the target point is behind the robot. To avoid this issue an Integral Action is added to the control law, as it is shown in Equation 4. Besides, the term ω_{max} is substituted by a K_p term to control the Proportional Action in relation to the Integral Action. The range of this constant is ($K_p < \omega_{max}$).

$$\omega = K_p sin\left(e_\theta(t)\right) + K_i \int e_\theta(t)dt \tag{4}$$

Figure 5 shows the implementation of the new control law with similar conditions of the previous experiment for the special case of initial orientation ($\theta = -135°$ and $e_\theta = 180°$). The lines represent the trajectories of the robot to reach the target point for different values of ($K_i = 0.00, 0.005, 0.0015, ..., 0.140$) and $K_p = 1.4$. These values have been empirically determined by trial and error.

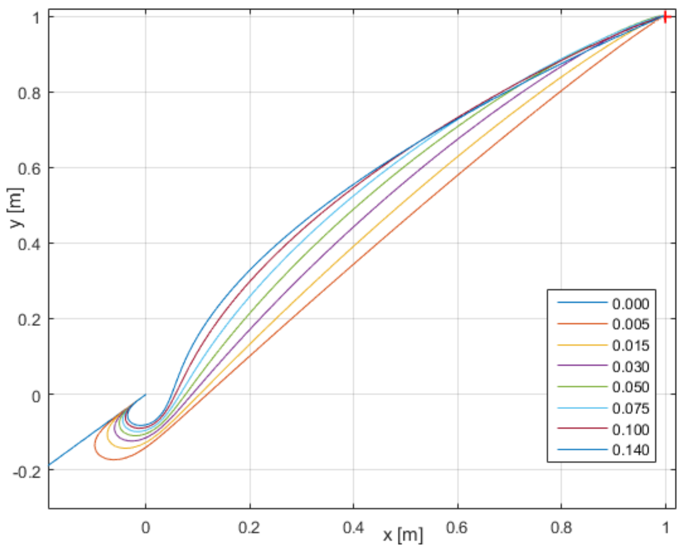


Fig. 5. Special case of $\theta = -135°$ for different values of K_i.

As can be seen for $K_i = 0.0$ the behavior is the same of the previous control law because the Integral Action is not actuating. While for some of the rest of the values, the behavior of the trajectory is improved. These parameters can be adjusted taking into account a trade-off between distance/energy consumption of the trajectory.

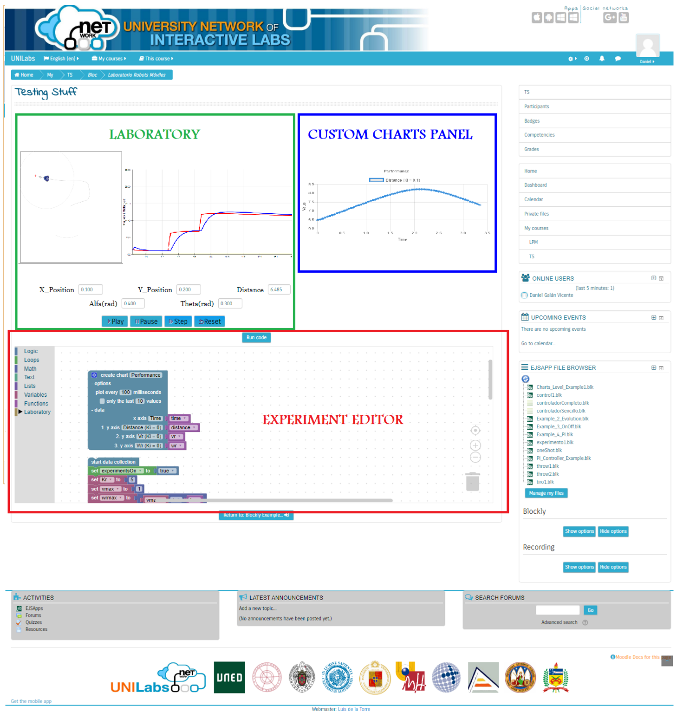
3. MOBILE ROBOTS CONTROL SIMULATOR

This section firstly explains the general architecture of the experimentation environment. The implementation of the Mobile Robots Control Simulator is then discussed. Finally, the experimentation possibilities for this laboratory are highlighted.

3.1 Experimentation environment general architecture

The experimentation environment, already used in previous works (Galán et al. (2017)), consists of three elements, which are clearly visible in Figure 6: (1) the Experiment Editor (marked with a red rectangle at the bottom), (2) the Custom Charts Panel (blue box at the top right) and (3) the Online Laboratory (green box at the top left).

Experiment Editor The Experiment Editor is the component in which the programming of the automated experiments is done. It is formed by a script area and a



X_Position 0.100 Y_Position 0.200 Distance 6.485
Alfa(rad) 0.400 Theta(rad) 0.300

Fig. 6. Visual experimentation environment with a remote laboratory of a Furuta Pendulum to perform Control experiments

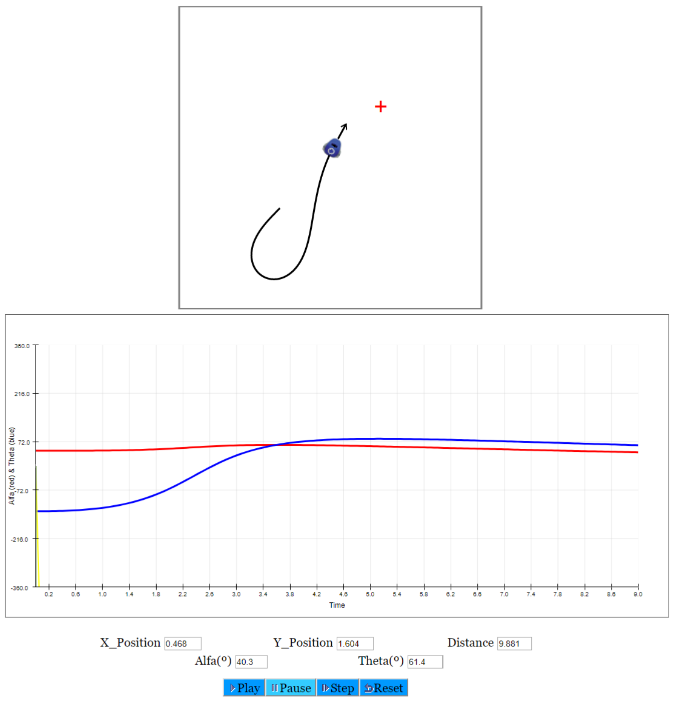
menu, located on the left, with the different categories in which the programming blocks are placed. These blocks are used to create the script experiments as jigsaw puzzles. In this work, Blockly (Fraser et al. (2013); Trower and Gray (2015)) has been extended with extra blocks to interact with the laboratories. Some of the main features that students can use to perform experiments for the particular case of the Mobile Robots Control lab are explained in Subsection 3.3.

Custom Charts Panel The environment developed in this work allows the creation and customization of charts. Students first decide the data they want to visualize and then, they can define the title, the sampling period and the maximum number of points they want to plot.

Online Laboratory Laboratories are JavaScript applications using Easy Java/JavaScript Simulations (EjsS) Esquembre and Sanchez (2012). These laboratories are automatically prepared to be used in the environment with no other adaptation or implementation. The type of the laboratory, whether virtual or remote, is entirely transparent for the experiment environment and its use is not influenced by the nature of the laboratory at all.

3.2 Mobile Robots Control virtual lab implementation

The Mobile Robots Control laboratory is a virtual lab created with EjsS to be used in conjunction with the experimentation environment proposed above. Figure 7 illustrates its graphical user interface during the execution of an experiment. The upper side shows the graphical representation of the scenario in which the robot (the blue image) and target (the red cross) are located. The trail of the robot's path and its orientation are also plotted.



X_Position 0.468 Y_Position 1.604 Distance 9.881
Alfa(º) 40.3 Theta(º) 61.4

Fig. 7. Graphical user interface of the Mobile Robots Control laboratory

Underneath it, there is a graph that shows the evolution of "alpha" (in red) and "theta" (blue) as a function of the simulation time, to study the performance of the controllers implemented by the students. Below these representations are the most representative data of the robot status (position on the x-axis and y-axis, distance to the target, theta, and alpha). And finally, there are buttons that control the evolution of the laboratory.

Note that only using this interface, it is impossible to perform experiments because there are no interactive elements to modify the parameters or to select any controller. The purpose is to create a very open laboratory were students, using the experimentation environment, are able to perform any type of experiment.

3.3 What can students do with this simulator?

Using the different blocks that the experiment editor includes, students can create their own scripts to:

(1) *Access and modify the value of variables*: For this laboratory, students have access to the variables that determine the pose of the robot, the location of the target, the distance and angle to reach the destination and the simulation time.
(2) *Modify original functions from the lab*: Students can change the behavior of the laboratory by changing the code of some specific functions, in this case, the ones that determine the linear and angular velocity of the robot ($expCalV$ and $expCalW$ respectively). In this way, students can include their own controller implementations and test them in real time.
(3) *Include events*: Students can incorporate events into the execution of the lab, for example, to change the target position when the robot is near.

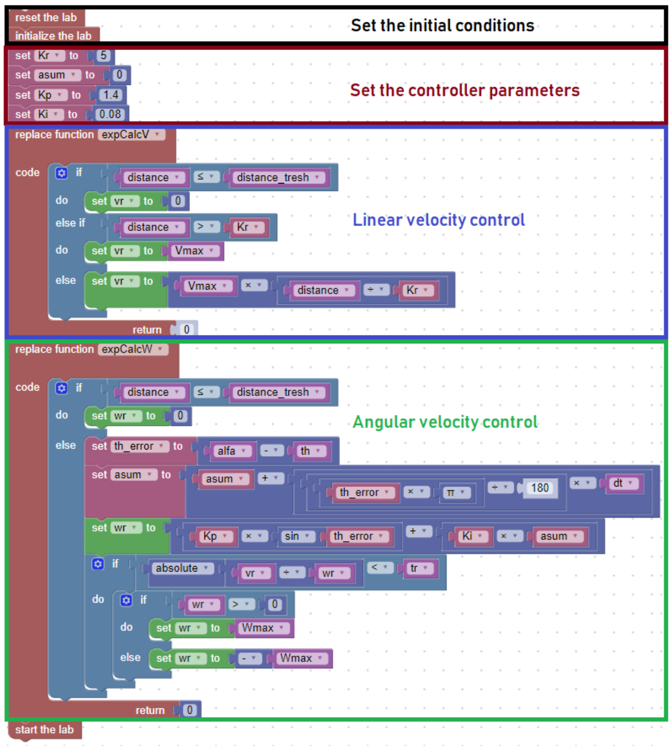
(4) *Design new graphs*: Students can visualize any value or expression in a real-time chart to perform a more exhaustive analysis of the results. For example, analyzing the time needed to reach the target using different controllers.

4. APPLYING THE PROPOSED CONTROL LAW IN THE MOBILE ROBOTS CONTROL SIMULATOR

This section describes how students can create an experiment to verify the performance of the proposed controller explained in Subsection 2.3 To do this, using the experiment editor, they have to merge different blocks to create a script in which:

(1) The initial conditions are automatically set (in case we wanted to test other initial conditions, we only have to add new statements with the desired conditions once the model has been initialized).
(2) The controller parameters are set (K_r for the linear velocity and K_p and K_i for the angular velocity).
(3) The function that controls the linear velocity (*expCalV*) is replaced with the necessary code to implement Equation 2.
(4) Equation 4 is coded using blocks to redefine the function in charge of controlling the angular velocity (*expCalW*).
(5) Finally, the experiment is executed to visualize the results.

The development of this algorithm is shown in Figure 8. For a better visualization, the code fragments that appear in the previous enumeration have been grouped together.



```
reset the lab
initialize the lab
set Kr to 5
set asum to 0
set Kp to 1.4
set Ki to 0.08
replace function expCalcV
code  if  distance ≤ distance_tresh
      do  set vr to 0
      else if  distance > Kr
      do  set vr to Vmax
      else  set vr to Vmax × ( distance ÷ Kr )
              return 0
replace function expCalcW
code  if  distance ≤ distance_tresh
      do  set wr to 0
      else  set th_error to alfa - th
            set asum to asum + ( ( ( th_error × π ) ÷ 180 ) × dt )
            set wr to ( Kp × sin th_error ) + ( Ki × asum )
            if  absolute ( vr ÷ wr ) < tr
            do  if  wr > 0
                do  set wr to Wmax
                else  set wr to - Wmax
              return 0
start the lab
```

Fig. 8. Script implementing the proposed controller detailed in Subsection 2.3

As the innovative part of this controller is given in the calculation of angular velocity, the analysis focuses on the influence of K_i on the robot's orientation to the target. To do this, the experiment was executed four times with different K_i values (0.8, 0.08, 0.008 and 0). The initial conditions were the same in each case; the robot started from the position $(-8.0; -8.0)$ with an initial orientation of $\theta = 225°$, while the target was situated at $(8.0; 8.0)$. This initial situation makes the orientation error $e_\theta = 180°$), giving rise to the problem described in Section 2. The results are shown in Figure 9.

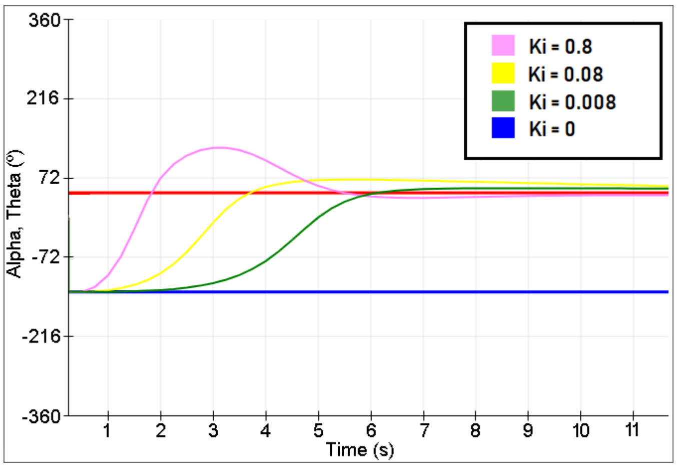


Fig. 9. Influence of K_i on the robot's orientation

It is possible to compare the evolution of *theta* over time between each of the four executions (using $K_p = 1.4$). As can be seen, there is an overshoot for the higher value of $K_i = 0.8$ (pink color line). Which means that the alignment of the robot with the destination point is exceeded and it has to be corrected. What is undoubtedly an unwanted behavior. The blue line color represents the control law that was used as basis (with $K_i = 0.0$). The best performance is obtained with $K_i = 0.008$ (green color line). Despite being the slowest response, it is the best of them because it has no overshoot. Which means that the robot reach the desired orientation in a smooth way. As was mentioned before, these values are only to show that the controller is working well. As additional step, these parameters can be adjusted using the known tuning methods taking into account that both, system and controller are nonlinear.

5. CONCLUSION

This paper presents a new on-line interactive simulator (developed in JavaScript using Easy Java/JavaScript Simulations) to control mobile robots. Most current virtual and remote control laboratories are closed applications which offer experimentation tasks purely based on interactivity. This limits the students' work with the laboratory. An experimentation environment capable of overcoming these limitations is also introduced. To this end, the environment presents some interesting features, such as:

- It is on-line and it can be accessed from Internet (the web-page where it can be found is: `http://unilabs.dia.uned.es/`).
- Users do not have to install any software to work with it. They only need a web browser (Firefox, Chrome, etc.).

- It adds the possibility of designing automated experiments with a visual and easy-to-use language, solving the limitations of interactive experimentation.
- It allows users to redefine functions and modify variables the designer might not have considered interesting at the time, opening laboratories to new experimental tasks that were not previously possible.

To verify the experimentation environment usefulness the new strategy to control mobile robots position presented also in this work has been successfully implemented.

Future works will include the improvement of the simulator based on the opinion of the students. Besides, the improvement of the proposed control law to eliminate the unwanted behavior for $e_\theta \leq \frac{-\pi}{2}$ and $e_\theta \geq \frac{\pi}{2}$.

REFERENCES

Ausín, V., Abella, V., Delgado, V., and Hortigüela, D. (2016). Aprendizaje basado en proyectos a través de las tic: Una experiencia de innovación docente desde aulas universitarias. *Formación universitaria*, 9(3), 31–38.

Broisin, J., Venant, R., and Vidal, P. (2017). Awareness and reflection in virtual and remote laboratories: the case of computer education. *International Journal of Technology Enhanced Learning*, 9(2-3), 254–276.

Browne, A.F. and Conrad, J.M. (2017). A versatile approach for teaching autonomous robot control to multi-disciplinary undergraduate and graduate students. *IEEE Access*, PP(99), 1–1.

Chwa, D., Hong, S.K., and Song, B. (2006). Robust posture stabilization of wheeled mobile robots in polar coordinates. In *The 17th International Symposium on Mathematical Theory of Networks and Systems*, volume 39, 343–348.

de Wit, C.C. and Sordalen, O.J. (1992). Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Transactions on Automatic Control*, 37(11), 1791–1797.

Egerstedt, M., Hu, X., and Stotsky, A. (2001). Control of mobile platforms using a virtual vehicle approach. *IEEE Transactions on Automatic Control*, 46(11), 1777–1782.

Esquembre, F. and Sanchez, J. (2012). Easy java simulations. In 9th *Workshop on Multimedia in Physics Teaching and Learning.*

Fabregas, E., Farias, G., Dormido-Canto, S., and Dormido, S. (2014). RFCSIM simulador de robótica móvil para control de formación con evitación de obstáculos. In *XVI Congreso Latinoamericano de Control Automático.* Cancún, México.

Fabregas, E., Farias, G., Dormido-Canto, S., Dormido, S., and Esquembre, F. (2011). Developing a remote laboratory for engineering education. *Computers & Education*, 57(2), 1686–1697.

Fabregas, E., Farias, G., Dormido-Canto, S., Guinaldo, M., Sánchez, J., and Dormido, S. (2016). Platform for teaching mobile robotics. *Journal of Intelligent & Robotic Systems*, 81(1), 131–143.

Fraser, N. et al. (2013). Blockly: A visual programming editor. *URL: https://code. google. com/p/blockly.*

Galán, D., de la Torre, L., Dormido, S., Heradio, R., and Esquembre, F. (2017). Blockly experiments for ejss laboratories. In 4th *Experimenta International Conference*, 139–140. IEEE.

Halimi, Wissam; Salzmann, C.J.H. and Gillet, D. (2018). *Enabling the Automatic Generation of User Interfaces for Remote Laboratories*, 778–793. Springer International Publishing, Cham.

Huang, Y., Yong, Y.S., Chiba, R., Arai, T., Ueyama, T., and Ota, J. (2016). Kinematic control with singularity avoidance for teaching-playback robot manipulator system. *IEEE Transactions on Automation Science and Engineering*, 13(2), 729–742.

Ionescu, C.M., Fabregas, E., Cristescu, S.M., Dormido, S., and Keyser, R.D. (2013). A remote laboratory as an innovative educational tool for practicing control engineering concepts. *IEEE Transactions on Education*, 56(4), 436–442.

Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, 384–389. IEEE.

Kühne, F., Lages, W.F., and da Silva Jr, J.M.G. (2005). Point stabilization of mobile robots with nonlinear model predictive control. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 3.

Malu, S.K. and Majumdar, J. (2014). Kinematics, localization and control of differential drive mobile robot. *Global Journal of Research In Engineering.*

Pastor, R., Sánchez, J., and Dormido, S. (2003). A xml-based framework for the development of web-based laboratories focused on control systems education. *International Journal of Engineering Education*, 19(3), 445–454.

Peralta, E., Fabregas, E., Farias, G., Vargas, H., and Dormido, S. (2016). Development of a khepera iv library for the V-REP simulator. *IFAC-PapersOnLine*, 49(6), 81–86. 11th IFAC Symposium on Advances in Control Education ACE 2016.

Potkonjak, V., Gardner, M., Callaghan, V., Mattila, P., Guetl, C., Petrović, V.M., and Jovanović, K. (2016). Virtual laboratories for education in science, technology, and engineering: A review. *Computers & Education*, 95, 309–327.

Siegwart, R., Nourbakhsh, I.R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots.* MIT press.

Trower, J. and Gray, J. (2015). Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 5–5. ACM.

Vargas, H., Sánchez, J., Duro, N., Dormido, R., Dormido-Canto, S., Farias, G., Dormido, S., Esquembre, F., Salzmann, C., and Gillet, D. (2008). A systematic two-layer approach to develop web-based experimentation environments for control engineering education. *Intelligent Automation & Soft Computing*, 14(4), 505–524.

Villela, V.J.G., Parkin, R., Parra, M.L., González, J.M.D., Liho, M.J.G., and Way, H. (2004). A wheeled mobile robot with obstacle avoidance capability. *Tecnología Y Desarrollo*, 1(5), 159–166.

Wang, H., Guo, D., Liang, X., Chen, W., Hu, G., and Leang, K.K. (2017). Adaptive vision-based leader follower formation control of mobile robots. *IEEE Transactions on Industrial Electronics*, 64(4), 2893–2902.