Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#

Sudip Chakraborty[1] & P. S. Aithal[2]

[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India

OrcidID: 0000-0002-1088-663X; E-mail: sudip.embedded@gmail.com

[2]ViceChancellor, Srinivas University, Mangalore, India

OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: Business Management.
Type of the Paper: Simulation based Research.
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI: http://doi.org/10.5281/zenodo.
Google Scholar Citation: IJAEML.

How to Cite this Paper:
Chakraborty, Sudip, & Aithal, P. S., (2021). Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(1), 29-37. DOI: http://doi.org/10.5281/zenodo.

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Forward Kinematics Demonstration of 6DF Robot using CoppeliaSim and C#

Sudip Chakraborty[1] & P. S. Aithal[2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.embedded@gmail.com
[2]ViceChancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: Robot researchers need the simulator to test their functions and algorithm before implementing into the real Robot. When we search for a good simulator, a customizable Robot simulator is not available to the researcher. Some good options are available in the market, but their price is too high, not affordable. After much research, we found a better solution around all aspects, fully customizable and entirely free of cost. There are two sides. One is a 3D view of robotics ARM, and another is a joint controller of robotics ARM. For 3D simulation, CoppeliaSim is the best option. It is free of cost and Open Source. The C# language is becoming famous for its managed code structure and ease of implementation. Our Robot controller is in the C# language. Microsoft visual studio is the best IDE to control the simulator. It is an unparallel IDE for the programmer. In the CoppeliaSim little bit, Lua script is in associated functions. For Communication Between the CoppeliaSim and our application, we will use the Remote API Framework.
Design/Methodology/Approach: Here we are using the prebuild robot model of CoppeliaSim IDE (Integrated Development Environment) to save the time and attach Lua scripts associated with robot object. From Visual studio IDE we control the Virtual Robot through TCP/IP socket object. Using GUI (Graphical User Interface) we send the command to the robot.
Findings/Result: The researcher who is developing robotics arm, is required the vast knowledge of position and rotation of different joints. Changing the joint value using the GUI element, we can observe the various robot pose. The researcher can proceed further Sending the multiple sequential command. Thus, we can simulate industrial process automation.
Originality/Value: Using both CoppeliaSim and visual studio IDE can create a better environment for our robotics research.
Paper Type: Simulation**.**
Keywords: Robot IDE, Robot Simulator, Robot Programming, 3D simulator.

1. INTRODUCTION :

Our current trend is to use robots in the industry as well as in the domestic field. From morning till at night, directly or indirectly gets the facility of robots. Various types of Robots we use according to our different purpose. Mobile Robot is used for transportation like food delivery, and the crewless vehicle is used in the industry production pipeline. Non-mobile Robot is a robotics arm that has a certain degree of freedom. A six degree of freedom robot is generally used in industry for various activities. It is also used for car painting, welding, pick and place, packaging, and many more. Every robot development is required an integrated development environment where the researcher can develop and test their algorithm. Nevertheless, flexible ide is not readily available. In a developing country like India, researchers whose financial strength is low did not get sufficient funds quickly to continue their research. To obtain excellent and reliable results, need lots of hardware. For software perspective, most of the time, researcher finds free or Opensource Robot Simulator for their experiments.
Here we introduce a good simulator for robotics research. The name of the simulator is CoppeliaSim from the creator of Vrep. It is an opensource and completely free. It has a lot of prebuilds industrial robots, and anyone can try them quickly. The researcher can also build and test their Robot. It has one

drawback. It is not a good debugger. To obtain the debugging solution, we use another good IDE from Microsoft Visual Studio. It is also Free for community edition, which is sufficient for us. Our backbone of the Communication between CoppeliaSim and Visual Studio is TCP/IP socket 6000. When the simulator runs, a server socket creates and waits for a command from the client. The robots do not start any movement until they do not interact through the User Interface of the visual studio application. For six joints, there are six sliders. Each slider value shows into the corresponding textbox. When the user changes any joint angle through the slider or typing in the textbox, the joints value sends to the simulator within the one-millisecond interval. The simulator sends an acknowledgment upon command reception. Receiving the command from the client, the server purse it and trigger the corresponding actuator.

2. RELATED WORKS :

Boris Bogaerts et al., in their paper, demonstrates the Communication between CoppeliaSim and VR environment. A video is rendering with the VR Interface application renders in CoppeliaSim. They created a separate application for the VR rendering, instead of integrating them into the simulator, to guarantee a consistent and fast framerate of the VR visualization, even when the simulator's simulation speed is inconsistent. In the CoppeliaSim, they used LUA script, and no external world communicated they created. They developed a module. It was not clear which software they used. Also lack of experiment procedures so the other researcher can recreate their experiment [1].
Saharsh Oswal and D Saravanakumar, in their paper "Line following robots on factory floors: Significance and Simulation study using CoppeliaSim," presents the design, assembly, and dynamic simulation of a line following integrated with a proximity sensor for collision avoidance and vision sensors for line tracking. They use the native Lua script in the simulator environment. Their experiments are appliable for mobile Robots. It is not used IK chain, which needed most of the industrial environment. Moreover, their experiments have not remote Communication where we can implement our debug IDE. Also, there is some scope to improve the debugging. It would be better to communicate with Robot through socket communication and debug the robots most comfortably [2].

Tamir Blum et al. have developed a reinforcement learning (RL) platform which was made in a modular way to allow for sharing and collaboration, to be straightforward to use and simple to build upon; there are many hurdles doing research challenging for robotic space applications using RL and machine learning, mainly due to insufficient resources for traditional robotics simulators like CoppeliaSim. Their solution to this is an open-source modular platform called Reinforcement Learning that simplifies and accelerates RL's application to the space robotics research field. Lack of their experiments is, no proper documentation to the researcher on code structure or not given module details [3].

Marc Freese et al., their experiments allow the user to pick from them as requires or as best for a given simulation. Moreover, special care has been given to develop those functionalities in a balanced manner, giving each one the same amount of attention as one would do in a real robotic system. Control is distributed and based on an unlimited number of scripts directly associated or attached to scene objects. We did not find any best way to debug the code because debug is one prime concern to the programmer for critical issues findings [4].

3. OBJECTIVES :

The objective of the research works are:
- To find a better IDE for our robotics research.
- Introduce C# language for robotics control and automation field, which has a useful framework.
- Demonstration of forward kinematics in a straightforward manner to the researcher.
- Find out a reliable, robust communication channel for Robot Control.

4. METHODOLOGY :

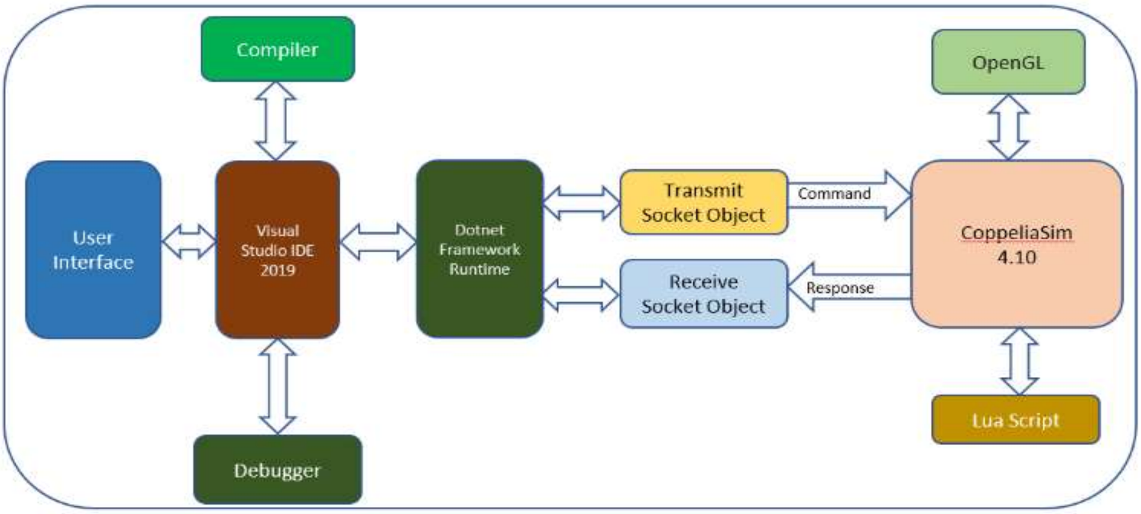

Fig. 1 : The block diagram of Data flow architecture

In the Fig. 1 depicts the complete architecture of our research work. The left portion is for control and debug, and the right side is for the CoppeliaSim part. Two application is required for our experiment. One is CoppeliaSim, and the other is Microsoft visual studio. CoppeliaSim can be download from *https://www.coppeliarobotics.com/downloads*. And for, visual studio downloads community edition from https://visualstudio.microsoft.com/downloads/.

Open the CoppeliaSim. From file menu open the scene which is located at C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\scenes\ik_fk_simple_examples. Furthermore, it is now time to attach the below script associated with the functions.

```
--------------------------------------------------------------
function sysCall_actuation()


    local client = server:accept()
    --print("request received")
    client:settimeout(10)
    local packet, err = client:receive()
    if not err then
        --print("received from client=",packet)

        cmd=string.sub(packet, 2,3)
        --------------------------------------------------
        if string.find(cmd, "FK") then

            i= string.find(packet, "J1=")
            j= string.find(packet, ",J2=")
            j1=string.sub(packet, i+3,j-1)
            ----------------------------------
            i= string.find(packet, "J2=")
            j= string.find(packet, ",J3=")
            j2=string.sub(packet, i+3,j-1)
            ----------------------------------
            i= string.find(packet, "J3=")
            j= string.find(packet, ",J4=")
            j3=string.sub(packet, i+3,j-1)
            ----------------------------------
            i= string.find(packet, "J4=")
            j= string.find(packet, ",J5=")
            j4=string.sub(packet, i+3,j-1)
            ----------------------------------
            i= string.find(packet, "J5=")
            j= string.find(packet, ",J6=")
            j5=string.sub(packet, i+3,j-1)
            ----------------------------------
            i= string.find(packet, "J6=")
            j= string.find(packet, ">")
            j6=string.sub(packet, i+3,j-1)
            ----------------------------------
```

Fig. 2: The code to purse of receive packet from remote client.

```
                -------------------------------
                sim.setJointPosition(simJoints[1],tonumber(j1))
                sim.setJointPosition(simJoints[2],tonumber(j2))
                sim.setJointPosition(simJoints[3],tonumber(j3))
                sim.setJointPosition(simJoints[4],tonumber(j4))
                sim.setJointPosition(simJoints[5],tonumber(j5))
                sim.setJointPosition(simJoints[6],tonumber(j6))
            end
            -----------------------------------------------
```

```
            -----------------------------------------------
            client:send(packet .. "\n")
        end
        -- print("closing socket")
         client:close()

end
-------------------------------------------------------------------------------
function sysCall_cleanup()
    simIK.eraseEnvironment(ikEnv) -- erase the IK environment
end
```

Fig. 3: The code to purse of receive packet from remote client.

After writing the code, if we press the run button, the system will not do anything. We will start now another side pending task. Open Microsoft Visual Studio 2019. Create a Windows Desktop Application and language as C#. Design the user interface like this. Double click the start button. It will navigate the associated function. We Create a TCP socket using port 6000. A status flag will indicate connection successfully, then send the data as below format. If the joint angle change, we can see the reflection in the simulator.

<FK#j1=12.45, J2=33.25, J3=56.23, J4=78.25, J5=45.22, J6=0.00>

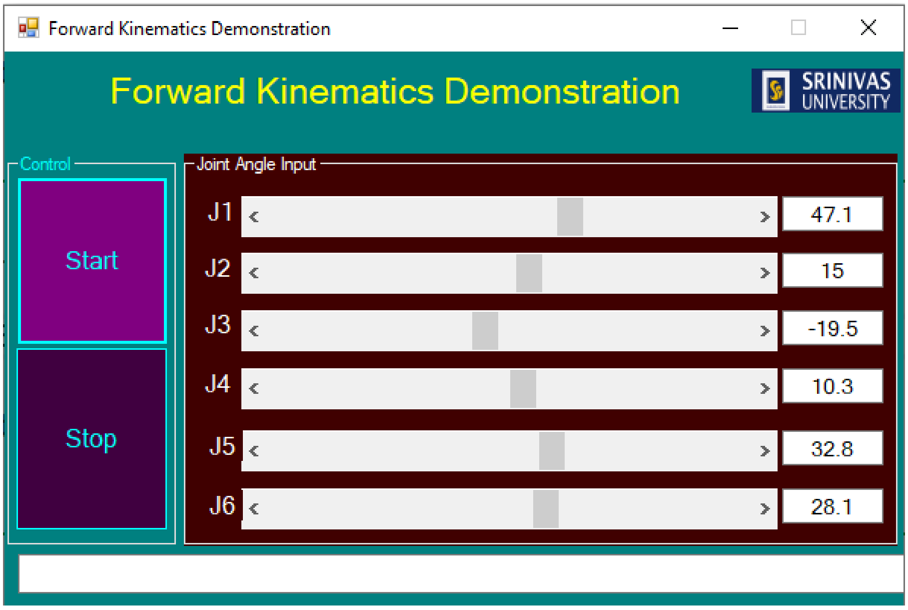

Fig. 4: The user interface to change the joints value.

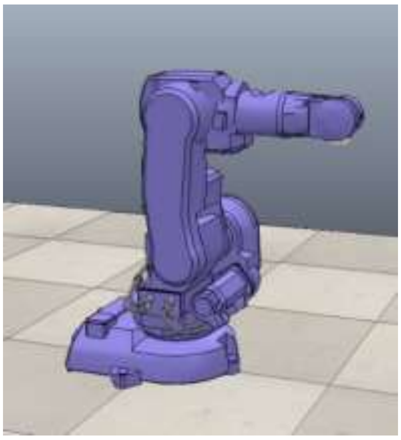

Fig. 5: The built-in robot for our research work.

5. FLOW DIAGRAM :

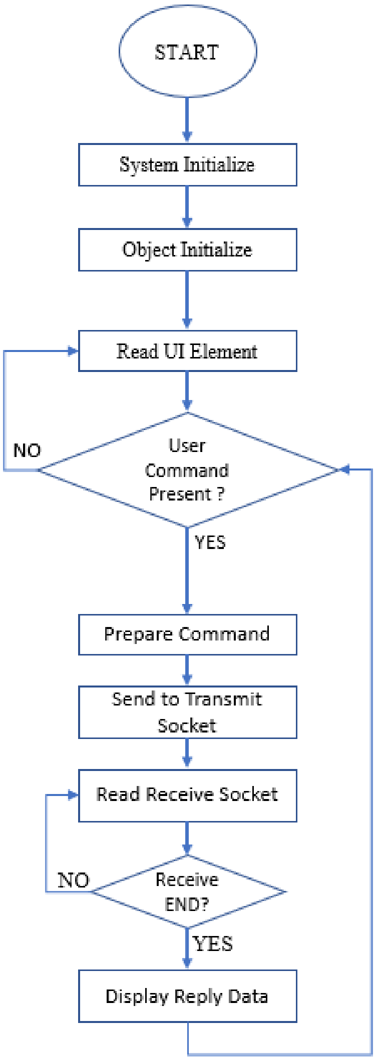

Fig. 6: The process flow diagram

Now we can look at our flow diagram. We must start the CoppeliaSim before our dot net application run. After our application starts, the system has initialized some components. Then The robot object is configured to start the operation. After initializing the robotics object, start a timer with a 1millisecond interval. It checks any user interaction that happens in 1 millisecond. If no user input is found, no action is taken, change the slider value. The system detects the user input then it starts to prepare the packet to send through the transmit socket.

After Making the packet ready, the application starts connecting with Robot through inter-process Communication via socket. If successful Communication happens, the simulator receives the command and processes it. Pursing the joint angles triggers the motor to the Robot and sends response data to our application. If the application was not received with five milliseconds, a timeout occurred and again attempted to execute the operation. After receiving the application's response, it purses and places it into the respective message box. After completing the one execution, again, it starts to receive attention from the user input.

We have some scope to use this type of Communication. Using socket communication, it can be possible to use a different system. In that scenario, we need two computers. We have to run the Robot simulator on another machine, and we can run our application. Before starting our application, the robot simulator should run first. Then our application starts. If unable to communicate application, an exception occurred, and an error message should display in the message box. If we use a communication medium as the internet, we can test this experiment around the glove. If one system in the USA and our application runs in India, it will run without an issue. We network latency may occur. In that case, an IP field has to be entered of remote pc IP where our robot simulator is running.

6. RECOMMENDATIONS :

After completing the task and observing all factors, we can recommend for robotics researcher who can execute the accurate result and get the system's best performance. The below steps to follow:

- ➢ The first step selects a suitable robot for our needs.
- ➢ Write script into the simulator.
- ➢ Write code into visual studio.
- ➢ First, run the simulator, then run the visual studio.
- ➢ Debug the process. Can set breakpoints where we desired and check the variable value.
- ➢ For sensor input, connect the sensor as a serial port. Create serial object and write associated action for that.
- ➢ Test the functions and algorithm in the simulator before implementing them into the real Robot. Otherwise, within a second, a robot can damage property and life. So, the unwritten truth is that we should always test the algorithm before update the firmware.
- ➢ For code found in the link: https://github.com/sudipchakraborty/Forward-Kinematics-Demonstration-Using-CoppeliaSim-and-C-

7. CONCLUSION :

Robotics experiments in the virtual environment are becoming popular. Many robotics researchers are working on virtual environments their proof of concept (POC) experiments worldwide. There are several causes for this popularity. First, the cost of the practical experiment is too high. Individual researchers cannot afford that much money. An expert should guide the experiment. Otherwise, it can damage property and life too. In the virtual experiment, all the above cause is nullified. So, it is rapidly grasping by robotics researchers. The CoppeliaSim is one of the open-source robotics software platforms that can simulate any robots with little effort. The CoppeliaSim and C# may build a better framework that is robust, failsafe, easy to use, and easy to debug.

REFERENCES :

[1] Boris Bogaerts, Seppe Sels, Steve Vanlanduit, Rudi Penne (2020). Connecting the CoppeliaSim robotics simulator to virtual reality. *SoftwareX*, *11*, 100426, 1-4.

[2] Saharsh Oswal and Saravanakumar, D. (2021). Line following robots on factory floors: Significance and Simulation study using CoppeliaSim. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1012, No. 1, p. 012008). IOP Publishing.

[3] Tamir Blum, Gabin Paillet, Mickael Laine, Kazuya Yoshida (2020). RL STaR Platform: Reinforcement Learning for Simulation-based Training of Robots. i-SAIRAS2020, 5032, 1-8.

[4] Marc Freese, Surya Singh, Fumio Ozaki, and Nobuto Matsuhira (2010). Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator. Conference Paper · November 2010. DOI: 10.1007/978-3-642-17319-6_8 · Source: DBLP.

[5] Rohmer, E., Singh, S. P., & Freese, M. (2013, November). V-REP: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1321-1326). IEEE.

[6] Huck, T. P. Ledermann, C. and Kröger, T. (2020). Simulation-based Testing for Early Safety-Validation of Robot Systems. 2020 IEEE Symposium on Product Compliance Engineering - (SPCE Portland), Portland, OR, USA, pp. 1-6, DOI: 10.1109/SPCE50045.2020.9296157.

[7] Kortmann M., Zeis C., de Alba-Padilla C. A., Grzesik B., K.-U. Schroeder and E. Stoll (2020). New approach on robotic arm design: fully modular arm architecture utilizing novel space interface. i-SAIRAS2020-Papers (2020), Virtual Conference 19–23 October 2020.

[8] Javier Pinzon-Arenas, Robinson Jimenez-Moreno, and Astrid Rubiano (2020), Virtual environment for smart robotic applications. *ARPN Journal of Engineering and Applied Sciences*, 15(22), 2698-2705.

[9] Igor Shardyko, Maria Samorodova, Victor Titov (2020), Decentralized Control of Robotic Arm with Elastic Joints. International Russian Automation Conference (RusAutoCon), 978-1-7281-6130-3/20, pp. 615-620.

[10] John Oyekan, Michael Farnsworth, Windo Hutabarat, David Miller and Ashutosh Tiwari (2020). Applying a 6 DoF Robotic Arm and Digital Twin to Automate Fan-Blade Reconditioning for Aerospace Maintenance, Repair, and Overhaul, Sensors 2020, 20, 4637; doi:10.3390/s20164637.

[11] Jiawei Hou, Yizheng Zhang, Andre Rosendo and S¨oren Schwertfeger (2020). Mobile Manipulation Tutorial. https://robotics.shanghaitech.edu.cn/static/robotics2020/MoManTu_Intro.pdf

[12] Jan Schneider, Tim Schneider, Boris Belousov, Georgia Chalvatzaki, Samuele Tosatto, Bastian Wibranek, (2020). Architectural Assembly with Tactile Skills: Simulation and Optimization. Retrieved from https://www.ias.informatik.tu-darmstadt.de/uploads/Team/BorisBelousov/schneider_ip.pdf

[13] Tamir Blum, Kazuya Yoshida (2020). PPMC RL Training Algorithm: Rough Terrain Intelligent Robots through Reinforcement Learning, arXiv:2003.02655. Retrieved from https://arxiv.org/pdf/2003.02655.pdf.

[14] Mehrnoosh Askarpour, Matteo Rossi, Omer Tiryakiler (2020). Co-Simulation of Human-Robot Collaboration: from Temporal Logic to 3D Simulation. Robots for reliable Engineered Autonomy (AREA'20). EPTCS 319, 1–8, DOI:10.4204/EPTCS.319.1.

[15] Omar Gamal, Xianglin Cai, Hubert Roth (2020). Learning from Fuzzy System Demonstration: Autonomous Navigation of Mobile Robots in Static Indoor Environment using Multimodal Deep Learning, *International Conference on System Theory, Control and Computing* (ICSTCC), 978-1-7281-9809-5/20, pp 218-225.

[16] Md Nizamuddin Ahmed (M.E), Veladri, K. (2016). Modeling and Simulation of 7-DOF Robotic Manipulator. *National Conference on Technological Advancements in Mechanical Engineering*, ISBN: 978-93-85100-57-4.

[17] Samuel Abhishek, S., Veladri, K. (2016). Trajectory Planning of a Mobile Robot. *National Conference on Technological Advancements in Mechanical Engineering*, ISBN: 978-93-85100-57-4.

[18] Tursynbek and Shintemirov, A. (2020). Modeling and Simulation of Spherical Parallel Manipulators in CoppeliaSim (V-REP) Robot Simulator Software. *International Conference Nonlinearity, Information and Robotics (NIR)*, Innopolis, Russia, pp. 1-6, DOI: 10.1109/NIR50484.2020.9290227.

[19] Zamfirescu and C. Pascal (2020). Modelling and simulation of an omnidirectional mobile platform with robotic arm in CoppeliaSim, 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, pp. 667-672, DOI: 10.1109/ICSTCC50638.2020.9259773.

[20] Rooban S., Shaik Dilawar Suraj, Shaik Babji Vali, Nagandla Dhanush, CoppeliaSim: Adaptable modular robot and its different locomotions simulation framework, Materials Today: Proceedings, 2021, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2021.01.055.

[21] Sun, Z., D. Li, L. Huang, Liu, B. and Jia, R. (2020), Construction of intelligent visual coal and gangue separation system based on CoppeliaSim, *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Dalian, China, pp. 560-564, DOI: 10.1109/CACRE50138.2020.9230077.

[22] Xinyang Tian, Qinhuan Xu, Qiang Zhan (2021), An analytical inverse kinematics solution with joint limits avoidance of 7-DOF anthropomorphic manipulators without offset, Journal of the Franklin Institute, 358(2), 1252-1272, ISSN 0016-0032.

[23] Premachandra, H. A. G. C., Thathsarana, K. M., Herath, H. M. A. N., Liyanage, D. L. F. M., Amarasinghe, Y. W. R., Madusanka, D. G. K., & Jayawardane, M. A. M. M. (2020). Design and Simulation of a Robotic Manipulator for Laparoscopic Uterine Surgeries. In *Innovation in Medicine and Healthcare*, *192*, (pp. 67-79). Springer, Singapore. https://doi.org/10.1007/978-981-15-5852-8_7.

[24] Zhang, J., Jin, L., and Yang, C. (2021). Distributed Cooperative Kinematic Control of Multiple Robotic Manipulators with Improved Communication Efficiency. *IEEE/ASME Transactions on Mechatronics*, DOI: 10.1109/TMECH.2021.3059441.

[25] Tursynbek I. and A. Shintemirov (2020). Infinite Torsional Motion Generation of a Spherical Parallel Manipulator with Coaxial Input Axes. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Boston, MA, USA, 2020, pp. 1780-1785, DOI: 10.1109/AIM43001.2020.9158791.

****