Modeling and Simulation of Spherical Parallel Manipulators in CoppeliaSim (V-REP) Robot Simulator Software

Iliyas Tursynbek and Almas Shintemirov
School of Engineering and Digital Sciences, Nazarbayev University
Nur-Sultan, Kazakhstan
iliyas.tursynbek@nu.edu.kz; ashintemirov@nu.edu.kz

***Abstract*—This paper presents a new methodology for modeling and simulation of spherical parallel manipulators (SPM) in CoppeliaSim (V-REP) robot simulator software. Using a SPM with coaxial input shafts as the case study, the steps for the robot SolidWorks CAD model importing, modeling in CoppeliaSim and interfacing with MATLAB for external control of the robot model are described in detail. The SPM motion simulation results are then presented and verified on an experimental 3D-printed system prototype.**

***Index Terms*—spherical parallel manipulator, robot simulation, collision detection, CoppeliaSim, V-REP**

I. Introduction

Parallel manipulators (PMs) are mechanisms with closed-loop kinematic chain architecture in which an end-effector, usually referred to as a moving/mobile platform, is connected to a fixed base through two or more chains. Contrary, serial manipulators are mechanisms which are comprised of a single open-loop kinematic chain. Thanks to their parallel architecture, PMs usually have a higher payload, are faster, more rigid, and more precise compared to their serial counterparts. Applications of PMs typically include pick-and-place robotized systems [1], orientation and stabilization platforms [2], pointing and tracking devices [3].

Despite of the above-mentioned advantages, PMs' main disadvantages are small workspace and complicated kinematic and dynamic analyses. These drawbacks make control of PMs a challenging task due to the necessity of the detailed kinematic analysis (especially, forward kinematic analysis) and obtaining Cartesian workspace and joint configuration space of a particular PM with the elimination of singular and self-collision configurations. Here, the singular configurations are referred to as the ones leading to a rank deficient Jacobian matrix of a manipulator [4], whereas the self-collision configurations are those resulting in a physical interference of two or more PM links.

Simulation tools are widely used for robot kinematics and dynamics modeling, motion optimizations, performance evaluation, and control strategy verification where simulations are performed before the experimental tests on real system prototypes. There is a plethora of simulation tools that allow to visualize and analyze kinematics and/or dynamics of PMs. Most of them are special simulators created for a particular problem or robot [5]–[10]. Among the general-purpose simulators, the most known example is *Gazebo* [11] developed by Open Source Robotics Foundation and used in DARPA Robotics Challenge [12]. Versions for Linux, macOS, Windows are available. It is also ROS [13] compatible. It supports the ODE, Bullet, Simbody and DART physics engines and can provide realistic 3D rendering. Custom plugins for robots and integrated sensors can be developed through Gazebo API. It allows dynamic robot bodies with scripting based on C++. Currently, Gazebo is being refactored into the new Ignition Gazebo simulator. Other popular open-source robot simulator is *Webots* [14] developed by Cyberbotics Ltd primarily for mobile robot simulations. The software runs on Windows, Linux and macOS. Robots may be programmed in C/C++, Python, Java, MATLAB or ROS through an API. The simulator uses ODE as the physics engine. *RoboDK* [15] is a proprietary multi-platform software that is mainly used for simulating industrial robotic arms. The RoboDK library includes over 400 industrial robot arms. The default RoboDK API is provided in Python, C++, MATLAB, etc. Analysis of the reviewed simulators revealed that they are not well suited for simulating closed kinematic chains and require advanced knowledge of the software.

One of the standard tools used for multibody dynamics simulation of closed-loop kinematic chains is *Adams* developed by MSC Software. It is a proprietary software requiring user experience for working. Alternatively, a widely used in robotics research general-purpose robot simulation framework *CoppeliaSim* (formerly known as *V-REP*) [16] developed by Coppelia Robotics is freely available for academic purposes. The simulator offers a wide functionality that can be easily integrated and combined through an embedded scripting and exhaustive API based on Lua, C/C++, Python, Java, MATLAB/Octave and ROS programming interfaces. CoppeliaSim supports different physics engines such as ODE, Bullet, Vortex, Newton. It can handle the inverse/forward kinematics of any type of mechanism with workspace visualization, sensors simulation, distance calculation, and collision detection. It has built-in examples of several PMs.

In this paper, the authors present the methodology for modeling and simulation of spherical parallel manipulators with coaxial input shafts (coaxial SPM) currently researched at ALARIS Laboratory (https://www.alaris.kz) of Nazarbayev University. It is a special case of a 3-DOF SPM capable of

This research was partially supported from the Nazarbayev University faculty development research grant project #090118FD5340.

infinite rotation around any axis within its workspace. Due to its user-friendly interface, link collision detection capabilities and availability of external API with MATLAB, CoppeliaSim software was selected as the robot simulation tool, realizing visual demonstration of the system motion, precomputed in MATLAB. Extensive publication and documentation review revealed the lack of detailed literature dedicated to modeling and motion control of PMs in CoppeliaSim. Therefore, the main contribution of this paper is an attempt to address this issue by presenting the coaxial SPM modeling and simulation in detail as a tutorial case study. The results of this work constitute a groundwork for the development of motion control algorithms for the considered SPM. Moreover, the methodology presented in this paper can be adopted to modeling other PM topologies.

The paper is organized as follows. Section II describes kinematics of the coaxial SPM with details on its 3D modeling in SolidWorks CAD software and manufacturing of a real 3D printed prototype, used for validating preliminary simulations. Section III presents details on importing the coaxial SPM 3D model to CoppeliaSim, and outlines the steps required for setting up simulations and linking with MATLAB using the remote API. Examples of the coaxial SPM motion control and self-collision detection are presented in Section IV. Finally, the paper is concluded with Section V, where a future work plan is outlined.

II. Coaxial SPM Model

A. Kinematics

Kinematic analysis of the coaxial SPM originates from that of the general SPM, which is an extensively studied topic [17]–[21]. This section presents a summary of the coaxial SPM's geometric and kinematic models, and foundations of its kinematic analysis.

The upper platform of the coaxial SPM, known as the *mobile platform*, is connected to the stationary *base* through three equally-spaced legs numbered as $i = 1, 2, 3$ in the counter-clockwise direction, and each of them is composed of two curved *links*: *proximal* (lower) and *distal* (upper). Angles α_1 and α_2 define the curvature of these links, respectively. The axes of the base, the intermediate, and the mobile platform joints intersect at the *center of rotation* and are defined by the unit vectors $\mathbf{u}_i$, $\mathbf{w}_i$, and $\mathbf{v}_i$, respectively, directed from the center of rotation towards the respective joints. The actuated joints $\mathbf{u}_i$ are coaxial to each other as demonstrated in Fig. 1a.

The stationary right-handed orthogonal coordinate system with its origin located at the center of rotation is shown in Fig. 1a. The z-axis is normal to the base and is directed upwards, while the x-axis is located in the plane generated by the z-axis and a central vertical plane of the proximal link 1 at the robot home configuration. The y-axis is determined by the right-hand rule. Angle β defines inclination of the mobile platform joints $\mathbf{v}_i$ from the z-axis.

The home configuration of the coaxial SPM is chosen such that all three proximal links are located 120° apart. In this case, the mobile platform is horizontal and its normal

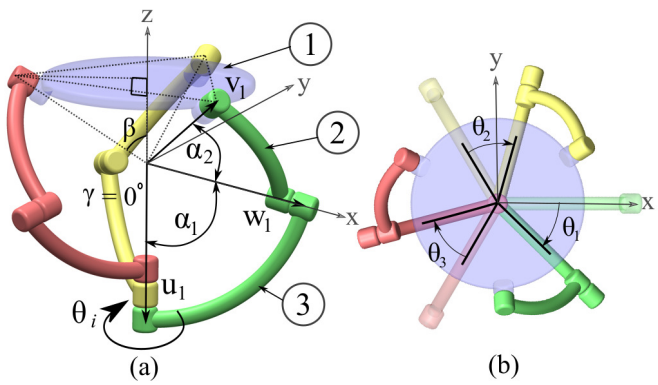


Fig. 1: Coaxial SPM kinematic model: (a) coordinate system and notation, where 1 - mobile platform, 2 - distal link, 3 - proximal link; (b) positive direction of the actuated joint positions with respect to the robot home configuration (shown as transparent).

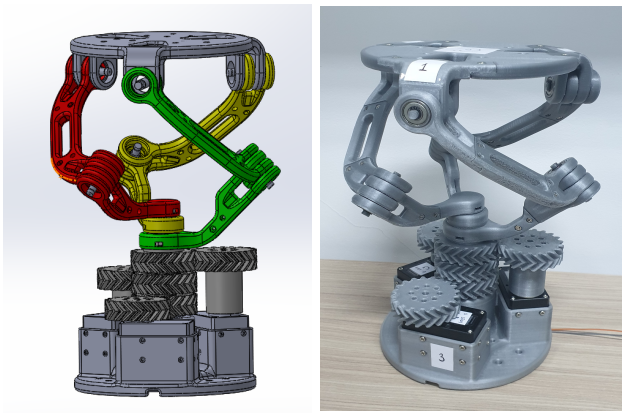
vector coincides with the positive z-axis. Input joint positions constituting vector $\boldsymbol{\theta} \triangleq \left[\theta_1, \theta_2, \theta_3\right]^T$ are measured from the planes defined by the z-axis and the unit vectors $\mathbf{w}_i$, $i = 1, 2, 3$ at SPM's home configuration to the planes of proximal links of the corresponding SPM legs with the clockwise direction being the positive direction as illustrated in Fig. 1b. At the home configuration, the vector of input joint positions is set to $\boldsymbol{\theta} = \left[0, 0, 0\right]^T$.

Once the geometric model of the manipulator is defined, its kinematic model can be derived with all mathematical relations between Cartesian and joint spaces. In [18] it was shown that this type of manipulators have 8 solutions to forward and 8 solutions to inverse kinematic problems. These solutions were analyzed by the authors in the context of the coaxial SPM in [22]. In the same work the approach for computation of unique kinematic solutions was presented allowing identification of the solutions corresponding to the same assembly mode of the manipulator as a necessary condition for their further use in the SPM control system design [23] based on the analysis of the space of feasible configurations [24].

B. 3D Modeling and Prototype Manufacturing

The coaxial SPM with geometrical parameters $\alpha_1 = 45^\circ$, $\alpha_2 = 90^\circ$, and $\beta = 90^\circ$ was created. A 3D CAD model of the coaxial SPM assembly corresponding to these parameters was designed in Solidworks CAD software (www.solidworks.com) and is presented in a rendered form in Fig. 2a.

Experimental verification of CoppeliaSim simulations was done using a physical prototype of the coaxial SPM, manufactured using 3D printing technology and assembled as demonstrated in Fig. 2b. The prototype dimensions are L 200 mm x W 200 mm x H 290 mm (bounding box at the home configuration). Actuation of the coaxial base joints is performed by three ROBOTIS Dynamixel XM540-W150 servomotors (http://en.robotis.com/) fixed on the SPM base platform in a circular arrangement being equally distributed



(a) (b)

Fig. 2: (a) A 3D CAD design and (b) an experimental 3D-printed prototype of the coaxial SPM.

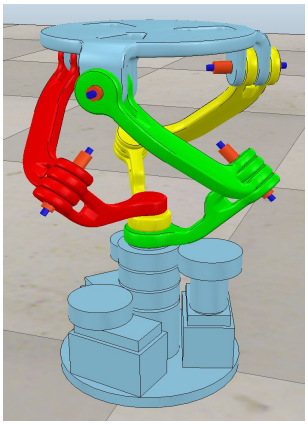


Fig. 3: CoppeliaSim simulation model of the coaxial SPM.

with 120° between each other. The central vertical planes of the proximal links are coincident with the corresponding straight lines connecting centers of actuating gear pair, at the home configuration. The actuators are controlled from MATLAB using Dynamixel SDK (Protocol 2.0) API. Double helical gears are used for transferring actuator torques to proximal links with gear ratio 1:1.

III. Modeling and Simulation in CoppeliaSim

A. Creating Coaxial SPM Model in CoppeliaSim

The first step of the robot simulation procedure is to create a properly defined manipulator model in CoppeliaSim satisfying all mechanical constraints, such that motion simulation of the manipulator model resembles behavior of its real physical prototype. The manipulator model designed in SolidWorks software was imported to CoppeliaSim environment. For importing the CAD model from an external application, it is important to use a not heavy CAD model, i.e. without too many triangles. Otherwise, a heavy CAD model would slow down graphical visualization and various calculation modules. For this reason the CAD model imported to CoppeliaSim was simplified in SolidWorks by removing all noncrucial design features such as holes and small details. Once finished, the manipulator CAD model was saved in STL format, one of the file formats supported by CoppeliaSim. The whole manipulator assembly was imported to the simulator as a single mesh and number of triangles contained in this mesh was reduced using the *decimate the mesh* function. Next, the *automatic mesh division* function was used to split the manipulator model to separate links. The base platform, actuators and gears were simplified and merged together to represent a single stationary base link, thus decreasing computational costs. Then, the imported CAD model was reoriented such that coordinate system of the model matches that of the kinematic model described in Section II. The resulting CoppeliaSim model of the coaxial SPM is presented in Fig. 3 and consists of 7 shapes representing the manipulator links: the base, 6 mobile links, and mobile platform. To improve visual aspects of the simulation, the legs were colored in different colours. After these preparatory steps were accomplished, the model was ready to be assembled resembling closed-loop chain kinematic architecture of the manipulator.

The resulting scene hierarchy is shown in Fig. 4. There are 2 closed loops in this scene. The first one is *base - proximal link 3 - distal link 3 - mobile platform - distal link 1 - proximal link 1*, and the second one is *base - proximal link 3 - distal link 3 - mobile platform - distal link 2 - proximal link 2*. In order to close the loop it is necessary to use *dummy* objects that are connected to the base and the last elements in the loops; all of them should located in the center of the base. It is due to those dummy objects the simulator can simulate PMs.

In order to make the *kinematics* calculation module to work properly the *IK groups* have to be enabled, added, and activated in the simulator settings. In the case of the coaxial SPM it is necessary to have 2 *IK groups* with 2 *IK elements* in each (*closure-tip-1* linked to *closure-target-1*, and *closure-tip-2* linked to *closure-target-2*, also shown in Fig. 4 with dashed arrows). All constraints (*X, Y, Z, Alpha-Beta, Gamma*) have to be enabled in the settings of *IK groups*.

The Damped Least Squares (DLS) calculation method with the damping coefficient equal to 0.0001 was selected for the *kinematics* calculation module. It should also be noted that all joints used are revolute (red cylinders in Fig. 3) and are also used as the intermediate elements in the above-mentioned loops. The actuated joints (highlighted with red colour in Fig. 4) are set to the *inverse kinematic* mode in *Scene Object Properties* menu, whereas remaining joints are set to the *passive* mode.

One of the important features of CoppeliaSim robot simulator is its built-in *collision detection* module that allows fast interference checking between any *shape* or collection of *shapes*. It uses data structures based on a binary tree of oriented bounding boxes [16], [25] for accelerations. In order to have this feature in the simulation of the coaxial SPM it is necessary to enable all *collision detections* under *calculation module* settings. Afterwards, new *collision objects* have to

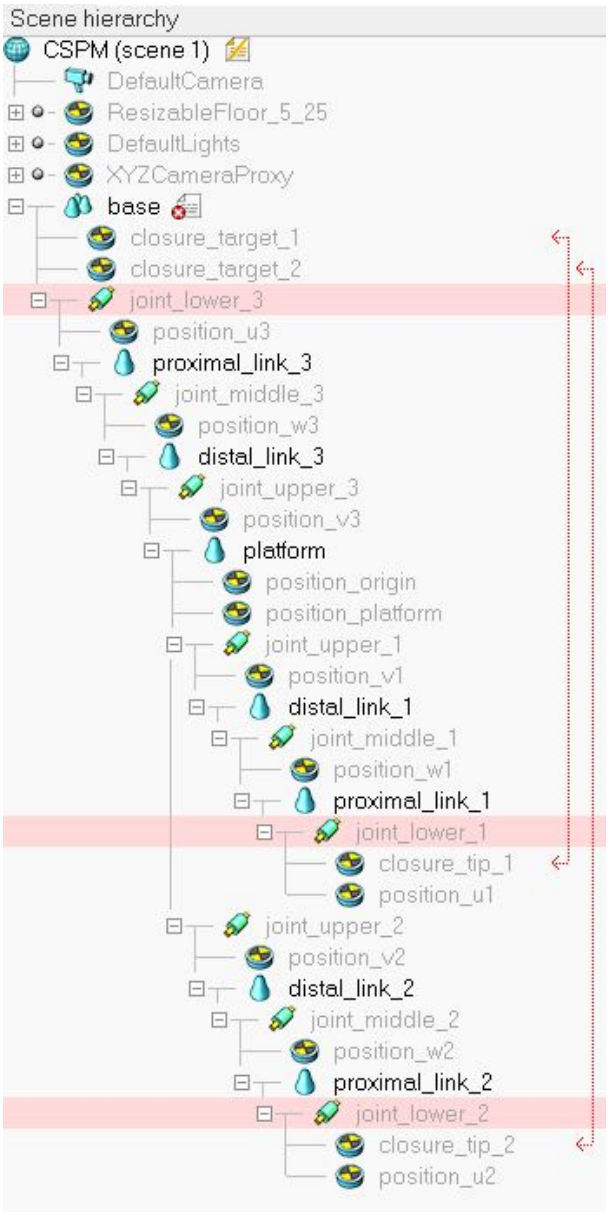


Fig. 4: CoppeliaSim scene hierarchy.

be added without enabling *explicit handling* mode. In this paper, collision checks between each link versus all other links (entities) have been implemented, resulting in 8 *collision objects*. Whenever the link collision happens CoppeliaSim changes link colours to indicate the collision event, and sends this information to an external client.

Each link in CoppeliaSim has a position and an orientation in the 3D space. In order to determine the position and the orientation of the link we can use either absolute coordinates or joint coordinates. Due to the fact that the origin of the global coordinate system is located in the center of the bottom surface of the base, *dummy* objects were added to the center of rotation and all joints (*position-origin*, *position-v1*, *position-v2*, *position-v3*, etc. in Fig. 4). In this way, the vectors representing these joints, i.e. $\mathbf{u}_i$, $\mathbf{w}_i$, and $\mathbf{v}_i$, are calculated as the difference between each link *dummy* and the center of rotation *dummy*.

B. Model Interfacing with MATLAB and Motion Control

The manipulator model created in CoppeliaSim was interfaced with MATLAB using the remote API link. The link allows running simulations directly from MATLAB with simultaneous data transfer between both programs. The mathematical model and equations described in [22] and implemented in MATLAB were used to verify simulated positions of the manipulator joints.

Generally, there are two ways to control PMs in CoppeliaSim. One way is by controlling the actuator positions (input angles) that through solving the forward kinematics problem would rotate the SPM top mobile platform. Alternatively, direct orientation control of the SPM top mobile platform will lead to solving the manipulator inverse kinematics problem. In this paper, the first approach was implemented.

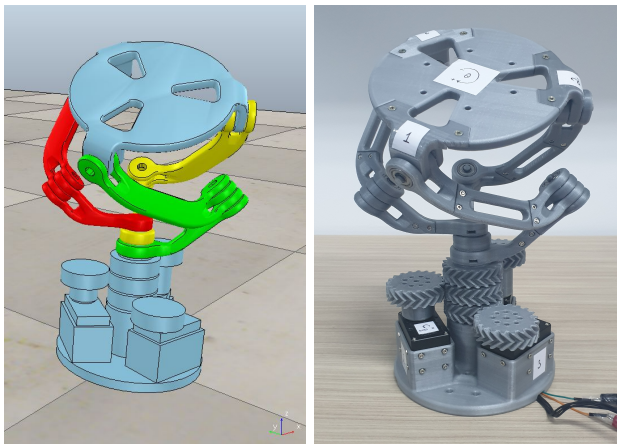
The first step in linking MATLAB to CoppeliaSim is to copy **vrchk.m**, **remoteApiProto.m**, **remApi.m**, and **remoteApi.dll** files from CoppeliaSim installation folder or website (www.coppeliarobotics.com) to MATLAB working folder. These files contain necessary API information and functions. After establishing communication with the simulator using *vrep.simxStart* function, the motion control of the simulated coaxial SPM can be realized. In order to be able to control simulated joints via MATLAB, at first, joint *handles* have to be obtained from the simulator using *vrep.simxGetObjectHandle* function. Afterwards, joint position can be read or varied using *vrep.simxGetJointPosition* and *vrep.simxSetJointPosition* functions, respectively. If collision detection *handle* is required to be returned to MATLAB, this is done using *vrep.simxGetCollisionHandle* and *vrep.simxReadCollision* functions. *Handles* are returned from CoppeliaSim in the form of unique numbers assigned to each object, joint positions are returned as angle values in radians, collision events variable takes values 1 when indicating the collision and 0 if no collision occurs.

The coaxial SPM simulation model is actuated via MATLAB commands sent to actuated joints (highlighted in red in Fig. 4). Before sending any commands to the simulator it is also necessary to verify that input joint vector $\boldsymbol{\theta}$ does not lead to a singular configuration. It is done in MATLAB using the approach for computing the coaxial SPM unique kinematic solutions presented in [22] and verifying that the Jacobian matrix at this specific SPM configuration is full-rank.

The same MATLAB-CoppeliaSim software configuration can be implemented on Linux platform, or configured in ROS using the remote API with Python scripting. Currently, authors are in the process of developing such arrangements for future integration of various sensors or other devices to the coaxial SPM via ROS framework.

IV. Simulation Results

Let's consider input joint positions $\boldsymbol{\theta} = \left[60°, 90°, 120°\right]^T$. To determine the resulting orientation of the coaxial SPM top



(a) (b)

Fig. 5: Simulated configuration (a), experimental prototype configuration (b).

mobile platform, i.e. to compute unique unit vectors $\mathbf{v}_i$, $i = 1, 2, 3$, as indicated in Fig. 1, the approach [22] was used. The resulting SPM orientation is obtained as follows

$$\begin{aligned} \mathbf{v}_{1,calc} &= \left[-0.8626,\ 0.0790,\ -0.4997\right]^T, \\ \mathbf{v}_{2,calc} &= \left[0.5002,\ -0.8659,\ 0.0003\right]^T, \\ \mathbf{v}_{3,calc} &= \left[0.3618,\ 0.7866,\ 0.5003\right]^T. \end{aligned} \tag{1}$$

The simulated orientation of the coaxial SPM has the following unit vectors $\mathbf{v}_i$, $i = 1, 2, 3$, obtained as the normalized difference between *dummy* objects attached to joints and the center of rotation:

$$\begin{aligned} \mathbf{v}_{1,sim} &= \left[-0.8623,\ 0.0794,\ -0.5001\right]^T, \\ \mathbf{v}_{2,sim} &= \left[0.5001,\ -0.8660,\ 0.0000\right]^T, \\ \mathbf{v}_{3,sim} &= \left[0.3622,\ 0.7866,\ 0.5001\right]^T. \end{aligned} \tag{2}$$

Configuration of the coaxial SPM corresponding to input joint positions $\boldsymbol{\theta} = \left[60°, 90°, 120°\right]^T$, both simulated and experimentally tested on the coaxial SPM physical prototype, can be observed in Fig. 5.

Time evaluations of the mobile platform orientation vectors $\mathbf{v}_i$, $i = 1, 2, 3$, during the SPM motion from the SPM home configuration to the configuration corresponding to $\boldsymbol{\theta} = \left[60°, 90°, 120°\right]^T$ are shown in Fig. 6. The unit vectors $\mathbf{v}_i$, $i = 1, 2, 3$, are represented individually by their x, y, and z components. It is seen that simulated motion of the coaxial SPM closely matches the expected numerically precalculated behavior. This comparison is done based on 25 samples and interpolation values between them.

Considering another SPM configuration with input joint positions $\boldsymbol{\theta} = \left[60°, 90°, 220°\right]^T$, we can observe the simulated link collision event detected by CoppelliaSim built-in collision detection module as illustrated in Fig. 7. Here, the collision between four SPM links, i.e. proximal link 2, distal link 2, proximal link 3 and distal link 3, happened and is indicated by the changed colours (to black) of the links.

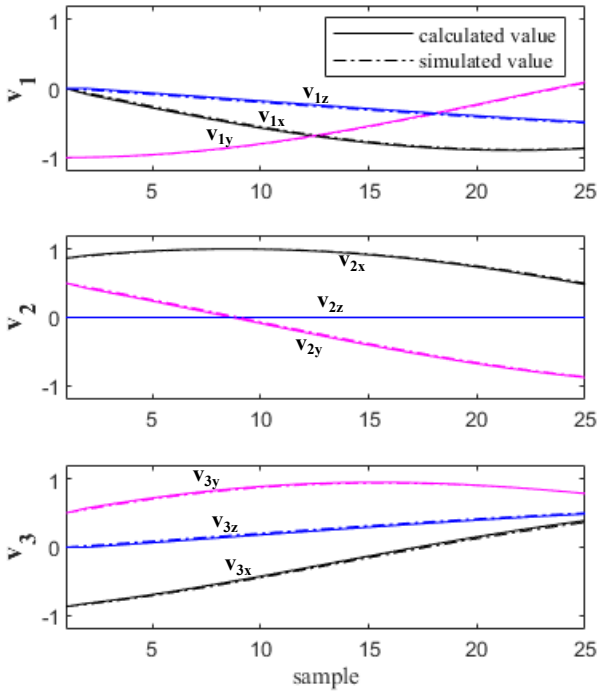


Fig. 6: Calculated and simulated orientations of the coaxial SPM.

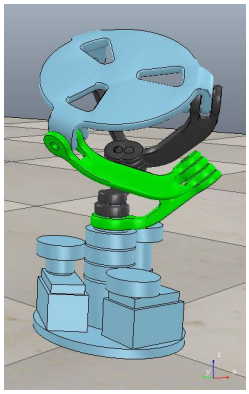


Fig. 7: Simulation model of the coaxial SPM with detected link collision (collided links are coloured in black).

In overall, based on the presented and other numerous computation results and experimental trials on placing the manipulator to various arbitrary configurations, it was observed that the CoppeliaSim simulated SPM model fully replicated the motion and resulting orientations of the coaxial SPM physical prototype (at this stage this was verified only visually) and correlated very well with numerically calculated SPM orientations. The accompanying video demonstration is available at the author's research lab website https://www.alaris.kz.

V. Conclusions and Future Work

In this paper, a new methodology was presented for modeling and simulation of parallel manipulators in CoppeliaSim robot simulator based on the example of a SPM with coaxial input shafts. Kinematics of this manipulator was discussed followed by the detailed explanation of CoppeliaSim modeling steps and the model control from MATLAB using an interface CoppeliaSim-MATLAB API. Simulation examples supported by experimental verification using a real physical prototype of the manipulator were demonstrated. It was shown that proposed methodology allows accurate simulation of the considered parallel manipulator configuration and can be adopted for modeling and motion simulation of other closed kinematics robot systems. Furthermore, the adopted in this study MATLAB based control approach of the manipulator's CoppeliaSim model has proven to be a promising testing/simulation tool for motion analysis and collision detection of PMs, that could be potentially useful for design optimization of manipulators and other applications.

This work is setting the basis for future extensions of authors' research work on SPMs such as generation and simulation of singularity and collision-free workspaces for manipulator motion and path planning and analysis. As immediate future work, we plan to further develop and test in simulations the coaxial SPM orientation control using the presented approach for ensuring the robot collision-free behavior before experimental trials on the real physical prototype of the manipulator. Another idea is to use the proposed methodology in developing learning algorithms (e.g. reinforcement learning) verified and tested via simulations.

References

[1] L. Rey and R. Clavel, "The Delta parallel robot," in *Parallel Kinematic Machines*, pp. 401–417, Springer London, 1999.

[2] D. Stewart, "A platform with six degrees of freedom," *Proceedings of the Institution of Mechanical Engineers*, vol. 180, no. 1, pp. 371–386, 1965.

[3] C. Gosselin and J.-F. Hamel, "The Agile Eye: a high-performance three-degree-of-freedom camera-orienting device," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation (ICRA 1994)*, pp. 781–786, 1994.

[4] C. Gosselin and J. Angeles, "Singularity analysis of closed-loop kinematic chains," *Robotics and Automation, IEEE Transactions on*, vol. 6, pp. 281 – 290, 1990.

[5] R. Diankov, *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[6] V. Petuya, E. Macho, O. Altuzarra, C. Pinto, and A. Hernandez, "Educational software tools for the kinematic analysis of mechanisms," *Computer Applications in Engineering Education*, vol. 22, no. 1, pp. 72–86, 2011.

[7] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[8] J. M. Porta, L. Ros, O. Bohigas, M. Manubens, C. Rosales, and L. Jaillet, "The CUIK suite: Analyzing the motion closed-chain multibody systems," *IEEE Robotics & Automation Magazine*, vol. 21, no. 3, pp. 105–114, 2014.

[9] T. Habra, H. Dallali, A. Cardellino, L. Natale, N. Tsagarakis, P. Fisette, and R. Ronsse, "Robotran-YARP interface: A framework for real-time controller developments based on multibody dynamics simulations," in *Computational Methods in Applied Sciences*, pp. 147–164, Springer International Publishing, 2016.

[10] V. Gupta, R. G. Chittawadigi, and S. K. Saha, "RoboAnalyzer," in *Proceedings of the Advances in Robotics on - AIR '17*, ACM Press, 2017.

[11] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2004.

[12] E. Krotkov, D. Hackett, L. Jackel, M. Perschbacher, J. Pippine, J. Strauss, G. Pratt, and C. Orlowski, "The DARPA robotics challenge finals: Results and perspectives," in *Springer Tracts in Advanced Robotics*, pp. 1–26, Springer International Publishing, 2018.

[13] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, 2009.

[14] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[15] A. Montaqim, "Offline programming software for industrial robots from RoboDK offers hundreds of virtual industrial robots from top robotics companies," *Robotic and Automation News*, 2015.

[16] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.

[17] C. Gosselin and E. Lavoie, "On the kinematic design of spherical three-degree-of-freedom parallel manipulators," *The International Journal of Robotics Research*, vol. 12, no. 4, pp. 394–402, 1993.

[18] C. M. Gosselin, J. Sefrioui, and M. J. Richard, "On the direct kinematics of spherical three-degree-of-freedom parallel manipulators of general architecture," *Journal of Mechanical Design*, vol. 116, no. 2, pp. 594–598, 1994.

[19] S. Bai and M. R. Hansen, "Forward kinematics of spherical parallel manipulators with revolute joints," in *Proceedings of the 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2008)*, 2008.

[20] A. Shintemirov, A. Niyetkaliyev, and M. Rubagotti, "Numerical optimal control of a spherical parallel manipulator based on unique kinematic solutions," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, pp. 98–109, 2016.

[21] T. Taunyazov, M. Rubagotti, and A. Shintemirov, "Constrained orientation control of a spherical parallel manipulator via online convex optimization," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 252–261, 2018.

[22] I. Tursynbek, A. Niyetkaliyev, and A. Shintemirov, "Computation of unique kinematic solutions of a spherical parallel manipulator with coaxial input shafts," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pp. 1524–1531, IEEE, 2019.

[23] I. Bonev, D. Chablat, and P. Wenger, "Working and assembly modes of the Agile Eye," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation ICRA 2006*, pp. 2317–2322, 2006.

[24] I. Tursynbek and A. Shintemirov, "Infinite torsional motion of a spherical parallel manipulator with coaxial input axes," in *Proceedings of the 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2020)*, 2020.

[25] S. Gottschalk, M. C. Lin, and D. Manocha, "Obb-tree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 171–180, 1996.