CALIPER: A Universal Robot Simulation Framework for Tendon-Driven Robots

Steffen Wittmeier, Michael Jäntsch, Konstantinos Dalamagkidis, Markus Rickert,
Hugo Gravato Marques and Alois Knoll

***Abstract*—The development of increasingly complex robots in recent years has been characterized by an extensive use of physics-based simulations for controller design and optimization. Today, a variety of open-source and commercial simulators exist for this purpose for mobile and industrial robots. However, existing simulation engines still lack support for the emerging class of tendon-driven robots.**

In this paper, an innovative simulation framework for the simulation of tendon-driven robots is presented. It consists of a generic physics simulator capable of utilizing CAD robot models and a set of additional tools for simulation control, data acquisition and system investigation. The framework software architecture has been designed using component-based development principles to facilitate the framework extension and customization. Furthermore, for inter-component communication, the operating-system and programming language independent Common Object Request Broker Architecture (CORBA) [1] has been used which simplifies the integration of the framework into existing software environments.

I. INTRODUCTION

Due to their versatile application capabilities, computer simulations have become an indispensable tool for engineers and researchers during the last decades. Typical areas of application are the modeling of natural phenomena [2], of economic processes [3] and of products under development for design optimization [4].

In robotics, as well, simulations have become an important tool that is mainly used offline for controller design and optimization as cheap and safe substitutes of the real hardware [5]. Today, a huge variety of physics-based open-source and commercial robot simulation platforms exist for that purpose, such as OpenRAVE [6], Microsoft Robotics Developer Studio [7], Gazebo [8] or Webots [9]. However, these simulators are particularly designed for either mobile or industrial robots and, despite their high flexibility in this area, are difficult to adapt to the specific requirements of tendon-driven robots. Hence, specialized simulation engines have emerged that can cope with the simulation of the dynamics of this class of robots. The most popular engine currently available is OpenSim [13]. However, it has been developed with focus on biomechanics which makes it again difficult to adapt for simulating tendon-driven manipulators or humanoids. Furthermore, OpenSim lacks the ability of importing common CAD data formats such as COLLADA [14] or VRML [15]—a feature provided by most simulators developed for traditional robotics. This, however, can make the model development process a highly time-consuming and error-prone task as the CAD model defined during robot engineering cannot directly be re-used, requiring the simulation model to be re-implemented for the simulation engine by hand. In addition, there has been a recent trend in robotics of employing physics-based simulations *online* as an internal model for control [16] or functional imagination [17]. These applications, however, impose very specific requirements on the simulation engine such as a real-time interface for updating and querying model parameters during simulation that are not supported by all simulators available.

Fig. 1. Prototype of the *anthropomimetic* [10] robot ECCE-I [11] developed within the EU-funded project ECCEROBOT [12]. The skeleton is hand-crafted using polymorph—a caprolactone polymer—which can easily be molded. The human muscles are imitated by elastic, tendon-driven actuators comprising a DC motor and gearbox in series with a kite line and shock cord as tendon.

In this paper, a universal and customizable robot simulation framework for tendon-driven robots is presented. It has been developed within the EU-funded project Embodied Cognition In A Compliantly Engineered Robot (ECCEROBOT) [11, 12] and has been evaluated by simulating the challenging dynamics of the highly complex, multi degree-of-freedom anthropomimetic [10] robot torso developed in this project [11, 18] (see Fig. 1). The software framework comprises a physics-based simulator capable of importing and exporting CAD models which is described in Section II as well as a set of additional graphical-user-interface (GUI) tools for advanced user-simulator interaction which are introduced in Section III. The software architecture of the framework is presented in

S. Wittmeier, M. Jäntsch, K. Dalamagkidis, M. Rickert, and A. Knoll are with the Chair of Robotics and Embedded Systems, Faculty of Informatics, Technische Universität München, Munich, Germany; Correspondence should be addressed to S.Wittmeier (wittmeis@in.tum.de)

H.G. Marques is with the Artificial Intelligence Laboratory, University of Zurich, Zurich, Switzerland

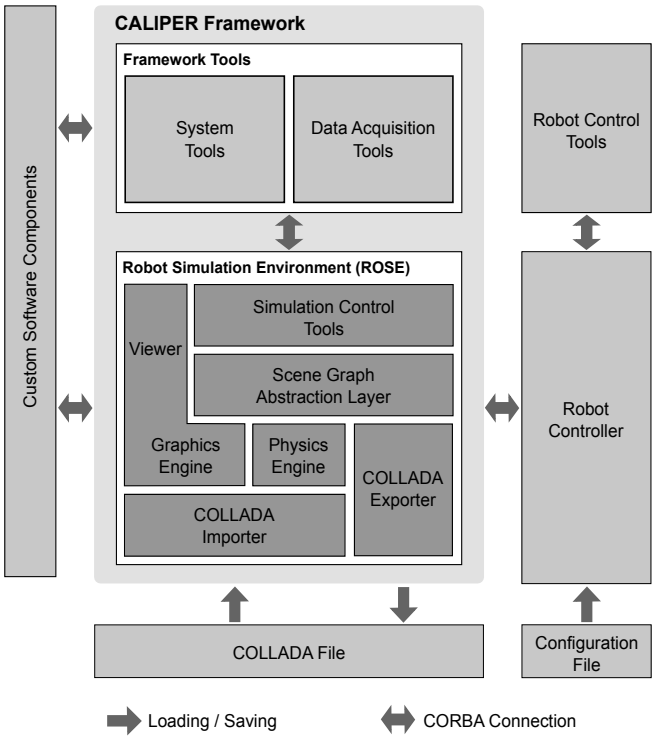

Fig. 2. CALIPER Framework Overview. The framework consists of the robot simulation environment (ROSE) which is capable of importing and exporting COLLADA [14] robot models and of a set of framework tools for user-simulator interaction. Robot-specific components such as the COLLADA model file, the Robot Controller, the Configuration File and the Robot Control Tools have been implemented for the ECCEROBOT platform to evaluate the provided ROSE interfaces but are not part of the framework (a demo video presenting the integration of CALIPER for the ECCEROBOT project can be found on [12]). The framework can easily be extended by custom software components using programming language and platform-independent CORBA [1] interfaces.

Section IV followed by conclusions and future work prospects in Section V.

II. ROBOT SIMULATION ENVIRONMENT

The core of the presented framework is the robot simulation environment (ROSE, see Fig. 2). It comprises (i) a physics engine for computing the dynamics of the simulated world, (ii) a graphics engine and a viewer for rendering and displaying the simulated scene, (iii) an abstraction layer for engine-independent interfacing with higher-layer software components, (iv) a COLLADA importer and (v) exporter for importing and exporting CAD robot models, respectively, as well as (vi) simulation control tools for basic user-simulator interaction. The ROSE receives inputs either from the user through the GUI of the provided framework tools (see Section III) or from the robot controller component. It can easily be extended by attaching custom software components using the provided CORBA interfaces (see Section IV).

A. Physics Engine

Important for physics-based simulators and particularly when used *online* for robot control are the accuracy and the performance of the simulation. The two benchmarks depend on various factors such as the numerical integration step-size or the level of detail of the simulation model. The most profound factor, however, is the selection of the physics engine since the latter computes the dynamics of the simulated bodies, performs collision detection and solves the constraint equations required for joint simulation. Nowadays, the most commonly used physics engines are [20]: (i) Open Dynamics Engine (ODE) [21], (ii) Bullet Physics [22], (iii) Havok Physics [23] and (iv) NVIDIA PhysX [24]. All these engines offer a sufficient simulation accuracy for the simulation of tendon-driven robots and a rich set of functionality (see Table I and [19]). PhysX, however, provides the highest performance of the four by exploiting the multi-parallel architectures of today's graphic processing units (GPUs). But, similar to Havok, its closed-source license restricts the development of the custom engine extensions that are required to simulate the particular dynamics of tendon-driven actuators. ODE, on the other hand, is open-source but lacks support for multithreading, hardware acceleration and, most importantly, soft body dynamics—a feature that is provided by the selected engine Bullet Physics. However, soft body dynamics are particularly important for an accurate simulation of tendon-driven actuators as they enable the simulation of the dynamics emerging from the collision of the tendon with the mechanical structure. For the presented framework custom Bullet Physics extensions have been implemented to simulate the dynamics of tendon-driven acutators. Currently supported are (i) DC-motor based, compliant tendon-driven actuators comprising linear-spring dampers [18], (ii) passive, linear spring-damper based *ligaments* [18] and (iii) Hill muscles [25]. The actuator type as well as the actuator model parameters can be conveniently configured through custom Collada extensions (see Section II-E). Moreover, the flexible class hierarchy chosen for the actuators makes it also possible to easily incorporate tendon-driven actuators based on different actuation concepts, such as pneumatic or hydraulic actuators and with altered characteristics, e.g. non-linear springs.

B. Graphics Engine

The second core component of the ROSE is the graphics engine which provides visual feedback to the user by rendering the current state of the simulated scene. Here, even more potential open-source and closed-source candidates exist, such as the Unreal Engine [26] developed by Epic, the CryENGINE [27] developed by the German company Crytek or OGRE [28], a very popular open-source engine. However, for CALIPER Coin3D [29], an OpenGL-based, open-source clone of the SGI Open Inventor 3D graphics application programming interface (API) is used. It is cross-platform compatible using Qt [30] and comes already with a variety of pre-defined viewers for basic user-interaction with the simulated scene.

C. Scene-Graph Abstraction Layer

The graphics and physics engine APIs are highly engine-specific and might even change with new releases. Thus,

TABLE I
PHYSICS ENGINE COMPARISON (FOR A DETAILED COMPARISON INCLUDING PERFORMANCE MEASURES SEE [19])

	PhysX	**Havok Physics**	**Bullet Physics**	**ODE**
Developer	NVIDIA	Havok	Community	Community
Open-Source	–	–	✓	✓
Programming Language	C++	C++	C++, Java	C++
Operating System	Windows / Linux	Windows	Windows / Linux / MacOS	Windows / Linux / MacOS
GPU acceleration	✓	–	experimental	–
Multithreading	✓	✓	✓	–
Rigid/Soft Body Dynamics	✓/✓	✓/✓	✓/✓	✓/–
Constraint Dynamics	✓	✓	✓	✓
Collision Detection	✓	✓	✓	✓

exchanging one of the engines or simply updating to a new release might result in myriads of changes in the respective software modules. The classical computer science approach to tackle such problems is to introduce an abstraction layer that hides the implementation specific details by providing a generic interface for the higher-layer software components. One example for such an abstraction layer for physics-based simulators is the Physics Abstraction Layer (PAL) [31] which provides a unified interface for a variety of physics engines including ODE, Bullet Physics, Havok Physics and PhysX. However, the PAL interfaces are limited to physical simulation entities and do not provide graphic-engine abstractions. Hence, a custom scene-graph abstraction layer (SGAL) has been developed for CALIPER that decomposes the simulation scene into physical and graphical *models*, *bodies*, *shapes* and *constraints*, each of which is represented by an interface class for standardized, engine-independent access (see Fig. 3). The SGAL physical and graphical scene-graph data structures are created dynamically at runtime by the Collada importer (see Section II-E) and are used during simulation for physical-to-graphical scene synchronization: at each simulation step the physics engine recomputes the transformation matrices of the simulated bodies. Subsequently, these updated matrices need to be set within the graphics engine to visualize the new positions and orientations to the user. This has been accomplished by implementing an observer pattern [32] in which the viewer registers to the simulator model and the latter notifies the viewer of any updated transformation matrix by calling an appropriate update method of the viewer-interface. With this scheme it was possible to significantly decrease the amount of data that has to be transmitted between the simulator model and the viewer at each simulation step. This in turn increases the simulation performance when scenes with a large number of bodies are simulated and reduces the network load when the simulator is distributed on a computer cluster—which is possible due to the component-based software design relying on the CORBA-based middleware OpenRTM [33] (see Section IV).

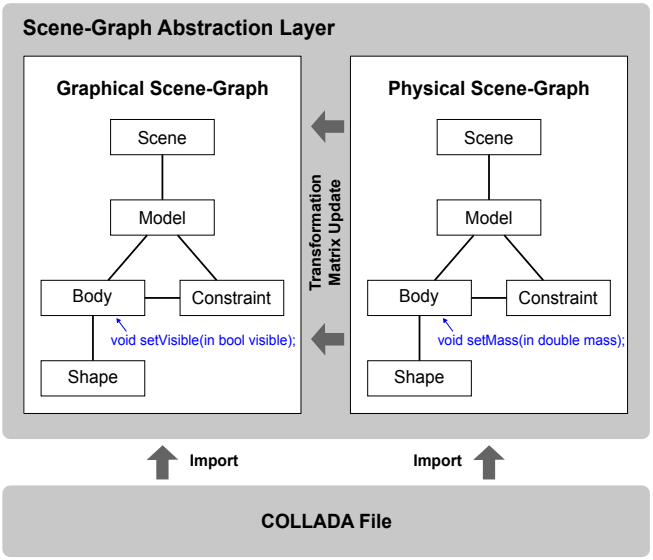

Fig. 3. Scene-Graph Abstraction Layer (SGAL). In the developed abstraction layer the physical and graphical simulation scenes are represented by identical scene-graphs, both consisting of *models*, *bodies*, *shapes*, and *constraints*. These scene-graph data structures are created dynamically by importing the COLLADA scene file into the physics and graphics engine, respectively. The abstraction layer is used for engine-independent interfacing with higher software layers and efficient physical-to-graphical scene synchronization.

D. Simulation Control Tools

Basic simulation control tools have been developed for the ROSE using the Qt toolkit. These tools provide controls for starting, stepping and stopping the simulation as well as for loading a simulation scene and modifying general simulation-specific parameters such as the maximum frames-per-second or the numerical integration step-size of the physics engine (see Fig. 5b).

E. Collada Importer

A variety of file formats for describing the geometry of the robot model or the kinematic chain currently exist, such as VRML [15] or URDF (provided by ROS [34]). However, for the presented framework it has been decided to use Collaborative Design Activity (COLLADA) [35], an open-standard XML schema for digital assets developed by Sony Computer Entertainment Inc. (SCEI) and maintained by the Khronos Group [14]. It is supported by the open-source CAD tool Blender which provides a model development tool-chain based exclusively on open-source tools and it can easily be

extended to include custom data that can be validated against a custom Document Type Definition (DTD).

For the presented framework a COLLADA importer module has been developed to convert the static COLLADA-XML specification of the simulation model into a dynamic model within the SGAL. For model-file parsing the open-source COLLADA parser Collada-DOM [36] was used and custom extensions were implemented to make it possible to define and parametrize the tendon-driven actuator and linear spring-damper extensions implemented for the physics engine (see [18] and Section II-A). Bullet Physics provides maximum numerical stability for physical units being in the range of [0.05,10.0]. Hence, a world scaling feature was added to the importer using the COLLADA `<meter>` tag that makes it possible to conveniently scale the entire model to fit this range.

F. Collada Exporter

Despite the advantages of directly using CAD data for the simulation model, the derivation of an accurate model of the highly complex robots developed nowadays is becoming increasingly difficult. Hence, manual or automatic tuning of model parameters (e.g. using machine learning techniques) is employed to capture the complex dynamics of todays systems [37]. To make these model revisions persistent, a physics and graphics engine-independent COLLADA exporter module has been developed using the interfaces provided by the SGAL. Similar to the importer, the exporter uses the Collada-DOM library for writing the XML structures of the COLLADA file.

III. Framework Tools

Based on the CORBA interfaces provided by the ROSE (see also Section IV) a set of generic GUI tools has been developed.

A. System Tools

Increasing system complexities require advanced management tools to identify and debug possible errors or to investigate the current state of the system. Therefore, a distributed, CORBA-based logging service has been developed for the framework using the open-source library Log4cpp [38]. Each log entry comprises a time stamp, a priority (e.g. Debug or Info), a category identifying the sender and a detailed log message. A unique log-file name is created automatically for each simulation session and a viewer has been developed for browsing and filtering the log (see Fig. 5e).

B. Data Acquisition Tools

Important for model performance analysis is the possibility to investigate dynamic parameters such as forces or joint angles during simulation. Typically, such parameters are either exported manually or automatically using file streams or similar output techniques. This approach, however, requires external tools for filtering and visualization as well as frequent re-compilation and restarting of the simulation environment which is highly time-consuming and requires detailed knowledge of the simulator internals. To compensate for these drawbacks, CALIPER provides a flexible data acquisition system

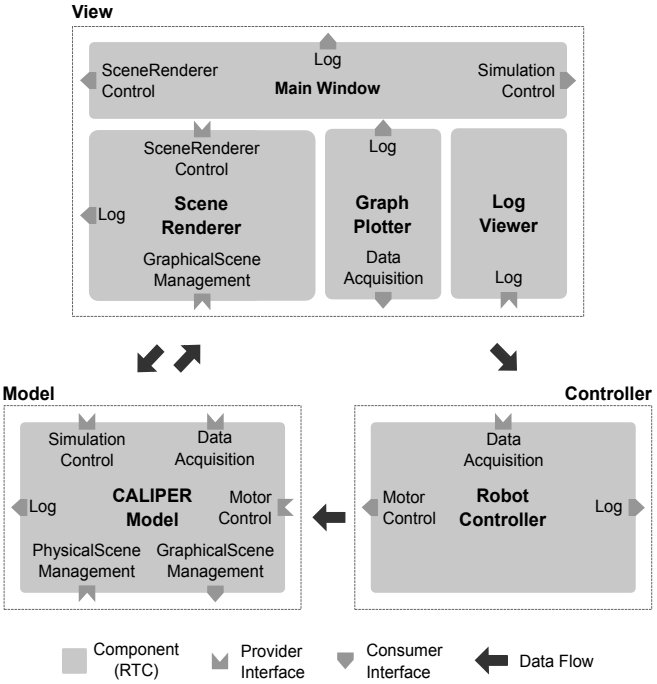

Fig. 4. CALIPER Component Architecture. The framework is designed using component-based development principles [41]. Each framework functionality is encapsulated in an individual software component with standardized, platform and programming-language independent producer and consumer CORBA interfaces. Hence, the framework can easily be extended or tailored depending on user-requirements without altering existing components. At the system level, the individual components can be grouped into *view*, *model* and *controller* components in accordance with the Model-View-Controller (MVC) [32] software design pattern. Additionally, every component is designed using the MVC pattern by separating the component into a data class, individual controller classes for each provided interface and an optional view class.

(DAS) for online parameter investigation. The DAS employs a producer-consumer pattern in which all existing simulation parameters are registered to the producer using Boost Functors [39] and unique string-identifiers. Subsequently, these identifiers can be used by the consumer to query the current value of the parameter. At the moment only scalar parameters are supported but complex data-types such as vectors or matrices will be added in the future. To visualize the queried model parameters a plotting widget has been developed using the open-source, Qt-based plotting library Qwt [40] (see Fig. 5d). The widget supports the scatter and line plotting of scalar parameters over time and provides tools for plot configuration, plot export, data export and printing.

IV. Software Architecture

Limited reusability, verifiability, extendability and maintainability are recurring problems of many existing software systems. One approach to tackle these problems is to make use of component-based software engineering (CBSE) [41]. In CBSE the overall system is divided into independent, replaceable components that encapsulate specific functionalities and provide standardized interfaces for inter-component communication.

One robotic middleware that supports such a CBSE approach and that is used for the presented framework is OpenRTM [33]. OpenRTM introduces state-driven robot-technology components (RTCs) and uses CORBA for inter-RTC communication. The advantage of CORBA is that it provides Remote Procedure Calls (RPCs) which offer a platform and programming-language independent way of executing a procedure in another address space and also enable advanced error handling using exceptions. Furthermore, CORBA makes it possible to distribute individual components of the framework on a computer cluster for performance scaling.

For the current framework the following RTCs were developed (see Fig. 4): (i) a CALIPER *Model* component that contains the physical model of the simulator and that provides interfaces for simulation control, physics-scene management, motor control and data acquisition; (ii) a *Main Window* component serving as a container for the remaining graphical-user-interface components; (iii) a *Scene Renderer* component comprising the graphical model and the Coin3D Viewer for scene rendering which provides a graphical scene management interface for transformation matrix update and a control interface for invoking rendering commands; (iv) a *Graph Plotter* that visualizes model as well as controller parameters using the data acquisition interface and (v) a *Log Viewer* for displaying the log messages received through the logging interface consumed by all other framework components.

At the system level, these components can be grouped into *view*, *model* and *controller* components—in accordance with the Model-View-Controller (MVC) software design pattern [32] (see Fig. 4). Since its first introduction in 1979 [42], this pattern has become highly popular in software development. One reason for this is that each MVC component can be developed and tested separately which in turn accelerates the development of secure and easy-to-maintain software. Hence, the MVC pattern has not only been deployed at the system level but also at the RTC level by dividing each RTC into a data model class, various interface classes as controllers and a GUI class as viewer for components comprising a user-interface.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

This paper presents a novel robot simulation framework for tendon-driven robots that is capable of importing and exporting CAD robot models using COLLADA, an XML-based, open-standard exchange format for digital assets. Custom physics engine extensions are included in the framework that make it possible to simulate DC-motor based, compliant tendon-driven actuators as well as passive linear spring-damper *ligaments* and Hill muscles. All extensions can be conveniently parameterized using custom COLLADA extensions. The exclusive usage of open-source 3rd-party libraries for the framework provide possible users with an unprecedented freedom in adjusting the simulation environment to the requirements of a specific robot or application. Moreover, the component-based and service-oriented framework design hopefully provides the foundation for a steadily-growing, community-based development of framework extensions such as sensor or actuator libraries. Another key-feature of CALIPER is the provision of real-time interfaces for robot control and static as well as dynamic model parameter access. These interfaces and the provided graphical user interface tools make it possible to efficiently investigate the model dynamics in real-time. The framework has been already successfully employed for simulating the dynamics of an anthropomimetic robot arm [18] as well as a software-in-the-loop (SIL) tool for automated model calibration based on Evolution Strategies and for controller development (unpublished data). A video demonstrating the CALIPER integration for the ECCEROBOT project can be found online [12].

B. Future Works

For future CALIPER versions the comprehensive feature list will be extended further. Functionalities that will be implemented include the extension of the data acquisition interface to support complex data types and the implementation of scene manipulation tools to create and modify simulation entities through a graphical user interface. Moreover, the framework will be extensively validated to further improve the stability and usability of the application. CALIPER is currently in preparation for publication under an open-source license.

VI. ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Challenge 2- Cognitive Systems, Interaction, Robotics - under grant agreement no. 231864- ECCEROBOT.

REFERENCES

[1] Common Object Request Broker Architecture CORBA. Object Management Group (OMG). [Online]. Available: http://www.corba.org/

[2] S. Wittmeier, G. Song, J. Duffin, and C.-S. Poon, "Pacemakers handshake synchronization mechanism of mammalian respiratory rhythmogenesis." *Proc Natl Acad Sci USA*, vol. 105, no. 46, pp. 18 000–18 005, Nov 2008.

[3] W. Leontief, F. Duchin, and D. B. Szyld, "New Approaches in Economic Analysis," *Science*, vol. 228, no. 4698, pp. 419–422, 1985.

[4] R. Blumhardt, "Fem - crash simulation and optimisation," *International Journal of Vehicle Design*, vol. 26, no. 4, pp. 331–347, Jan. 2001.

[5] J.-J. E. Slotine and W. Li, "On the Adaptive Control of Robot Manipulators," *The International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987.

[6] Open Robotics Automation Virtual Environment (OpenRAVE). [Online]. Available: http://www.openrave.programmingvision.com/

[7] J. Jackson, "Microsoft robotics studio: A technical introduction," *Robotics Automation Magazine, IEEE*, vol. 14, no. 4, pp. 82–87, dec. 2007.

[8] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2004, pp. 2149 – 2154 vol.3.

[9] O. Michel, "Cyberbotics Ltd. Webots TM : Professional Mobile Robot Simulation," *Int. Journal of Advanced Robotic Systems*, vol. 1, pp. 39–42, 2004.

[10] O. Holland and R. Knight, "The anthropomimetic principle," in *Proceedings of the AISB06 Symposium on Biologically Inspired Robotics*, 2006.

[11] H. Marques, M. Jäntsch, S. Wittmeier, O. Holland, C. Alessandro, A. Diamond, M. Lungarella, and R. Knight, "Ecce1: The first of a series of anthropomimetic musculoskeletal upper torsos," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, 2010, pp. 391 –396.

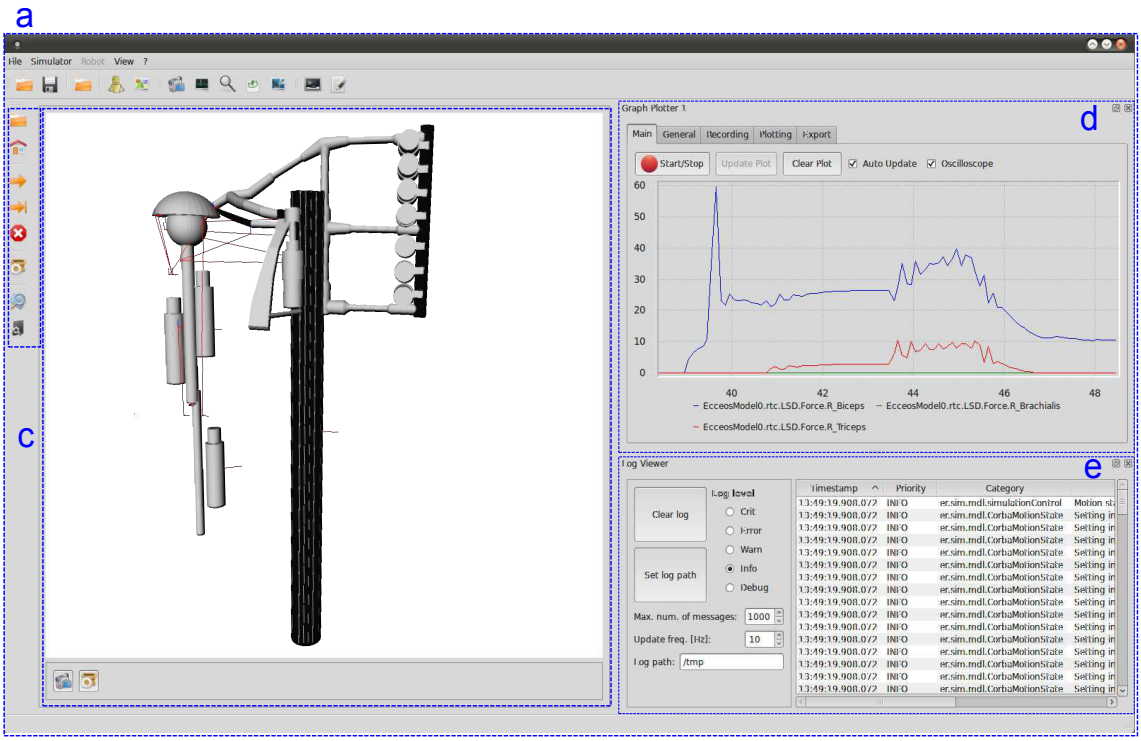

Fig. 5. CALIPER GUI Screenshot. The individual framework tools can be started using the interfaces provided by the *Main Window* (**a**) which also contains tools for saving the state of the current session (e.g. window size and position). CAD models can be loaded and simulated using the *Simulation Control Tools* (**b**). For visual feedback, the current state of the simulation can be displayed by the *Scene Renderer* (**c**). Dynamic model parameters can be directly analyzed in real-time during simulation using the *Graph Plotter* (**d**). In case of errors, the current system state can be investigated by browsing the system log using the *Log Viewer* (**e**).

[12] Embodied Cognition In A Compliantly Engineered Robot (ECCEROBOT). [Online]. Available: http://www.eccerobot.eu

[13] S. Delp, F. Anderson, A. Arnold, P. Loan, A. Habib, C. John, E. Guendelman, and D. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *Biomedical Engineering, IEEE Transactions on*, vol. 54, no. 11, pp. 1940 –1950, 2007.

[14] COLLADA – 3D Asset Exchange Schema. Khronos Group. [Online]. Available: http://www.khronos.org/collada/

[15] A. L. Ames, D. R. Nadeau, and J. L. Moreland, *The VRML 2.0 sourcebook (2nd ed.)*. New York, NY, USA: John Wiley & Sons, Inc., 1997.

[16] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[17] H. G. Marques and O. Holland, "Architectures for functional imagination," *Neurocomput.*, vol. 72, no. 4-6, pp. 743–759, 2009.

[18] S. Wittmeier, M. Jäntsch, K. Dalamagkidis, and A. Knoll, "Physics-based Modeling of an Anthropomimetic Robot," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, accepted.

[19] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pp. 281–288, 2007.

[20] M. DeLoura, "Middleware showdown," *Game Developer*, vol. 8, 2009.

[21] Open Dynamics Engine (ODE). [Online]. Available: http://www.ode.org

[22] E. Coumans. Bullet Physics Library. Sony Computer Entertainment. [Online]. Available: http://www.bulletphysics.com

[23] Havok Physics. Havok Inc. [Online]. Available: http://www.havok.com

[24] NVIDIA Physx. NVIDIA. [Online]. Available: http://www.nvidia.com/object/physx_new.html

[25] A. V. Hill, "The heat of shortening and the dynamic constants of muscle," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 126, no. 843, pp. pp. 136–195, 1938.

[26] Unreal Engine. Epic Games. [Online]. Available: http://www.unrealtechnology.com/

[27] CryENGINE. Crytek GmbH. [Online]. Available: http://www.crytek.com/

[28] Object-Oriented Graphics Rendering Engine (OGRE). [Online]. Available: http://www.ogre3d.org/

[29] Kongsberg Oil & Gas Technologies. Coin3D. [Online]. Available: http://www.coin3d.org

[30] Qt – A cross-platform application and UI framework. Nokia Corporation. [Online]. Available: http://qt.nokia.com/products/

[31] Physics Abstraction Layer. [Online]. Available: http://pal.sourceforge.net

[32] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[33] N. Ando, T. Suehiro, and T. Kotoku, *A Software Platform for Component Based RT-System Development: OpenRTM-Aist*. Springer Berlin/Heidelberg, 2008, vol. 5325.

[34] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *Proc. International Conference on Robotics and Automation*, 2009.

[35] R. Arnaud and M. C. Barnes, *Collada: Sailing the Gulf of 3d Digital Content Creation*. AK Peters, Ltd, 2006.

[36] COLLADA Document Object Model (DOM). [Online]. Available: http://sourceforge.net/projects/collada-dom/

[37] J. Sturm, C. Plagemann, and W. Burgard, "Body schema learning for robotic manipulators from visual self-perception," *Journal of Physiology-Paris*, vol. 103, no. 3-5, pp. 220–231, 2009, neurorobotics.

[38] Log4cpp. [Online]. Available: http://log4cpp.sourceforge.net/

[39] Boost C++ Libraries, functions. [Online]. Available: http://www.boost.org/doc/libs/1_43_0/doc/html/function.html

[40] Qwt—Qt Widgets for Technical Applications. [Online]. Available: http://qwt.sourceforge.net/

[41] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Addison Wesley, 1998.

[42] T. Reenskaug, "Thing-model-view-editor—an example from a planningsystem," Xerox PARC, 5 1979.