m-PaRoLa: a Mobile Virtual Laboratory for Studying the Kinematics of Five-bar and 3RRR Planar Parallel Robots ⋆

Adrián Peidró, Carlos Tendero, José María Marín
Arturo Gil, Luis Payá, Óscar Reinoso

Miguel Hernández University, 03202 Elche, Spain
e-mail: apeidro@umh.es, carlos.tendero@alu.umh.es, jmarin@umh.es, arturo.gil@umh.es, lpaya@umh.es, o.reinoso@umh.es

Abstract: This paper presents m-PaRoLa, an educational virtual laboratory that consists of Javascript simulations for analyzing parallel robots. These simulations can be run indistinctly on web browsers of desktop computers or mobile devices (smartphones, tablets), which turns m-PaRoLa into a mobile virtual laboratory ready to be integrated in m-learning methodologies for teaching robotics and mechanisms. The presented simulations, which are highly intuitive and visual, allow the user to analyze diverse kinematic problems of parallel robots.

Keywords: Virtual Laboratories, m-Learning, Parallel Robots, Javascript, Java

1. INTRODUCTION

Mobile learning (m-learning) is a teaching and learning methodology that makes use of mobile devices with wireless Internet connection, mainly smartphones and tablets. Some advantages of this methodology are (Asabere, 2013; Alioon and Delialioğlu, 2017; Bhullar, 2014):

- Possibility of learning anywhere and anytime
- Increased communication, collaboration, and interaction between students and with instructors
- Most mobile devices are more affordable and easier to use than laptops and desktop computers, and most students own and are proficient at using smartphones
- Increased motivation of students

However, learning with mobile devices also has some disadvantages (Bhullar, 2014) related to usability issues (too small screens and limited or uncomfortable input methods) or inappropriate/offensive use of mobile devices (especially among younger students), among others.

An educational virtual laboratory typically is a web-based environment that allows the user to experiment with some aspects of a simulated system, with the purpose of facilitating the comprehension of these aspects. Some of the advantages of virtual over real laboratories are reduced costs, flexibility, and safety (Potkonjak et al., 2016). Traditionally, students have accessed virtual laboratories through web browsers or applications running in desktop computers. However, there have also been some relatively recent efforts to develop educational mobile virtual laboratories (mv-labs) that exploit the aforementioned advantages of m-learning. Glavinic et al. (2007) presented an mv-lab for learning Digital Design, and discussed some issues related to usability and human-computer interaction in graphical user interfaces to be used in the small screens of mobile devices (Glavinic et al., 2009). Bottentuit Jr. and Coutinho (2007) proposed another mv-lab for learning Organic Chemistry. Bhosale and Livingston (2014) presented an mv-lab for learning about Network Security. Jardim et al. (2014) proposed an architecture of ubiquitous learning (u-learning, which can be considered as an evolution of m-learning) that allows the users to study anywhere, anytime, and from any device, and they applied it to teach about Computer Networks. Oluwole et al. (2015) developed an mv-lab for teaching concepts on Chemistry, Physics, and Computer Science. Finally, Merabet et al. (2015) presented an mv-lab for teaching about biometrics identification, electronics, and programming languages.

Due to its strongly multidisciplinary nature, robotics is one of the topics on which many virtual and remote laboratories have been developed (Potkonjak et al., 2016; Torres et al., 2006; Othayoth et al., 2017). However, educational *mobile* virtual laboratories on robotics are less common. Considering the increasing trends and advantages of m-learning, and considering robotics as one of the fields in which it will be fundamental to instruct future engineers, this paper presents the first steps of the development of m-PaRoLa (mobile Parallel Robotics Laboratory, `http://arvc.umh.es/mparola`), an educational mv-lab for learning about the kinematics of parallel robots and mechanisms. At present, m-PaRoLa consists of two Javascript simulators that allow the user to experiment and learn about different kinematic concepts of two well-known planar parallel robots: the five-bar and 3RRR robots.

This paper is organized as follows. Section 2 introduces parallel robots and virtual tools for simulating and studying them. Sections 3 and 4 present the five-bar and 3RRR parallel robots, respectively, as well as the web-based simulators we have developed for studying these robots. Finally, Section 5 presents the conclusions and future work.

⋆ Work supported by the Spanish Ministries of Education (grant No. FPU13/00413) and Economy (project No. DPI 2016-78361-R).

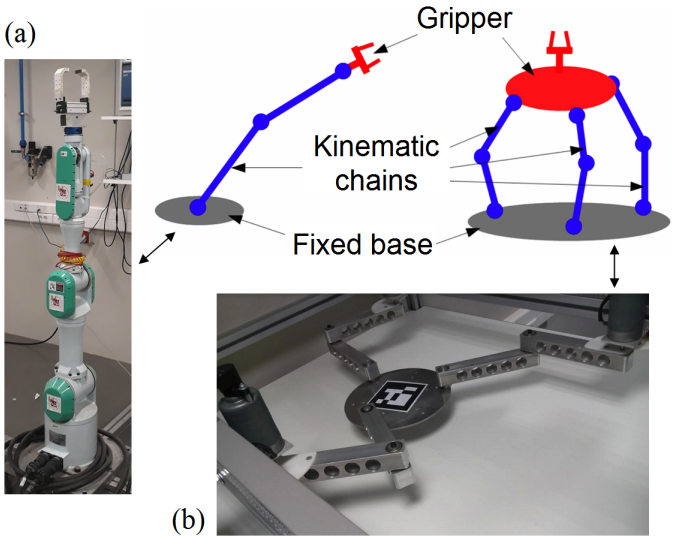

Fig. 1. Serial (a) and parallel (b) manipulators.

2. SIMULATING PARALLEL ROBOTS

Parallel robots (Fig. 1b) are manipulators in which the gripper is controlled and connected to the fixed base through two or more kinematic chains in parallel. This is an alternative to serial arm-like manipulators commonly found in industry, in which the gripper is controlled by a single open kinematic chain (Fig. 1a). Parallel robots have some advantages over serial ones, such as higher stiffness, higher payload/weight ratio, and greater dynamic performance (they can attain very high accelerations). However, they also have some drawbacks: they can reach singular configurations where it is not possible to completely control the motion of the gripper, their workspace is more limited, and their forward kinematic problem is more complex than in serial robots.

PaRoLa (Parallel Robotics Laboratory, `http://arvc.umh.es/parola`) (Peidró et al., 2016; Peidró et al., 2015) is a (non-mobile) educational virtual laboratory which allows students to experiment with the simulations of several parallel robots and mechanisms, with the purpose of understanding their kinematics, workspace, singularities, and dynamics. PaRoLa is composed of a collection of Java applets developed using Easy Java Simulations authoring tool (Esquembre, 2004). Like most virtual laboratories developed in the past, the original objective of PaRoLa was to allow students to directly run from web browsers the educational Java applets embedded in websites, without having to download and run the `.jar` files on their desktop computers. However, most popular web browsers have been gradually abandoning their support of embedded Java applets during the last years, so that currently it is not possible to run these applets directly embedded in websites. Moreover, `.jar` simulators cannot be downloaded and run on mobile devices.

Since its release 5.0, Easy Java Simulations (`http://www.um.es/fem/EjsWiki`) allows the user to develop simulations both in Java or Javascript languages. The advantage of simulations written in Javascript is that these can be run directly on websites, without having to download them. Furthermore, Javascript simulations can be indistinctly run on web browsers of both desktop computers and mobile devices, favoring their portability and their use in m-learning strategies.

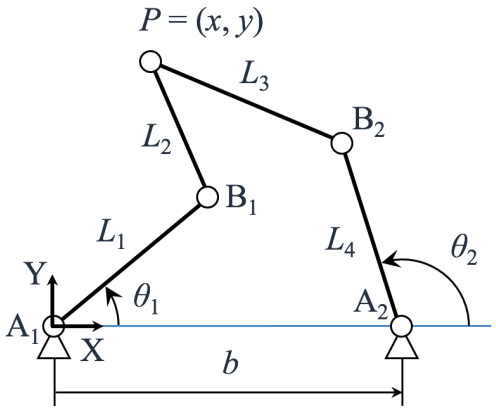

Fig. 2. Five-bar planar parallel robot.

Taking this into account, this paper presents m-PaRoLa: a Javascript-based mv-lab evolved from PaRoLa, developed by means of Easy Java/Javascript Simulations. m-PaRoLa conserves the functionalities of PaRoLa, besides allowing for a more intuitive, graphical and "elastic" way of defining the geometry of the simulated robots (see Section 3.2). The purpose of m-PaRoLa is to offer the user the possibility of experimenting with simulated parallel robots and mechanisms (and also remote robots, in the future) in order to study and learn about diverse aspects of these robots from any device (including mobile devices). In next sections, we will present and describe the simulators currently included in m-PaRoLa, which allow the user to experiment with simulated five-bar and 3RRR parallel robots. These simulators are available at `http://arvc.umh.es/mparola`, and have been successfully tested on different smartphones and tablets with different operating systems (iOS, Android) and different browsers (Google Chrome, Mozilla Firefox). Also, we will describe how we solved some issues encountered when trying to implement several functionalities in the simulators, due to the small size of the screens of mobile devices and their limited input methods.

3. SIMULATOR OF THE FIVE-BAR ROBOT

This section describes the five-bar planar parallel robot, its associated kinematic problems, and the Javascript simulator developed for studying these problems.

3.1 The Five-bar Planar Parallel Robot

The five-bar parallel robot, shown in Fig. 2, is a closed kinematic chain composed of five bars (A_1B_1, B_1P, PB_2, B_2A_2, A_1A_2) interconnected through five revolute joints (A_1, B_1, P, B_2, A_2). Bar A_1A_2 is the fixed base of this manipulator. Joint P is the gripper, which is connected to the fixed base through kinematic chains A_1B_1P and A_2B_2P. This robot has two degrees of freedom: by controlling angles θ_1 and θ_2 (e.g., by means of electric motors connected to joints A_1 and A_2), it is possible to control the position (x, y) of joint P in plane XY.

The forward kinematic problem of this robot consists in calculating the position (x, y) of the gripper for given

known values of θ_1 and θ_2. This problem consists in intersecting two circles (one circle is centered at B_1 and has radius L_2, the other is centered at B_2 and has radius L_3) and, therefore, it has two different solutions: for given angles (θ_1, θ_2), joint P can be at the position shown in Fig. 2 or at its symmetric position with respect to line B_1B_2.

The inverse kinematics of this robot consists in determining the necessary angles (θ_1, θ_2) to place gripper P at a desired known position (x, y). For a given position (x, y), angle θ_1 has two possible solutions: one solution is shown in Fig. 2, the other solution places joint B_1 at the symmetric position of the one shown in Fig. 2 (symmetric with respect to line A_1P). These two solutions are the intersections of two circles: one is centered at A_1 and has radius L_1, and the other is centered at P and has radius L_2. Analogously, there are two possible angles θ_2 that place the gripper at the desired position (x, y): one solution is shown in Fig. 2, the other solution places joint B_2 at the symmetric position of the one shown in Fig. 2 (symmetric with respect to line A_2P). There are two possible solutions for angle θ_1, and these are independent of the two possible solutions of angle θ_2. Thus, the inverse kinematics of this robot has four different solutions (four possible combinations of the two solutions of θ_1 with the two solutions of θ_2).

The workspace of this robot is the set of points that gripper P can reach. When subject to kinematic chain A_1B_1P, P can reach all points in an annular region $\mathcal{A}_1$ enclosed between two concentric circles centered at joint A_1: one circle has radius L_1+L_2 (chain A_1B_1P completely stretched) and the other has radius $|L_1-L_2|$ (chain A_1B_1P completely folded). Analogously, when subject to chain A_2B_2P, P can reach all points in an annular region $\mathcal{A}_2$ enclosed between two concentric circles centered at A_2: one circle has radius $L_3 + L_4$ (chain A_2B_2P completely stretched) and the other has radius $|L_3 - L_4|$ (chain A_2B_2P folded). Thus, the workspace of this robot is the intersection of annuli $\mathcal{A}_1$ and $\mathcal{A}_2$. The boundaries of this workspace are shown in blue line in panel p1 of Fig. 3.

The singularities of this robot are the configurations for which bars B_1P and B_2P are parallel, i.e., when these bars are either completely extended or folded. For these configurations, it is not possible to control the motion of the gripper along the direction perpendicular to these bars. Singularities occur along singular curves that divide the workspace into different singularity-free regions.

Finally, the design of the robot is defined by five geometric parameters: the position b of A_2 along axis X, and $\{L_1, L_2, L_3, L_4\}$, which are the respective lengths of bars $\{A_1B_1,\ B_1P,\ B_2P,\ A_2B_2\}$. The shapes of the workspace and singular curves depend on all these five parameters.

3.2 Javascript Simulator of the Five-bar Robot

The simulator of the five-bar robot is available at `http://arvc.umh.es/mparola/five-bar/five-bar.html`. Figure 3 shows a screenshot of this simulator running on Google Chrome browser in Android.

As Fig. 3 shows, the presented simulator has four panels:

- Panel p1 is a schematic representation of the five-bar robot in plane XY.

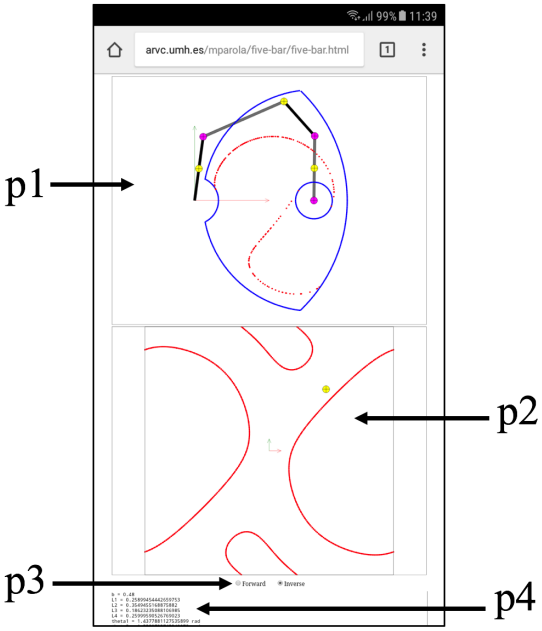

Fig. 3. Javascript simulator of the Five-bar robot.

- Panel p2 represents plane (θ_1, θ_2) for ranges: $-\pi \leq \theta_1 \leq \pi$, $-\pi \leq \theta_2 \leq \pi$.
- Panel p3 is just a selector that allows the user to choose between simulating the forward or inverse kinematic problems.
- Panel p4 is a read-only text that indicates at all times the current values of all relevant parameters: b, L_1, L_2, L_3, L_4, θ_1, θ_2, x, and y.

In non-mobile virtual laboratories designed to be used with desktop computers (like PaRoLa), the large size of the screens of these computers allow the graphical user interfaces of these virtual laboratories to have several windows and detailed menus to configure and perform the simulated experiments. However, mobile virtual laboratories are much more limited in this aspect since they must run on the relatively small screens of mobile devices (Glavinic et al., 2009), and the graphical interfaces of these labs often must be greatly optimized in order to offer the user the required functionalities in an intuitive manner, respecting the small screens and limited input methods of mobile devices, which can be challenging. This impedes using too many menus and windows in the graphical interfaces of mv-labs, and encourages using minimalist designs that favor graphics over text. Next, we will describe the different capabilities of the developed simulator, explaining how we have optimized its graphical user interface to implement several functionalities in the little space available.

Simulating the forward kinematics. First, the user should choose to simulate the forward kinematics in panel p3. Then, one must provide angles θ_1 and θ_2 so that the simulator can solve the forward kinematics and compute the position (x, y) of joint P, showing the solution graphically in panel p1 and textually in panel p4. Angles θ_1 and θ_2 can be provided by directly dragging the yellow circles placed at the center of bars A_1B_1 and A_2B_2, respectively, in panel p1. Alternatively, the user can also provide the desired values of θ_1 and θ_2 by dragging the yellow circle in plane (θ_1, θ_2) in panel p2 (this allows one to vary both

angles simultaneously). As explained in subsection 3.1, the forward kinematics of this robot has two solutions. To switch between these two solutions, the user must click on the yellow circle placed on the gripper (joint P).

Simulating the inverse kinematics. Let us assume now that the user chooses to simulate the inverse kinematics in panel p3. In that case, one must provide the position (x, y) of the gripper so that the simulator can compute angles θ_1 and θ_2, showing the solution graphically in panels p1 and p2, and textually in panel p4. In order to provide the position of the gripper, the user must drag the yellow circle placed on joint P (whose function was to switch between the two solutions of the forward kinematic problem, when simulating that problem).

Recall from subsection 3.1 that the inverse kinematic problem of each chain A_iB_iP has two different solutions. To switch between the two solutions of each chain, one must click the yellow circle placed at the center of bars A_1B_1 or A_2B_2 (these circles are used for dragging and specifying the desired values of angles θ_1 and θ_2 when simulating the forward kinematics).

Visualizing the workspace and singularities. Panel p1 represents at all times the boundaries of the workspace (in blue). Also, singular curves are represented in red in plane (θ_1, θ_2) (panel p2), and also in panel p1 if the inverse kinematics is being simulated (the red singular curves shown in panel p1 depend on the chosen solution of the inverse kinematics). While simulating the inverse kinematics, the user can check how the posture of the robot varies as the gripper is dragged along the workspace, checking how kinematic chains A_iB_iP stretch or fold when approaching the boundaries of the workspace. The user can also check how bars B_1P and B_2P remain aligned when placing the gripper on a red singular curve in panel p1. If the user moves the gripper outside the workspace, the robot seems to "break" since the inverse kinematics has no real solutions there (these are unreachable positions). Something similar occurs when simulating the forward kinematics: if the user crosses the singular curves in panel p2, the robot will seem to "break" since there are not real solutions to the forward kinematics beyond these curves.

The shapes of the workspace and singular curves depend on the design of the robot. Thus, if the user modifies the geometric parameters $\{b,\ L_1,\ L_2,\ L_3,\ L_4\}$, these shapes will change. In order to modify the geometry of the robot, the user must drag *and drop* the magenta circles placed on joints A_2, B_1, and B_2. Dragging joint A_2 varies both b and L_4. Similarly, dragging joint B_1 varies both L_1 and L_2. Finally, dragging joint B_2 varies both L_3 and L_4. Thus, varying these joints deforms the bars of the robot as if it was made of rubber, and the workspace and singular curves deform as a consequence. This functionality is very useful for studying how the shapes of the workspace and singular curves depend on the design of the robot. For example, it can be checked using the simulator that if b is much smaller than L_i, and if all L_i are very similar, then we get a wide circular workspace with large singularity-free regions.

This graphical method for varying the design of the robot (i.e., dragging joints A_2, B_1, and B_2) has a drawback: it always modifies two geometric parameters simultaneously (unlike the simulators of virtual lab PaRoLa, which allow the user to modify only the desired parameter). However, the advantage of this graphical method is that it does not require additional menus (as in PaRoLa) that would complicate the graphical user interface, which must remain as simple and minimalist as possible, as argued above.

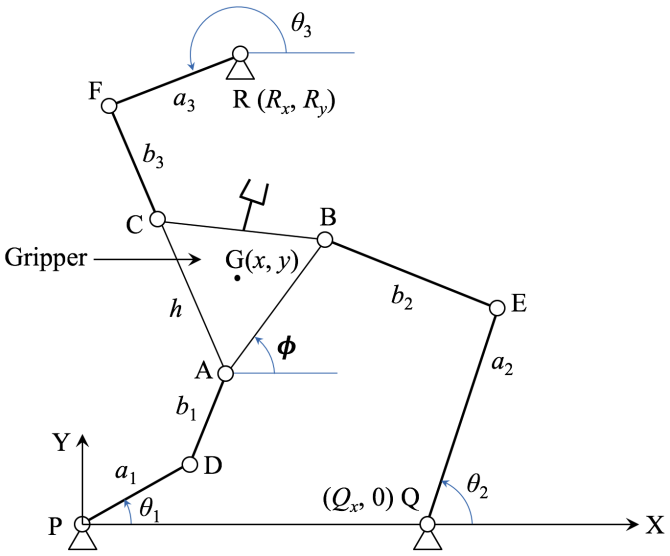

Fig. 4. 3RRR planar parallel robot.

4. SIMULATOR OF THE 3RRR ROBOT

This section describes the 3RRR planar parallel robot, its associated kinematic problems, and the Javascript simulator developed for studying these problems.

4.1 The 3RRR Planar Parallel Robot

The 3RRR planar parallel robot, shown in Fig. 4, is a manipulator composed of an equilateral triangular platform ABC connected to a fixed base PQR through three kinematic chains connected in parallel: PDA, QEB, and RFC. This robot has three degrees of freedom: controlling angles $\{\theta_1, \theta_2, \theta_3\}$ (e.g., by means of electric motors connected to joints P, Q, and R), it is possible to control the orientation ϕ of the gripper and the position (x, y) of its center G. Fig. 1b shows a photography of a real 3RRR robot.

The geometric parameters of the 3RRR robot are (see Fig. 4): Q_x (position of joint Q along axis X), (R_x, R_y) (position of joint R in plane XY), h (side of the equilateral gripper), and $\{a_1, b_1, a_2, b_2, a_3, b_3\}$ (which are the respective lengths of bars {PD, DA, QE, EB, RF, FC}).

The forward kinematics of this robot consists in calculating the position (x, y) and orientation ϕ of the gripper for given values of $(\theta_1, \theta_2, \theta_3)$. This problem has six solutions in the complex domain (Gosselin and Sefrioui, 1991) (i.e., when x, y, and/or ϕ are complex numbers), and depending on the concrete values of the geometric parameters and of angles $(\theta_1, \theta_2, \theta_3)$, two, four, or six of these solutions will be real (i.e., they will have zero imaginary part).

The inverse kinematics of this robot consists in calculating angles $(\theta_1, \theta_2, \theta_3)$ necessary to place the gripper at a desired position (x, y) and with a desired orientation

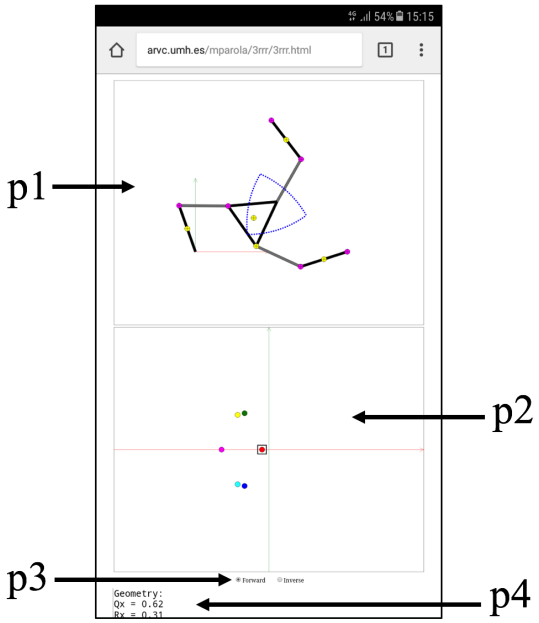

Fig. 5. Javascript simulator of the 3RRR robot.

ϕ. If (x, y, ϕ) are known, then so are the positions of joints {A, B, C} and, therefore, the inverse kinematic problem of chains {PDA, QEB, RFC} can be solved as the inverse kinematics of chains $\{A_1B_1P, A_2B_2P\}$ of the five-bar robot. Thus, there are two possible solutions for each angle θ_i $(i = 1, 2, 3)$ of the 3RRR robot, and the inverse kinematics of this robot has eight different solutions (corresponding to possible combinations of the two solutions for each angle θ_i).

The workspace of the 3RRR robot is the set of positions and orientations that its gripper can attain. Therefore, it is a set in three-dimensional space (x, y, ϕ). However, it is easier to visualize the constant-orientation workspace, i.e., the set of points (x, y) that can be attained by the center G of the gripper when its orientation ϕ remains constant. The constant-orientation workspace of the 3RRR robot is the intersection of three annular regions $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$. Annular region $\mathcal{A}_1$ is enclosed between two concentric circles centered at the point obtained when translating joint P by vector $\overrightarrow{AG}$: one circle has radius $a_1 + b_1$ (chain PDA completely extended) and the other has radius $|a_1 - b_1|$ (chain PDA completely folded). Analogously, $\mathcal{A}_2$ is enclosed between two circles centered at $\overrightarrow{PQ} + \overrightarrow{BG}$, with radii $a_2 + b_2$ and $|a_2 - b_2|$. Finally, $\mathcal{A}_3$ is enclosed between two circles centered at $\overrightarrow{PR}+\overrightarrow{CG}$, with radii a_3+b_3 and $|a_3-b_3|$. When varying any geometric parameter of the robot or the orientation ϕ of the gripper, the shape of the constant-orientation workspace will change. The boundaries of this workspace are shown in blue line in panel p1 of Fig. 5.

4.2 Javascript Simulator of the 3RRR Robot

The simulator of the 3RRR robot is available at `http://arvc.umh.es/mparola/3rrr/3rrr.html`. Figure 5 shows a screenshot of the simulator running on Google Chrome browser in Android, which exhibits four panels:

- Panel p1 is a schematic representation of the 3RRR robot in plane XY.
- Panel p2 only appears if the user chooses to simulate the forward kinematics in panel p3. Panel p2 shows the complex plane of angle ϕ, i.e.: the horizontal axis represents $\mathrm{Re}(\phi)$, whereas the vertical axis represents $\mathrm{Im}(\phi)$. More details of this panel will be given later.
- Panel p3 allows the user to choose between simulating the forward or inverse kinematic problems.
- Panel p4 is a read-only text that indicates the current values of all relevant parameters: geometric parameters (Q_x, R_x, R_y, a_i, b_i, h), controlled angles (θ_i), and position/orientation of the gripper (x, y, ϕ).

As in the simulator of the five-bar robot, these panels contain several interaction elements, optimized to offer the user diverse functionalities in the little space available (and with the limited input methods) in the screens of mobile devices. These functionalities are explained next:

Simulating the forward kinematics. After choosing to simulate the forward kinematic problem in panel p3, the user must provide angles $(\theta_1, \theta_2, \theta_3)$. These angles must be provided by dragging the yellow circles placed at the center of bars PD, QE, or RF. When doing this, the simulator solves the forward kinematic problem, obtaining (x, y, ϕ) and representing the corresponding posture in panel p1. According to subsection 4.1, the forward kinematic problem of the 3RRR robot has six solutions in the complex domain. These six solutions are represented in panel p2, which represents plane $(\mathrm{Re}(\phi), \mathrm{Im}(\phi))$. Each solution is represented as a circle in a different color. As the user varies angles $(\theta_1, \theta_2, \theta_3)$, the solutions move along this complex plane, with the possibility that the number of real solutions changes. To see the posture of the robot in panel p1 corresponding to each solution, the user must click on the desired solution in panel p2 (the chosen solution appears enclosed by a black square in panel p2). If the user chooses a non-real solution (i.e., with $\mathrm{Im}(\phi) \neq 0$), the corresponding posture of the robot shown in panel p1 will be incorrect (the robot will appear as if it was "broken", which means that complex solutions are not physically possible). When two real solutions coincide on the real axis to become complex conjugates, these solutions are singular (and it can be checked in the simulator that, for these singular solutions, all three lines {DA, EB, FC} simultaneously intersect at a single point).

Simulating the inverse kinematics. In order to simulate the inverse kinematics, the user must provide the position (x, y) and the orientation ϕ of the gripper. The position is provided by dragging the central yellow circle of the triangular gripper. This translates the gripper along plane XY without changing its orientation. To vary the orientation ϕ, the user must drag the yellow circle placed on joint B of the gripper: this will rotate the gripper about its center. To change the displayed solution of the inverse kinematics of each kinematic chain, the user must click on the yellow circles placed at the centers of bars {PD, QE, RF} (these circles were used for varying angles $\{\theta_1, \theta_2, \theta_3\}$ when simulating the forward kinematics).

Visualizing the constant-orientation workspace. At all times, panel p1 represents in blue the boundaries of the constant-orientation workspace, for the current value of ϕ. If ϕ is varied in any way (when simulating the kinematics),

or if any of the geometric parameters of the robot varies, the shape of the constant-orientation workspace is recomputed. While the user simulates the inverse kinematics, the gripper can be dragged toward the boundaries of the workspace to visualize how different chains stretch or fold. When moving the center of the gripper beyond these boundaries, the robot seems to "break" because there are not real solutions to the inverse kinematics outside these boundaries (these are unreachable points).

As stated above, the shape of the constant-orientation workspace will vary if any of the geometric parameters of the robot changes. The geometric parameters of the 3RRR robot can be varied in similar ways to the five-bar robot: by dragging the magenta circles placed on different joints of the robot. This stretches and compresses the different bars of the robot as if it was made of rubber. For example, moving joint Q simultaneously varies parameters $\{Q_x, a_2\}$. Moving R varies $\{R_x, R_y, a_3\}$. Moving D, E, or F varies $\{a_1, b_1\}$, $\{a_2, b_2\}$, or $\{a_3, b_3\}$, respectively. Finally, moving joint A varies size h of the gripper, as well as $\{b_1, b_2, b_3\}$.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a mobile virtual laboratory for studying some kinematic concepts of two well-known parallel robots: the five-bar and 3RRR robots. Unlike in desktop-based non-mobile existing virtual laboratories, a mobile virtual laboratory must have an intuitive and visual interface that offers the user all necessary functionalities, considering the constraints of small size of the screens and the limited input methods of mobile devices. This forces us to greatly optimize the graphical user interfaces and make use of interaction elements with multiple functionalities, such that each interaction element exhibits one functionality or another depending on the kinematic analysis that the user wants to perform.

In the future, we will continue adding more robots (including real remote robots, like our 3RRR robot shown in Fig. 1b) to the presented mobile virtual laboratory m-PaRoLa, departing from the already implemented robots available in the non-mobile lab PaRoLa. With the purpose of recycling most of the Java code written for PaRoLa in the past, we will implement an Internet-based client-server model similar to the one proposed by Saenz et al. (2016). According to this model, the client side will be a Javascript simulation that will run on the web browser of the student (which can be a mobile device or a desktop computer), and this client will communicate with Java code currently available in PaRoLa, which already implements several kinematic calculations that are necessary for performing the presented simulations and analyses.

REFERENCES

Alioon, Y. and Delialioğlu, O. (2017). The effect of authentic m-learning activities on student engagement and motivation. *British Journal of Educational Technology*.

Asabere, N.Y. (2013). Benefits and challenges of mobile learning implementation: Story of developing nations. *Intl Journal of Computer Applications*, 73(1), 23–27.

Bhosale, Y.S. and Livingston, J. (2014). V-lab: a mobile virtual lab for network security studies. *Intl Journal of Computer Applications*, 93, 35–38.

Bhullar, M.S. (2014). A new method of learning: M-learning (mobile learning). In *2014 9th International Conference on Computer Science & Education*, 322–325.

Bottentuit Jr., J.B. and Coutinho, C. (2007). Virtual laboratories and m-learning: learning with mobile devices. In *Proceedings of International Multi-Conference on Society, Cybernetics and Informatics*, 275–278.

Esquembre, F. (2004). Easy java simulations: a software tool to create scientific simulations in java. *Computer Physics Communications*, 156(2), 199–204.

Glavinic, V., Kukec, M., and Ljubic, S. (2007). Mobile virtual laboratory: Learning digital design. In *2007 29th International Conference on Information Technology Interfaces*, 325–332.

Glavinic, V., Kukec, M., and Ljubic, S. (2009). Digital design mobile virtual laboratory implementation: A pragmatic approach. In C. Stephanidis (ed.), *Universal Access in Human-Computer Interaction. Addressing Diversity*, 489–498. Springer, Berlin, Heidelberg.

Gosselin, C.M. and Sefrioui, J. (1991). Polynomial solutions for the direct kinematic problem of planar three-degree-of-freedom parallel manipulators. In *Fifth International Conference on Advanced Robotics (ICAR'91)*, volume 2, 1124–1129.

Jardim, R.R., Lemos, E., Herpich, F., Bianchim, R., Medina, R., and Nunes, F.B. (2014). U-lab cloud: A ubiquitous virtual laboratory based on cloud computing. In *The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM)*, 259–262.

Merabet, A., Akhrouf, S., Boubetra, D., Belhadj, F., Selmani, L., and Boubetra, A. (2015). Bba virtual laboratory through m-learning. In *2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)*, 181–184.

Oluwole, O., Nicolae, G., Olawale, O., and Oludele, A. (2015). Mobile virtual laboratory in Nigeria. *Intl J. of Engineering and Computer Science*, 4, 11417–11421.

Othayoth, R.S., Chittawadigi, R.G., Joshi, R.P., and Saha, S.K. (2017). Robot kinematics made easy using roboanalyzer software. *Comput Appl Eng Educ*, 25(5), 669–680.

Peidró, A., Gil, A., Marín, J.M., and Reinoso, Ó. (2016). A web-based tool to analyze the kinematics and singularities of parallel robots. *J. Intell. Robotic Syst.*, 81(1), 145–163.

Peidró, A., Reinoso, O., Gil, A., Marín, J.M., and Pay, L. (2015). A virtual laboratory to simulate the control of parallel robots. *IFAC-PapersOnLine*, 48(29), 19 – 24.

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

Saenz, J., Esquembre, F., García, F., de la Torre, L., and Dormido, S. (2016). A new model for a remote connection with hardware devices using javascript. *IFAC-PapersOnLine*, 49, 133–137.

Torres, F., Candelas, F., Puente, S., Pomares, J., Gil, P., and Ortiz, F. (2006). Experiences with virtual environment and remote laboratory for teaching and learning robotics at the University of Alicante. *Int. J. Engng Ed.*, 22, 766–776.