Robot Modeling, Motion Simulation and Off-line Programming Based on SolidWorks API

Nikolay Bratovanov
Robotics Lab, Faculty of Automation
Technical University of Sofia
Sofia, Bulgaria
nbratovanov@tu-sofia.bg

***Abstract*—The paper contributes to the development of a general-purpose modeling, motion simulation and off-line programming system applicable to various manipulators and mechanisms. It is based on the popular CAD software SolidWorks, widely used in many industries, and its application programming interface (API). Implemented as a SolidWorks macro (written in VBA), the system allows the direct usage of the existing robot 3D models created for manufacturing purposes and eliminates the need for additional motion simulation software, thus providing an integrated solution for the simultaneous execution of design and simulation tasks. The lightweight integration into SolidWorks and the effective interface to controllers or standalone simulators that generate the motion of the manipulators distinguish the proposed development from the currently available CAD-based robot motion simulators.**

Keywords-modeling; motion simulation; off-line programming; robotic systems; mechanisms; application programming interface; layout analysis; SolidWorks; Visual Basic for Applications;

I. INTRODUCTION

The benefits of robotics simulation software systems have been recognized by scientists and engineers, with applications ranging from simple robot path simulation to complete robotic cell layout analysis [1]. Various simulation tools that accurately represent the proposed robot and cell geometry, and the robot kinematic and dynamic performance are valuable instruments for optimizing design, verifying feasibility, designing workcell layout, verifying robot motion programs, and evaluating cell performance [2]. These useful features turn robotics simulation software systems into an essential element of modern and agile manufacturing plants, as they make possible to visualize and test the robotic system, even if it does not exist physically. Such an off-line programming approach based on 3D models drastically reduces programming and set-up time and allows the cell to keep production while the programmers realize new robot programs in an office environment [3], [4].

However, as noted in [5], off-line programming is not yet commonly and widely used in applications, as commercial off-line programming systems are very expensive, with prices even much higher than the robotic systems themselves. Furthermore, most of the available CAD-based developments are oriented towards (and limited to) universal serial six degrees of freedom (DOF) robots used for the automation of widespread industrial applications such as welding, spraying, machine tending, and assembly [6], [7], [8], [9]. Meanwhile, the alternative type of simulation software – the specialized 3D simulators – suffer from drawbacks associated with the lack of integrated modeling, design and evaluation tools, limited industrial applicability, and high licensing prices. In addition, these systems fail in providing an integrated solution that would allow engineers to perform design and simulation tasks simultaneously, without the need of using several different software products.

All these essential limitations raise questions concerning the effectiveness, availability and applicability of present-day robot modeling, motion simulation and off-line programming systems. A previous paper of the same author [10] proposes a solution to these issues by creating a SolidWorks-based simulation system. This paper is an extension of the previous work focusing on the implementation of a more efficient motion simulation approach, optimized kinematic modeling procedures, improved off-line programming functionality, simulation of multi-robot systems, and others. Despite being capable of providing motion planning on its own, the proposed tool relies on a more efficient external motion generation approach, directly interfacing to controllers or standalone simulators, thus contributing to the versatile and lightweight nature of the simulation system. In addition, it can be implemented in each SolidWorks license without adding cost and affecting SolidWorks performance (in contrast to the bulky architecture of most CAD-based simulators). Being integrated into SolidWorks, engineers have the capability to interactively design and verify their work via motion simulation within one and the same environment – a task especially effective when designing robots, which operate in tight mechanical constraints imposed by the tools and equipment. The developed modeling, motion simulation and off-line programming system has found a broad industrial application in semiconductor and flat panel industries.

II. SYSTEM ARCHITECTURE

The proposed implementation is based on the SolidWorks application programming interface (API) that provides complete freedom and flexibility in the process of developing custom tools and applications. Such an application composed as a VBA-based SolidWorks macro defines the basics of the proposed simulation software and shapes its specific system architecture.

A. SolidWorks API

The SolidWorks application programming interface (API) is a powerful tool, which incorporates hundreds of SolidWorks interfaces comprised of various methods and properties that can be addressed via VBA, VB.NET, C# or C++ programs. Based on the native SolidWorks object model (Fig. 1), the SolidWorks API provides direct access to the inherent 3D CAD software's functionality. This useful feature allows users to develop custom applications and take full advantage of the powerful SolidWorks capabilities for the execution of unique tasks.

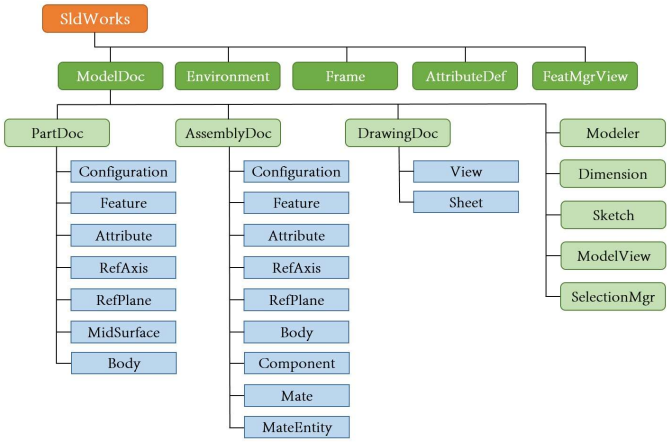

Figure 1. High-level SolidWorks object model hierarchy.

Employing the above concept, the proposed motion simulation and off-line programming system is explicitly developed as a SolidWorks API-compliant program written in Visual Basic for Applications, featuring all the necessary methods and interfaces required by the specific functionality of the simulation tool.

B. VBA Application (SolidWorks Macro)

The main goal of the VBA application for motion simulation and off-line programming is to actuate the existing 3D models developed for design and manufacturing purposes, as specific motion commands are being executed by the user. In other words, it must establish a connection between SolidWorks and the motion control software (MCS) of the manipulator being simulated. This key functionality is associated with setting up a motion data transferring procedure between the two platforms that is based on the following sequence – first, motion data is derived from a text file (generated by the MCS), containing the current values of all joints/logical axes of the manipulator being controlled. This is followed by a processing of the collected data, which is then sent to the already defined equations of motion (kinematic model) of the simulated device. Procedures such as component selections, motion equation calculations, and matrix transformations then take place, resulting in 3D model actuation that exactly corresponds to the motion cycle executed by the user. Other essential SolidWorks macro functionalities are associated with object-handling simulations, derivation of robot coordinates, 'motion-by-drag' execution, and implementation of an intuitive, macro-compliant graphical user interface (GUI). The structure of the developed VBA application for motion simulation and off-line programming based on the SolidWorks API is shown in Fig. 2.

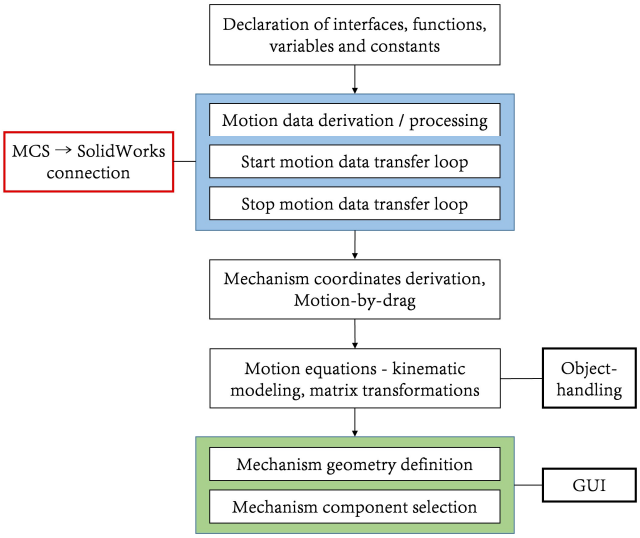

Figure 2. Structure of the VBA application based on the SolidWorks API.

III. SolidWorks API-compliant 3D Modeling

Before moving on to the implementation of the VBA macro application, an important preliminary step related to analysis and modification of the existing 3D assemblies has to be performed. The purpose is to develop new 'modified' assemblies that are compliant with the SolidWorks API methods and interfaces for component selection, matrix transformation and actuation of the 3D models, and thus are compatible with the overall structure of the simulation system. The modification procedure consists of decomposing the existing assemblies created for manufacturing purposes, and reassigning the coordinate frame of each moving component (link) of the simulated mechanisms. The educational serial manipulator Scorbot-ER IX has been used as an example throughout the following sections.

A. Decomposition of the Existing 3D Assemblies

The goal of the first step of the modification procedure is to solve issues related to the structure of the existing assemblies and the mates between individual links. Generally, these issues are associated with improper mate definitions, invalid structure of the assemblies, presence of 'fixed' components, etc. caused by inadequate modeling of the simulated mechanisms. It is important to note that designers cannot be blamed in this regard, simply because the described issues have little-to-no effect on manufacturing-related processes such as creation of drawings, execution of CAM, FEA-based simulations, etc. When it comes to motion simulation, however, it is critical for the components of the simulated mechanisms to be designed as individual parts, which are then used for the creation of new SolidWorks API-compliant assemblies. The described process requires analyzing the structure and mechanics of the manipulators that would later allow the user to modify the existing 3D assemblies (Fig. 3). Additional requirements are associated with removing all mates between the individual links, as their functionality is explicitly implemented in the VBA macro (via the kinematic modeling procedure). The elimination of internal components is another recommended modification step that helps improve the overall performance of the simulation system and deals with confidentiality issues.

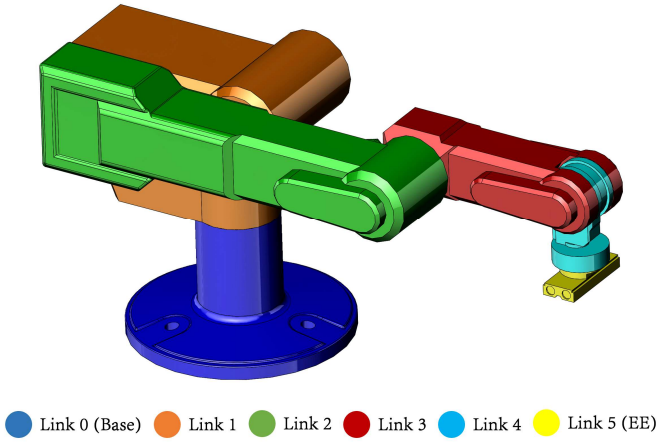

Figure 3. Analysis and decomposition of the Scorbot-ER IX robot assembly.

B. *Reassignment of Coordinate Frames*

The second modification step has a direct impact on the performance, functionality and accuracy of the simulation tool. Its goal is to make sure that the coordinate frame of each link is properly attached, in accordance with the manipulators' geometry and the corresponding kinematic modeling defined in the VBA macro. Just like the previous step, the reassignment of coordinate frames is always necessary due to the arbitrary frame attachments inherent to most of the available assemblies created for design and manufacturing purposes. Indeed, rarely would someone encounter 3D assemblies in industry where all frames are attached in a strict fashion, dictated by a specific rule. This is completely understandable since frame attachment has little to do with the creation of assembly drawings, exploded views, mounting instructions, and other similar tasks, typical for standard, design-oriented, static assemblies. In order to perform precise motion simulation and off-line programming tasks, however, a proper convention for link frame attachment must be adopted by the SolidWorks API-compliant assemblies (already created in the previous modification step). In the general case of simulating open-loop serial robots (Fig. 4), the coordinate frame attachment procedure is based on the common Denavit-Hartenberg (DH) convention, introduced in 1955 [11].

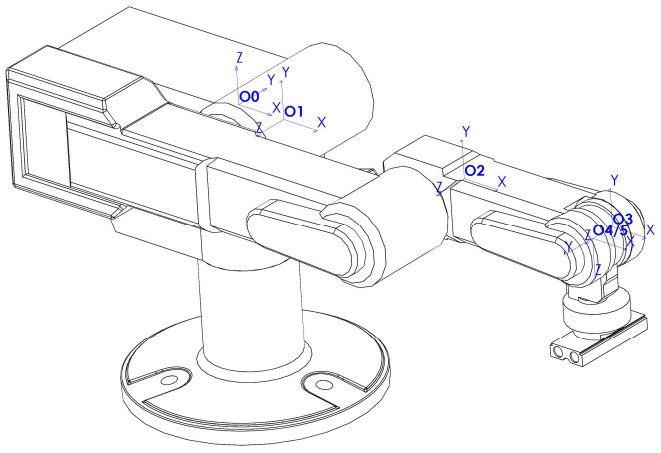

Figure 4. Scorbot-ER IX frame attachment based on the DH convention.

IV. Geometric and Kinematic Modeling

In order to actuate the SolidWorks API-compliant models, an essential geometric and kinematic modeling VBA procedure must be implemented. Its goal is providing a formal description of the geometry of the simulated mechanisms, defining their kinematic equations, performing matrix transformations, and actuating their links via specific SolidWorks API methods and interfaces. As already mentioned, the general case of simulating open-loop serial structures utilizes the efficient DH convention, which provides complete information in terms of the geometry of each link, as well as its connections with neighbor links, based on the definition of four quantities called DH parameters. Two of these parameters – a (link length) and α (link twist) – describe the link's geometry, while the other two – d (link offset) and θ (joint angle) – describe the link's connection to neighbor links. Given the above information, the relationship between the attached frames of each moving component can be easily defined using coordinate transformation and matrix calculation techniques, thus representing the pose of one link with respect to another as a function of the four parameters of the links, see (1).

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1) $$

Next, the link transformations can be multiplied together to find the single transformation that relates the frame of the robot's last link n (the end-effector) to the frame of the 0 link (the base). The derived transformation ${}^{0}_{n}T$ will be a function of all the robot's joint variables and, as a result, the further development of the robot's kinematic equations becomes straightforward. Adopting the described approach into the geometric and kinematic modeling VBA procedure requires the definition of multiple quantities, sub-procedures and functions representing constant/variable DH parameters, 3x1 position vectors, 3x3 rotation matrices, 4x4 transformation matrices, matrix transpose, matrix multiplication, etc. Some of the additional functionalities include selection of components, actuation of 3D models, and execution of object-handling tasks. It is important to remark that the SolidWorks API-based simulation system does not impose any limitations in terms of the utilized methods for geometric and kinematic modeling – thus, it supports the simulation of various robots and mechanisms, contributing to the overall flexibility and versatility of the developed system. Furthermore, the motion definition approach employed by the VBA application is based on a distinctive technique where the mechanisms' joint variables receive their values from a source, external to the SolidWorks API-based simulation system (robot controllers, standalone simulators or any other type of advanced software). In this manner, a real-time connection between the external source of motion planning (generally the manipulators' MCS) and the geometric and kinematic modeling procedure is established, contributing to the fast and lightweight nature of the proposed motion simulation system.

V. Connection to Motion Control Software

The primary goal of establishing a connection between the MCS of the simulated mechanisms and SolidWorks is the

implementation of a real-time communication between the two platforms, facilitating a motion data sharing process, essential for the functionality of the VBA application. The proposed concept offers a number of advantages, which distinguish the SolidWorks API-based simulation system from the currently available robot simulation and off-line programming software. For example, the structure of the developed system is simplified, as complex motion planning and control algorithms are not part of its architecture. On the other hand, the provided MSC-to-SolidWorks interface greatly expands the versatility and applicability of the simulation system, turning it into a universal solution for motion simulation and off-line programming of virtually any mechanism used in any field.

The implementation of the VBA procedure responsible for establishing the MCS-to-SolidWorks connection consists of reading an external text file containing the current values of all joints/logical axes of the manipulator being controlled. This is followed by a proper processing of the collected data, which is then sent to the manipulator's motion equations, already defined in the geometric and kinematic modeling procedure. The first step of the approach is related to analyzing and identifying the structure of the text file, corresponding to the exact mechanism being simulated. The purpose is providing the VBA application with information concerning the number of the available joint variables/logical axes as well as their specific arrangement. Once the system is able to identify and extract the particular value of each variable for a single time slice, a specific loop-based procedure takes place. Its purpose is to perform a smooth motion simulation by continuously reading the external motion data provided by the corresponding text file and transferring it to the mechanism's equations of motion. Therefore, a loop that facilitates the execution of the mentioned text file analysis, as well as the reading and data transferring tasks in an uninterrupted fashion, is developed. In this manner, the motion data generated by the simulated mechanism's MCS is continuously supplied to the corresponding kinematic equations, resulting in a smooth actuation of its 3D model. Some of the additional requirements associated with the described procedure include setting the correct loop frequency in accordance with the specific refresh rate of the motion control software, implementing a start/stop procedure that launches and terminates the execution of the file reading loop, and performing a unit conversion or any other type of motion data processing (if necessary).

VI. Additional Functionality and GUI

The implementation of a VBA procedure responsible for establishing the MCS-to-SolidWorks connection is the final step essential for the basic functionality of the developed motion simulation system. In order to enhance its overall performance and provide more advanced features associated with off-line programming and object-handling, however, three additional procedures are implemented in the VBA-based macro.

A. Motion-by-Drag

The so far described functionality of the SolidWorks API-based motion simulation system implies that all mates between the links of the utilized 3D models have to be removed in order for the system to operate in a proper way. In the standard case of performing motion simulations based on motion patterns and programmed cycles already existing in the manipulators' MCS ('Simulator' mode), this mate elimination has no negative effect on the application's functionality, as the simulation process is straightforward (MCS-to-SolidWorks). In the opposite case of performing SolidWorks-to-MCS off-line programming tasks, however, the modified 3D models and the corresponding VBA procedures turn inefficient. For this reason, the implementation of the so-called 'Drag' mode is considered. Its purpose is to allow the users to manually move the manipulator inside the SolidWorks graphics area, and be able to derive the values of its joint variables for any given configuration. The provided data can then be properly used inside the MCS, thus facilitating an efficient off-line programming approach.

The effective implementation of the above procedure relies mainly on the proper adaptation of the mechanisms' 3D models (the same ones utilized for standard motion simulations), which incorporates the addition of dimension and proximity sensors. These virtual tools provide useful information related to the displacement of the manipulators' joints, as well as their ranges and direction of motion. Specific VBA procedures comprised of various references, selections, conditional statements and calculations related to the supplied sensor information are then implemented, resulting in the creation of a robot coordinate derivation functionality. The final step of the 'motion-by-drag' development is associated with the definition of all connections (mates) between the mechanisms' links, as they are mandatory for the execution of the manual drag (the defined mates are disabled as soon as 'Simulator' mode is on, and vice versa).

B. Object-Handling

Another additional functionality is associated with the simulation of realistic object-handling operations, quite popular among many industrial robots. Its implementation is closely related to the geometric and kinematic modeling procedure, as the object-handling simulation concept is based on the same DH principle. When an object is picked by the manipulator, the coordinate frames of the handled object and the robot's end-effector start moving in an identical manner, as if the two bodies become a single component. As soon as the object is released, this relationship is broken, rendering the object stationary in space, with its frame arbitrarily positioned and orientated with respect to the robot base. The implementation of the described approach requires the availability of 3D models, with properly attached coordinate frames, of the objects being handled. Another key requirement is related to the introduction of specific variables in the text file that is generated by the robots' MCS. Their purpose is to indicate when a particular object is being picked or released, and by which end-effector (in case there are more than one). The final step of the implementation consists of defining the exact logic of the object-handling algorithm that is applicable to the specific manipulator being simulated.

C. Graphical User Interface (GUI)

In order to facilitate the overall functionality of the proposed modeling, motion simulation and off-line programming system, an intuitive graphical user interface (GUI) serving as an interface between the user and the procedures implemented in the VBA application, is developed (Fig. 5). Some of its most important

features are related to switching between ‘Simulation’ or ‘Drag’ modes, obtaining joint coordinates, and executing ‘motion-by-drag’ along specific robot joint coordinates/logical axes.

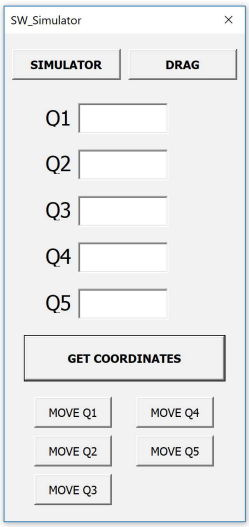

Figure 5. A GUI corrseponding to the Scorbot-ER IX SolidWorks macro.

The implementation of the GUI utilizes a number of controls such as toggle buttons, text boxes and labels, which are directly connected to the corresponding VBA procedures.

VII. Examples and Simulations

The applicability, versatility and efficiency of the proposed modeling, motion simulation and off-line programming system are validated by presenting examples and simulations from three different fields. The purpose of this section is to demonstrate that the developed system is relevant not only to universal robots (such as the Scorbot-ER IX example), but instead, to a wide range of applications, spanning from industrial manufacturing to human body mechanics.

A. Semiconductor Industry Automation

The first example of a successful utilization of the developed simulation system is associated with the field of semiconductor industry automation and wafer handling robotics. Manufacturing of semiconductor devices employs a specific type of high-performance robots, which are responsible for material handling inside complex processing machines and technology equipment. The successful implementation of these robotic systems, often comprised of multiple robots, a number of process stations, various peripheral equipment and other specialized components, all located in a highly compact and narrow environment, is often determined by the efficiency of the available simulation system and the execution of realistic motion simulations, precise off-line programming and in-depth customer layout analyses. In this regard, the proposed simulation system is specially adapted to the field of semiconductor industry. As a result, SolidWorks API-compliant 3D assemblies of wafer handling robots, their corresponding VBA macro files and GUIs are developed. These key simulation elements are then integrated with 3D models of the specific technology equipment provided by the customers, leading to the development of advanced and highly efficient 3D assemblies representing the whole automated system (Fig. 6).

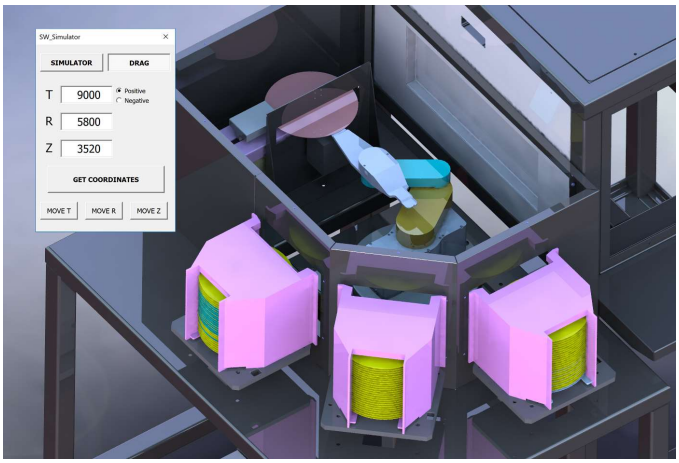

Figure 6. 3D models of a wafer handling robot and a specific customer tool.

The creation of such integrated assemblies, fully compliant with the functionality of the SolidWorks API-based simulation system, allows users to perform a variety of tasks associated with realistic motion simulations, off-line programming, station teaching, layout & reachability analysis, throughput estimation, visual collision detection, etc. Another important feature is the production of realistic, high-quality animations, which are useful for the creation of presentation and demo materials. All of the specified advanced techniques aid in the execution of various application, design and marketing tasks, saving time, effort and costs, and thus significantly enhancing productivity.

B. Overconstrained Parallel Mechanisms

The second example is related to the simulation of a special type of overconstrained parallel mechanisms, which has found broad industrial application in semiconductor manufacturing. Adapting the SolidWorks API-based simulation system to such mechanisms requires a different approach for geometric and kinematic modeling. This is explained by the fact that unlike serial robots, the DH convention is not compliant with parallel mechanisms, as their specific structure is described in a much more complex manner (Fig. 7). The required modification, in turn, affects the procedure of attaching coordinate frames to the mechanism’s components, as it is mandatory for the two procedures to be implemented in complete accordance, in order for the simulation system to function in a proper way.

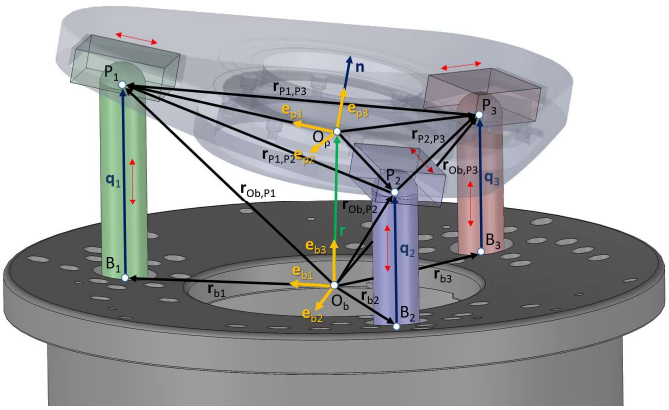

Figure 7. Modeling and simulation of a special type of parallel mechanisms.

Furthermore, the modified geometric and kinematic modeling approach requires additional vector/matrix operations, a special iterative procedure (based on the Newton-Raphson method), as well as multiple additional functions and algorithms, which are implemented as separate VBA procedures. Other than that, the SolidWorks macro development workflow remains unchanged, based on the algorithms, described in the previous chapters.

C. *Human Body Mechanics*

The final example of the proposed work is associated with the performance of a human leg motion simulation. The specific task focuses on the six DOF knee joint that allows the lower part of the leg (tibia and fibula) to be arbitrarily positioned and oriented with respect to its upper part (the femur) (Fig. 8).

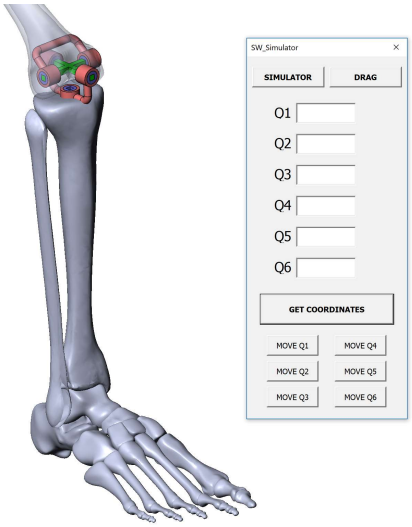

Figure 8. Motion simulation of a human leg (a six DOF knee joint).

The described functionality of the knee joint is facilitated by six specially designed components, which serve as the joint's links. Their specific arrangement and one DOF relative mobility allow the mechanism's final link (the tibia) to independently perform three translational and three rotational movements with respect to its zero link (the femur), as is the case with a real human leg. In order to integrate this functionality within the SolidWorks API-based simulation system, proper 3D modeling of the mentioned components, their corresponding coordinate frame attachments (based on the DH convention), and precise implementation of a formal motion description must take place. As a result, the user is able to 'animate' the human leg's 3D model, turning it into a beneficial instrument that can be utilized by various specialists in the field. This example serves as proof that the proposed SolidWorks API-based simulation system provides versatile motion simulation and off-line programming functionalities, with applicability ranging from industrial manufacturing and robotics to human body mechanics.

VIII. CONCLUSIONS

This paper contributes to the implementation of a general-purpose motion simulation and off-line programming system, oriented toward manipulators and mechanisms used in various fields. Based on the popular 3D CAD software SolidWorks and its powerful API functionality, the developed system can be considered as a universal and highly-efficient simulation tool relevant to a wide range of applications. The advanced capabilities of the proposed system are associated with the implementation of a SolidWorks macro written in VBA. Comprised of various procedures such as selection of moving components, geometric and kinematic modeling, connection to external motion control software, implementation of object-handling, development of an intuitive graphical user interface, etc., the advanced VBA application serves as a key component of the simulation system, facilitating its versatile functionality. The implemented concept is based on using the mechanisms' existing 3D models created for design and manufacturing purposes, thus eliminating the need for additional simulation software and saving time, effort, costs and resources. As a result, a unified tool that combines the useful features of specialized simulators and universal CAD software is provided, allowing engineers to simultaneously perform in-depth design, simulation, analysis and optimization tasks, based on a single platform. Future improvements are related to implementation of efficient algorithms for collision detection, improved GUI functionality, optimized structure of the VBA application, add-in programming (VB.NET or C#), automated procedures for assembly modifications and coordinate frame attachments, etc.

REFERENCES

[1] P. Neto, J. Pires, A. Moreira, "Robot Path Simulation: a Low Cost Solution Based on CAD," IEEE Conference on Robotics Automation and Mechatronics (RAM), Singapore, June 28–30, 2010.

[2] J. Rubinovitz, Handbook of Industrial Robotics. John Wiley & Sons, New York, Chapter 37, 1999.

[3] Y. Yong, M. Bonney, Handbook of Industrial Robotics. John Wiley & Sons, New York, Chapter 19, 1999.

[4] S. Mitsi, K. Bouzakis, G. Mansour, D. Sagris, G. Maliaris, "Off-line programming of an industrial robot for manufacturing," International Journal of Advanced Manufacturing Technology, August 2005.

[5] H. Wu, H. Deng, C. Yang, Y. Guan, H. Zhang, H. Li, "A Robot Off-line Programming System Based on SolidWorks," IEEE Conference on Robotics and Biomimetics, Zhuhai, China, December 6–9, 2015.

[6] K. Baizid, A. Meddahi, A. Yousnadj, S, Ćuković, and R. Chellali, "Industrial Robotics Platform for Simulation, Design, Planning and Optimization based on Off-line CAD Programming," MATEC Web of Conferences, January 2016.

[7] H. Chen, W. Sheng, N. Xi, M. Song, Y. Chen, "CAD-based automated robot trajectory planning for spray painting of free-form surfaces," Industrial Robot: An International Journal, 29(5), pp. 426–433, 2002.

[8] K. Baizid, R. Chellali, A. Yousnadj, A. Meddahi, B. Toufik, "Genetic Algorithms Based Method for Time Optimization in Robotized Site," IEEE/RSJ International Conference on Intelligent Robots and System (IROS), pp. 1359–1364, 2010.

[9] P. Neto, N. Mendes, R. Arajo, J.N. Pires, A.P. Moreira, "High-level robot programming based on CAD: Dealing with unpredictable environments," Industrial Robot: An International Journal, 39(3), pp. 294–303, 2012.

[10] N. Bratovanov, V. Zamanov, "Modeling and Simulation of Robots for Semiconductor Automation by Using SolidWorks API," Proceedings of Technical University of Sofia, vol. 66, issue 2, pp. 71–80, 2016.

[11] J. Denavit, R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," Journal of Applied Mechanics, vol. 1, pp. 215–221, June 1955.