Rapid Prototyping Toolchain for Humanoid Robotics Applications

Safwan Choudhury
Electrical and Computer Engineering
University of Waterloo. Waterloo, ON.
schoudhu@uwaterloo.ca

Derek Wight
Quanser Consulting Inc.
Markham, ON. Canada
derek.wight@quanser.com

Dana Kulić
Electrical and Computer Engineering
University of Waterloo. Waterloo, ON.
dkulic@uwaterloo.ca

***Abstract*—This paper introduces a rapid development toolchain for the design and dynamic simulation of robotic and/or mechatronic applications. The toolchain provides a fast and seamless workflow from developing a mechanical system in Computer Aided Design (CAD) software to automatically generating full dynamic simulations with real time 3D visualization. Subsequent design changes in CAD are reflected to the dynamic simulation blocks by simply updating the kinematic and dynamic parameters with minimal user input. The toolchain is demonstrated on the development of a 14 degree of freedom bipedal robot, validating its usefulness for designing complex robotic systems.**

***Index Terms*—Electromechanical Design, Robotics Toolchain, Dynamic Simulations.**

I. Introduction

The electromechanical design and development of multibody robotic systems is an iterative process, starting from a mechanical model in Computer Aided Design (CAD) software, transferring its parameters to a dynamic simulator for analysis and revising the design to improve the performance of the system. This process is repeated until the mechanical design achieves some desired goal. The iterative nature of the design and analysis process can often become time consuming and cumbersome. However, for high degree of freedom (DoF) multibody systems, the iterative approach is necessary as small changes in the mechanical design can have a significant impact on the overall dynamics of the system and the resulting system behaviour as well as implications for control design. Another popular approach is the use of optimization tools [1], [2] to determine the optimum design configuration. However, the resulting configuration may not be realizable with the available hardware components and must still be verified in simulation before hardware implementation.

A. Existing Solutions

There exists a wide variety of dynamic simulation environments with a range of capabilities [3], [4], [5], [6], [7], [8]. These environments provide a feature complete package, which combine an underlying dynamics engine with an interface for visualizing the simulations. The underlying computation engines obtain the complete equations of motion for the system described by its kinematic/dynamic parameters using techniques including, but not limited to, Lagrange multipliers [9], Kane's method [10] and port-based modeling [11]. The equations of motion are integrated to obtain the system state at each time step. Most simulators provide support for importing common Virtual Reality Markup Language (VRML) or Standard Tessellation Library (STL) files generated by CAD tools for visualization [3], [4]. However, only a few provide direct compatibility with CAD tools to import kinematic/dynamic parameters for simulation.

Open Architecture Humanoid Robotics Platform (OpenHRP) [5] is a commonly used dynamic simulation environment for humanoid robots that supports importing VRML files. However, kinematic and dynamic parameters of the robot are specified as plain text, making it cumbersome for rapid iterations.

MapleSim [12] is a multibody simulation package that provides limited functionality to communicate with CAD applications through its MapleSimConnector toolbox. This toolbox uses the underlying Maple engine with command-line access for retrieving the parameters of each link. However, the CAD application has to be actively running in the background and it is left to the user to generate Maple worksheets for batch importing of the parameters to update the link parameters in MapleSim through its API.

SimMechanics [13] provides mechanical import functionality to generate Extensible Markup Language (XML) files containing link parameters directly from CAD along with the corresponding STL files for visualization. However, there are limitations on how joint constraints are defined in CAD to successfully generate an equivalent model in Simulink. Another drawback of this approach is that the visualization generated during simulations significantly impacts the simulation speed.

B. Toolchain Development

In this paper, we develop a streamlined toolchain for rapid design iterations of robotic and mechatronic systems. Our approach consists of an add-in for the CAD package SolidWorks [14] that exports key physical parameters of the model for dynamic simulations. The exported file captures the kinematics (tree/chain hierarchy, frame transformations) and dynamics (mass/inertia properties) of the system from CAD assemblies and subassemblies.

In addition to exporting the physical parameters of the system, the exporter add-in also captures 3D meshes of each link subassembly and generates corresponding X3D files. The kinematic hierarchy of links is used to compose a comprehensive scene file to render real-time 3D animation using OpenGL. These exported files are used to generate full dynamic simulations in the Matlab/Simulink environment using SimMechanics [13] for multibody simulation and Quanser's QUARC toolbox [15] for 3D visualization.

The remainder of this paper is organized as follows. Section II outlines the process of exporting the CAD model from SolidWorks. Section III outlines the model generation/update capabilities of the toolchain which allow for rapid design iterations to be analyzed in simulation under the influence of a controller. Section IV provides a case study on the use of this toolchain in designing a 14 DOF lower body humanoid robot. Lastly, the conclusions and future work are presented in Section V.

II. CAD Export

We developed an add-in for CAD software package SolidWorks to export a multibody system for dynamic simulations in Simulink with realtime visualization. Consider a standard multibody system with n joints and $n+1$ links. The links are numbered from 0 (base) to n and each $joint_i$ connects $link_{i-1}$ to $link_i$.

The mechanical design of each link is represented by its own CAD assembly (or part) file. The coordinate system at the origin of this CAD file is treated as the local coordinate system xyz_i rigidly attached to the link i.

The top most CAD assembly (referred to as the *master assembly*) contains all $n+1$ links as subassembly files connected and constrained by the mechanical relationship which defines the behaviour of each joint n. For example, a revolute joint connecting two link subassemblies is defined by the appropriate constraints (i.e. concentric relationship).

Our add-in for SolidWorks uses this master assembly file as a starting point to export the multibody system for dynamic simulations. Once installed, a new tab is added directly in SolidWorks (as shown in Figure 1) presenting the user with several export options. The initial export process

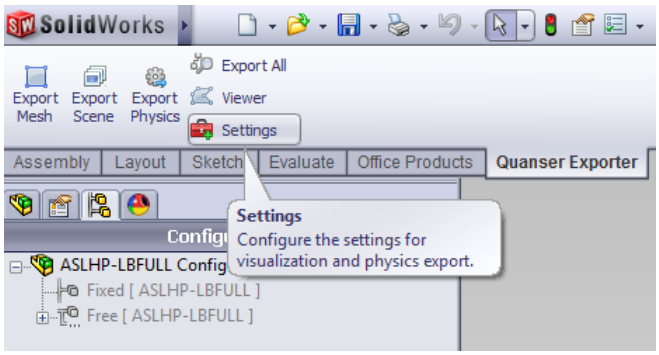

Fig. 1. Our exporter add-in for SolidWorks (named Quanser Exporter in the SolidWorks tab) to capture relevant physical and visualization data.

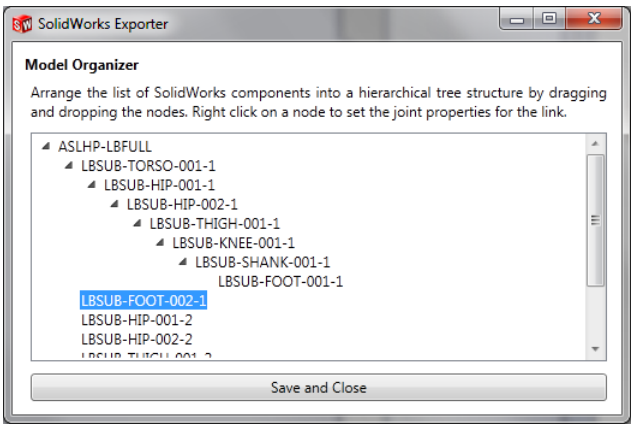

Fig. 2. The *Model Organizer* window used to define the kinematic structure of the system during initial export.

prompts the user with a flat list of all subassemblies in the current file via a *Model Organizer* interface (shown in Figure 2). Each subassembly is treated as a node that can be ordered (through a drag-and-drop interface) to form the desired kinematic tree/chain hierarchy.

A. Physics Export

Each link subassembly is parsed in the order defined in the *Model Organizer* window to extract key kinematic/dynamic parameters. All numerical values (i.e. distance between links) are extracted directly using CAD tools.

Assuming that the system is in the home configuration at the time of export, the relative frame transformations T_i^{i-1} between subsequent frames are extracted in accordance with the kinematic hierarchy defined in the *Model Organizer*. The absolute frame transformations T_i^0 are also computed with respect to the base link (if the base link is fixed in the master assembly file). For floating base systems (i.e. base link has 6 DOF), the absolute frame transformations T_i^W are taken with respect to the coordinate system of the master assembly file (world frame). The mass and dynamic properties are extracted with internal CAD tools in the local coordinate system of the link subassembly file. The mass, distance to the center of mass (COM) and inertia tensor of the link (at the COM position) are extracted.

Our add-in captures these key parameters and generates a structured XML file. A XML node is generated for each link containing the extracted kinematic/dynamic parameters (as shown in Figure 3). Furthermore, each XML link node is organized in accordance with the kinematic hierarchy of the system. As a result, our add-in captures the entire physical description of the multibody system in a portable, language independent file.

B. Mesh/Scene Export

In addition to exporting the physical description of the model, our exporter add-in also generates files for visualization. These files can be used in conjunction with dynamic simulations to provide the user with visual feedback of

the system under control. Some dynamic simulators (SimMechanics, MapleSim) currently allow users to specify a VRML file for each link of the multibody system. While SolidWorks currently has some support to export VRML files for each link subassembly individually, there are no options to generate the 3D meshes in X3D (successor to VRML).

We have developed a process for automated batch generation of full 3D meshes in the X3D format. Once the export process has been initiated, the meshes are extracted for each link subassembly in the master assembly file directly from the CAD layout. Each mesh is aligned with the local coordinate system of the link such that there is a direct mapping between the visualization and the physical model.

The overall mesh generation process is optimized for complex multibody systems. A typical CAD subassembly representing each link may contain hundreds of smaller CAD part/assembly files. During the mesh generation process our add-in creates a new flattened version of each link subassembly and discards any visual details not visible on the surface. The surfaces of this flattened model are tessellated to generate the mesh file for each link. As a result, the mesh files are light weight and this helps speed up the rendering process. At the time of writing, our toolset does not yet provide the user with the ability to specify the level of detail for tessellation.

The output mesh files are stored in the same directory as the XML file containing the physical description of the system. Furthermore, each link XML node is also updated with a complete path to its corresponding mesh file alongside the kinematic/dynamic parameters.

Our add-in also generates a *scene* file from the CAD master assembly. This is a parallel XML file which contains a complete visual description of the system from CAD. The scene file contains a list of all links alongside their corresponding mesh files. These meshes are then organized in accordance with the kinematic tree/chain hierarchy provided by the user in the *Model Organizer*.

```
- <MassProperties>
    <Mass>0.534887028663829</Mass>
  - <COM>
      <X>-3.9E-07</X>
      <Y>0.04741578</Y>
      <Z>-0.00980513</Z>
    </COM>
  - <Inertia Format="Row Major" Columns="3" Rows="3">
      <Ixx>0.001186264679004351</Ixx>
      <Iyy>0.00070094139771874721</Iyy>
      <Izz>0.0014829006492190239</Izz>
      <Ixy>-7.9742722453230943E-05</Ixy>
      <Izx>3.6541669161088704E-05</Izx>
      <Iyz>-0.00014857992698925752</Iyz>
    </Inertia>
    <Volume>0.00016279636319438745</Volume>
    <Area>0.085515263680874087</Area>
  </MassProperties>
- <MaterialProperties>
  - <Colour>
      <Red>0.753</Red>
      <Green>0.753</Green>
      <Blue>0.753</Blue>
    </Colour>
```

Fig. 3. Example of kinematic and dynamic parameter extraction straight from CAD model stored in the exported XML file.

As a result, the scene file recreates the visual layout of the complete multibody system from the CAD master assembly file. This file encapsulates the full multibody system in a format which is compatible with the 3D visualization blocks that are packaged with the QUARC toolbox. In addition to the model itself, additional supplementary mesh files (i.e. ground plane) are imported into the scene for visualizing the environment during simulation.

C. CAD Update

With minimal user input, our add-in captures a complete physical and visual description of the system in its *current state*. However, the main benefits of our approach become apparent after the initial export. Once the user defines the structure of the system (kinematic hierarchy and joint definitions) through the *Model Organizer*, it is stored in memory for later use. Subsequent updates to the CAD model can be exported with a single click if the overall structure of the links and joint definitions do not change.

This allows the user to export a model for simulation from CAD, analyze the behaviour of the system under control, tweak the mechanical design and immediately re-export the revised model for simulation. For example, increasing the length of a link may alter its dynamic properties while the overall kinematic structure remains the same. The revised CAD model with increased link length can be exported with a single click. The export process simply recalls the structure of the system and regenerates the XML files with updated kinematic/dynamic properties. Our add-in also allows the user to regenerate only the mesh file for the updated link to reflect the changes in CAD. This process makes it fast and easy for rapid iterations during the design phase.

III. Model Generation

The XML files exported by our add-in encapsulate the relevant information which describes the CAD model into structured and portable files. One of the key advantages of our approach is that the information in these files can easily be parsed to generate an equivalent model in most dynamic simulators. We have developed a model generation counterpart for the Matlab/Simulink environment which uses SimMechanics for multibody dynamic simulation and Quanser's QUARC toolbox for 3D visualization. This provides a semi-automated toolchain for designing robotics and/or mechatronic systems.

Our model generation approach provides several Matlab functions, scripts and libraries to parse the files exported by our SolidWorks add-in and generate the equivalent mechanical system in Simulink. The generated Simulink model contains (a) SimMechanics blocks with the link kinematic and dynamic parameters from CAD and (b) visualization blocks from QUARC libraries with the link meshes and scene file. The generated physics and visualization counterpart is prepopulated with CAD data and connected according to the kinematic hierarchy of the system defined in the *Model Organizer*.

The model generation process is initiated by calling a function in the Matlab terminal inside the CAD export folder. By default, our approach generates a physical subsystem (i.e. the plant) for forward dynamics simulation, whose output drives a generated visualization subsystem. The user can also generate an inverse dynamics model through the command line.

A. Physical Model

Our physical model generation process parses the XML output files in the export folder to create a CAD-equivalent system in Simulink. In order to streamline the process, we have developed a library (shown in Figure 4) of masked link subsystems representing common link configurations. Each of these subsystems contains a combination of SimMechanics joint, body, actuator and sensor blocks to represent a combination of $joint_i$ and $link_i$ (shown in Figure 5).

The input to each link subsystem (**CS1**) is a connection from $link_{i-1}$ and the driving signal for the joint. The output of the link subsystem is its local coordinate system (**CS2**) and the joint sensor signals. The joint actuator/sensor signals are set depending on the type of dynamic simulation. By default the model generation process configures the physical model in forward dynamics mode so the joint actuator port is connected to the force/torque command input for $joint_i$ and joint angle/displacement is measured at the output. Alternatively, for the inverse dynamics simulation, the joint acceleration is set as the input, while the joint force/torque is the output.

The output coordinate system of the $joint_i$ block is connected to a SimMechanics body block representing $link_i$. The masked parameters are preconfigured to set the local dynamic properties (i.e. COM position, inertia tensor) and relative frame transformation appropriately.

Each link subassembly from CAD is recreated with a link subsystem from our library depending on the link type. The masked parameters are populated with the kinematic and dynamic parameters from the corresponding XML link

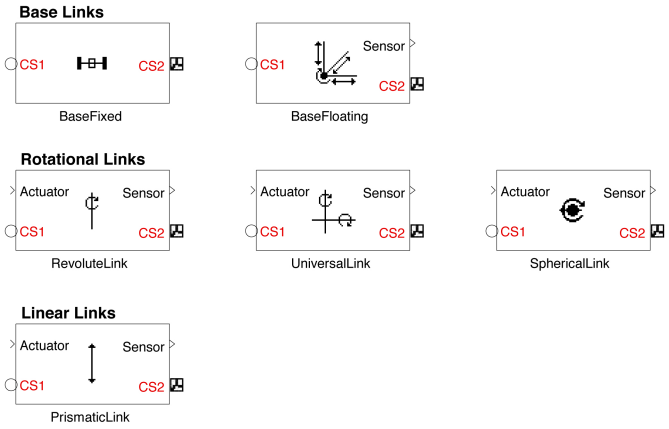

Fig. 4. Our standard link subsystems for physical model generation with prepopulated kinematic and dynamic parameters from CAD.

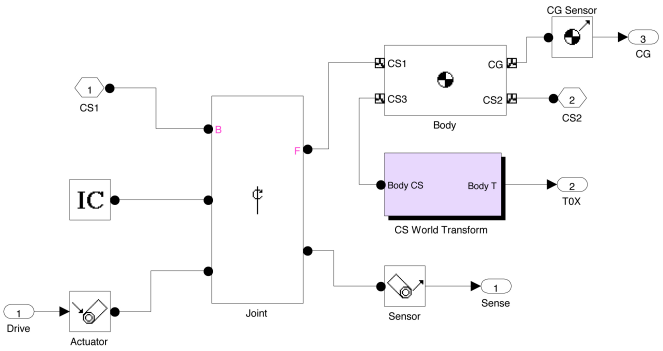

Fig. 5. SimMechanics blocks used to compose each CAD-equivalent link subassembly in Simulink

node. The overall hierarchy of links is parsed and each link subsystem is connected accordingly. Our model generation process also automatically handles the signal routing from the input/output ports of the physical model. In the default forward dynamics configuration, the $n \times 1$ torque/force vector is connected to all joint actuator blocks and the output signals from the sensors are also routed accordingly. For the inverse dynamics case, the vector of joint positions, velocities and accelerations are connected to the joint actuator blocks and the sensors are preconfigured to output the resulting torque/force vector.

In addition to the link subsystem, a ground block representing the coordinate frame of the CAD master assembly is connected to the base $link_0$. The entire process of creating an equivalent model for dynamic simulation is automated. The user simply calls the appropriate function and the generated physical model subsystem is placed in a new (or existing) Simulink diagram.

B. Visualization Model

Our visualization model generation process generates a single subsystem containing blocks to initialize and drive the scene file generated from CAD. The mesh for each link in the kinematic chain is driven by the output of the generated physical model (forward dynamics subsystem) so that there is a 1:1 mapping between the plant and what is being rendered by the visualization. When a simulation is running, an external 3D viewer application (part of the QUARC toolbox) is used to render the scene in realtime using OpenGL (CAD-equivalent visualization scene file shown in Figure 6).

Our approach allows the user to receive immediate visual feedback of the CAD model under the influence of control. This also allows the user to make changes to the mechanical design from visual observations (i.e. quick changes to improve joint limits).

C. Model Update

Our export add-in makes it possible to make changes to the mechanical design in CAD and generate the new kinematic and dynamic parameters immediately. After the initial model generation from CAD data, these new changes can be

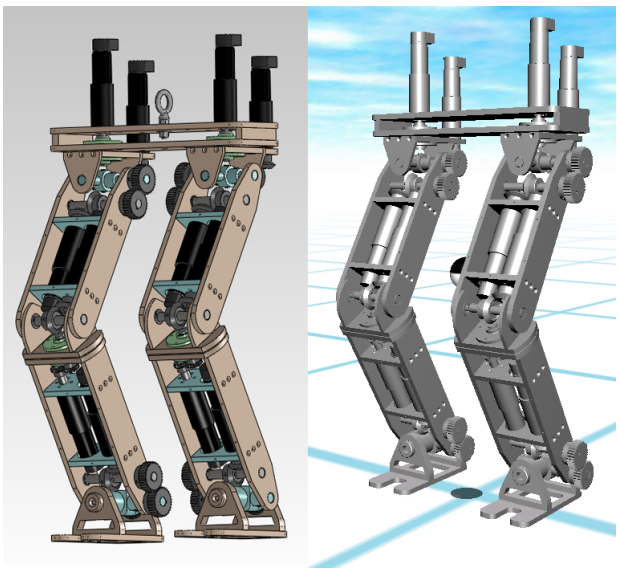

Fig. 6. CAD master assembly shown on the left is used to automatically generate the meshes and scene file to recreate the visualization shown on the right.

reflected back into the dynamic simulation by simply calling the update function. The update functionality also allows the user to specify the path to the previously generated physical model. The masked parameters on each link subsystem inside this model are simply updated with the new changes while the signal routing and everything else is left intact. The updated mesh files are loaded by the QUARC 3D viewer during the next simulation run.

This streamlines the iterative design process by allowing a user to export a system from our add-in and generate the CAD-equivalent physical/visual model in Simulink. The behaviour of the system can then be analyzed in dynamic simulations to revise the CAD design. Once the new changes are applied, the revised parameters are easily exported back into dynamic simulations for further analysis.

IV. Case Study

This toolchain was used to design a 14 DOF lower body humanoid robot[1] at the University of Waterloo. This is a particularly challenging design problem due to the complexity of the mechanical system and the control challenge of maintaining balance. The toolchain was used to quickly analyze the effects of design revisions (i.e. compare motor positioning) in simulation prior to manufacturing.

A. Dynamic Simulations

The proposed toolchain was used during the design phase to estimate the torques at each joints for appropriate motor sizing. Changes to the mechanical design such as motor positioning and material of the links can significantly alter the torques required at some joints. In these situations it is useful to make incremental changes to analyze their impact on the system performance and immediately use this knowledge to tweak the design.

A common motion for humanoid robots is the bending of the knee and hip joints while a foot is swinging over during the gait cycle. During this phase, the motors at the hip joint must carry the overall weight of the leg below it. The choice of actuators on the leg plays a crucial role in its overall weight and as a result, also plays an important role in the torques required at these joints. Similarly, when the swing foot comes into contact with the ground, the knee joint absorbs a significant amount of torque so the motors must be sized appropriately.

By using our approach, the user can maintain multiple configurations of the same mechanical design in CAD and export them for direct comparison with the dynamic simulator. Since the model is already defined, exporting each configuration takes less than a minute. The parameters of these configuration files can be used along with our model update feature to simulate each configuration and compare the results. In the case of bending the knee and hip joints, there may be several choices of motors which alter the system performance (due to large masses further down the leg). The torque profiles of each joint can be analyzed in the simulation (as shown in Figure 7) to select the most desirable configuration.

B. Visualization

Additional meshes can also be added to the scene to serve as visual aids. For example, coordinate systems can be visualized during simulation by attaching arrow meshes to each link. Alternatively, common Simulink blocks can be used to determine if a particular joint is out of its limits and use the resulting signal to drive the colour of the mesh file (i.e. turn a link *red* if the joint is out of its limits).

A particularly useful visual aid for floating base multibody systems is the ability to see where the COM of the system is during simulation. An additional mesh can be added to the scene driven by the COM calculation from forward

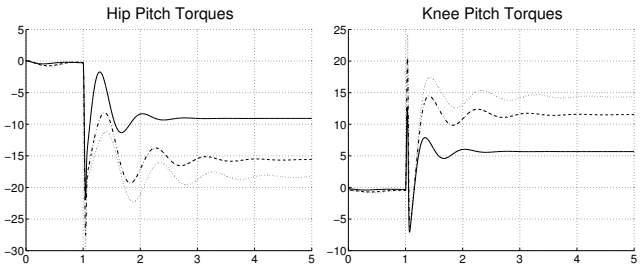

Fig. 7. Hip and knee joint torque requirements while a leg is raised for different sets of motors at the joints. The dotted and solid lines represent the heaviest and lightest motors, respectively. The dashed line represents the motor set with medium weight (between the heavy and light motors). Our toolchain allows for fast incremental changes to revise the mechanical design and compare the torque requirements.

[1]A video demonstration of our toolchain is available online at: https://ece.uwaterloo.ca/~dkulic/videos/Humanoids2012-480p.mp4

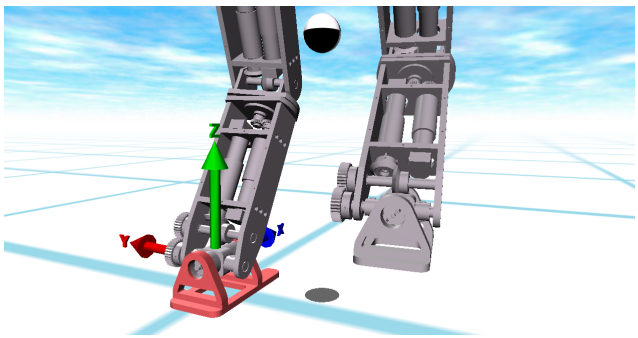

Fig. 8. Realtime visualization during simulations allows the user to get immediate visual feedback on important information like COM position.

kinematics at each time step. The use of this visual aid is shown in Figure 8.

By having the visualization rendered in an external application, our approach is capable of running simulations much faster than the built-in 3D viewing capabilities of SimMechanics. This increase in runtime speed is especially useful during the design phase where rapid iterations are common. The runtime for a 60s dynamic simulation[2] at 1KHz *with* visualization is compared in Table I. If the viewer is also being used to render the 3D mesh for each link through supplied VRML files, it takes much longer. In contrast, our approach takes a fraction of the time for the exact same system to be simulated. The simultaneous visualization using the QUARC visualization toolset has little or no impact on the simulation speed. In fact, running the same dynamic simulation without the QUARC visualization toolset only reduces the overall runtime slightly.

TABLE I
60S DYNAMIC SIMULATION RUNTIME

Configuration	**Runtime** (s)
No Visualization	47.5
SimMechanics Visualization	367
Proposed Toolset Visualization	49

V. CONCLUSIONS

A streamlined toolchain was presented to enable rapid prototype development during the design and development phase of complex multibody systems. The toolchain exports a physical and visual representation of the CAD model which can be used to automatically generate dynamic simulations. After the initial model generation, subsequent exports simply update the physical model parameters and/or visualization blocks allowing for faster design iterations. The toolchain was used extensively during the design and development of a 14 DOF biped, demonstrating its usefulness in designing a physical robot. Commercial licensing for the toolchain will be available in the future with QUARC [15].

[2]The simulations were executed on a standard PC available at the time of writing (2.4GHz Intel Core 2 Duo, 4 GB RAM).

While our CAD export add-in and model generation approach is extremely useful in its current form, we hope to make a few changes to the improve the toolchain process. The current process requires the user to explicitly define the link hierarchy and joint descriptions through the *Model Organizer*. In future versions we hope automatically infer the relationships by analyzing the mechanical constraints from the CAD model. One of the key requirements of our approach is that the local coordinate system be defined at the origin of each subsystem link. While this is possible for designing new systems using the toolchain, we hope to add support for user defined local coordinate systems through reference geometry entities in CAD. Lastly, future versions of the toolchain will provide more granular control (i.e. controlling the level of detail during mesh export) and the ability to export colour properties and/or textures directly from CAD.

ACKNOWLEDGMENT

This research was supported by the Natural Sciences and Engineering Research Council of Canada and Quanser Inc.

REFERENCES

[1] C. Paul and J. Bongard, "The road less travelled: morphology in the optimization of biped robot locomotion," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, 2001, pp. 226 –232 vol.1.

[2] D. Wollherr, M. Hardt, M. Buss, and O. von Stryk, "Actuator selection and hardware realization of a small and fast-moving, autonomous humanoid robot," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, 2002, pp. 2491 – 2496 vol.3.

[3] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[4] O. Michel, "Webots: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 40–43, 2004.

[5] F. Kanehiro, H. Hirukawa, and S. Kajita, "OpenHRP: Open Architecture Humanoid Robotics Platform," *The International Journal of Robotics Research*, vol. 23, no. 2, pp. 155–165, 2004.

[6] R. Ponticelli and M. Armada, "Vrsilo2: dynamic simulation system for the biped robot silo2," in *Proceedings of the 9th International Conference on Climbing and Walking Robots*, 2006.

[7] T. Reichenbach, "A dynamic simulator for humanoid robots," *Artificial Life and Robotics*, vol. 13, no. 2, pp. 561–565, 2009.

[8] G. A. Medrano-Cerda, H. Dallali, M. Brown, N. G. Tsagarakis, and D. G. Caldwell, "Modelling and simulation of the locomotion of humanoid robots," in *Proceedings of the UK Automatic Control Conference*, 2010.

[9] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 137–146.

[10] D. Rosenthal and M. Sherman, "High performance multibody simulations via symbolic equation manipulation and Kane's method," *Journal of the Astronautical Sciences*, vol. 34, pp. 223–239, 1986.

[11] C. Paredis, A. Diaz-Calderon, R. Sinha, and P. Khosla, "Composable models for simulation-based design," *Engineering with Computers*, vol. 17, pp. 112–128, 2001, 10.1007/PL00007197.

[12] MapleSoft Inc. MapleSim: High-Performance Physical Modeling. [Online]. Available: http://www.maplesoft.com/products/maplesim/

[13] MathWorks Inc. SimMechanics: Multibody Simulations for MATLAB and Simulink. [Online]. Available: http://www.mathworks.com/products/simmechanics/

[14] Dassault Systèmes. SolidWorks: 3D CAD Design Software. [Online]. Available: http://www.solidworks.com/

[15] Quanser Inc. QuaRC: Control Simulation Software. [Online]. Available: http://www.quanser.com/english/html/solutions/fs_soln_software_QuaRC.html