Title:	A Reconfigurable robot workCell for fast set-up of automated assembly processes in SMEs
Acronym:	**ReconCell**
Type of Action:	Innovation Action
Contract Number:	680431
Starting Date:	1-11-2015
Ending Date:	31-10-2018

Deliverable Number:	D4.1
Deliverable Title:	Sub-system for 3D simulation, visualization and interfacing with the user
Type (Public, Restricted, Confidential):	PU
Authors :	C. Schlette, E. Guiffo Kaigom, M. Priggemeyer, D. Losch, G. Grinshpun, R. Waspe, J. Roßmann, B. Ridge, and M. Tamosiunaite
Contributing Partners:	MMI, JSI, UGOE

Estimated Date of Delivery to the EC:	31-10-2016
Actual Date of Delivery to the EC:	04-11-2016

Contents

1 Introduction

The key feature of the ReconCell system will be its highly-modular, manufacturer-independent multi-robot design (see Figure 1) which allows for self-reconfiguration (e.g. by automatically positioning fixtures and sensors). Despite its complexity, (re-)configuration and (re-)programming will be enabled by a comprehensive, functional 3D virtual model of the system, which supports simulation, automated simulation-based optimization as well as simulation-based control of the assembly cell. This central user interface will be implemented based on previous developments of the project partners, particularly the VEROSIM system co-developed by the Institute for Man-Machine Interaction (MMI) of the RWTH Aachen University (see. Section 2). At MMI, we work on according technologies as part of the "eRobotics" methodology [23], a development platform for roboticist to exchange ideas and to collaborate with experts from other disciplines. The central method in eRobotics are "Virtual Testbeds", where complex technical systems and their interaction with prospective working environments are first designed, programmed, controlled and optimized in 3D simulation, before commissioning the real system.

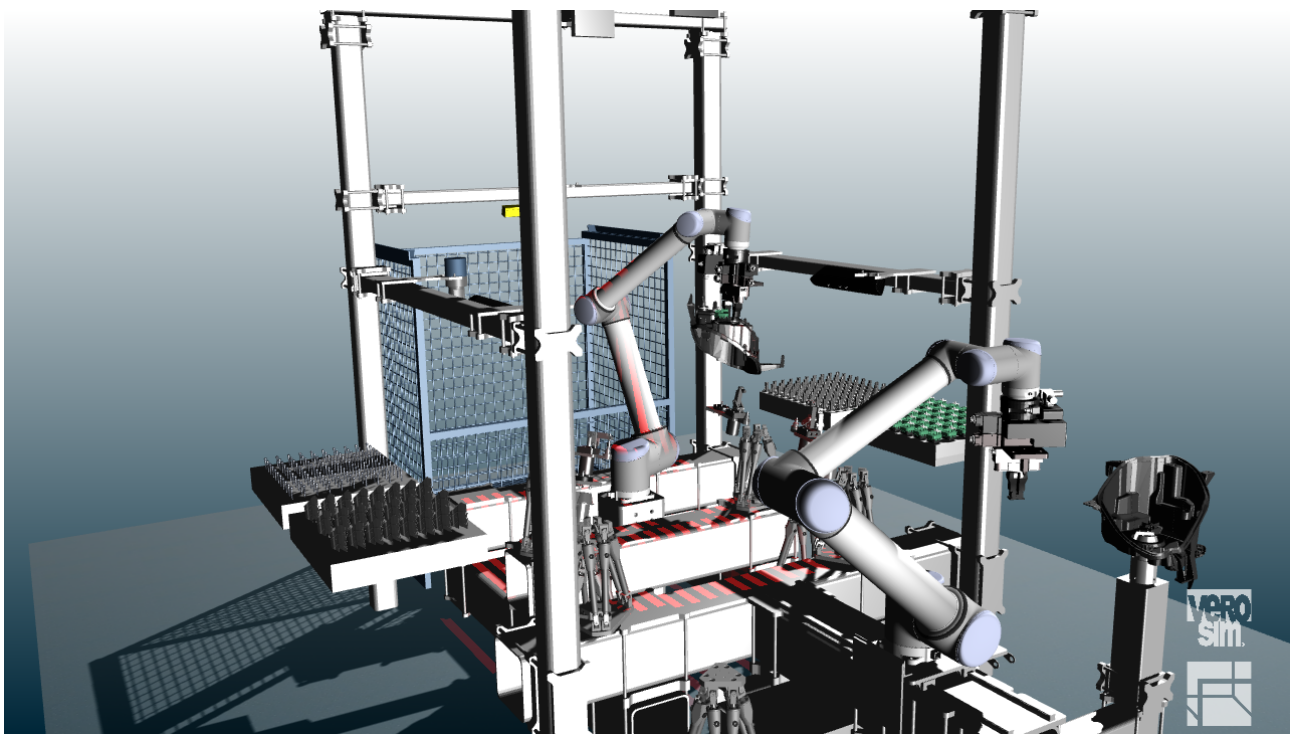


Figure 1: Design study of the ReconCell system, consisting of two manipulators (here two Universal Robots UR10 equipped with gripper systems by Schunk), passive hexapod fixtures which are reconfigurable by the robots and a frame for cameras and other sensors (here two combinations of Point Grey Bumblebees and Microsoft Kinects) which are also reachable and reconfigurable by the robots.

Figure 2 depicts the intended workflow with the ReconCell system, which evolves around the functional 3D virtual model of the ReconCell system. The CAD model and additional information about a desired product are given into the 3D simulation-based user interfaces for automated preparations, e.g. geometries and materials for the initialization of collision detection and dynamics simulation as well as a basic description of the desired assembly sequence. The

basic assembly sequence will then be refined to a detailed sequence of applied and parameterized skills mainly by means of visual programming (see Section 6.2). The resulting representation based on "ActionBlocks" will allow for carrying out the assembly process (including simulation of robot kinematics and dynamics) in virtual reality (see Section 3) and in real experiments (see Section 4), simulation of assembly steps (see Section 5) and sensor simulation (see Section 6.3). The simulation will also enable automated means of optimization, e.g. by deriving optimized positions for fixtures and sensors. Finally, if all steps have been verified in simulation, the very same "ActionBlock" representation will be the basis for commanding and excuting the assembly process on the real setup.

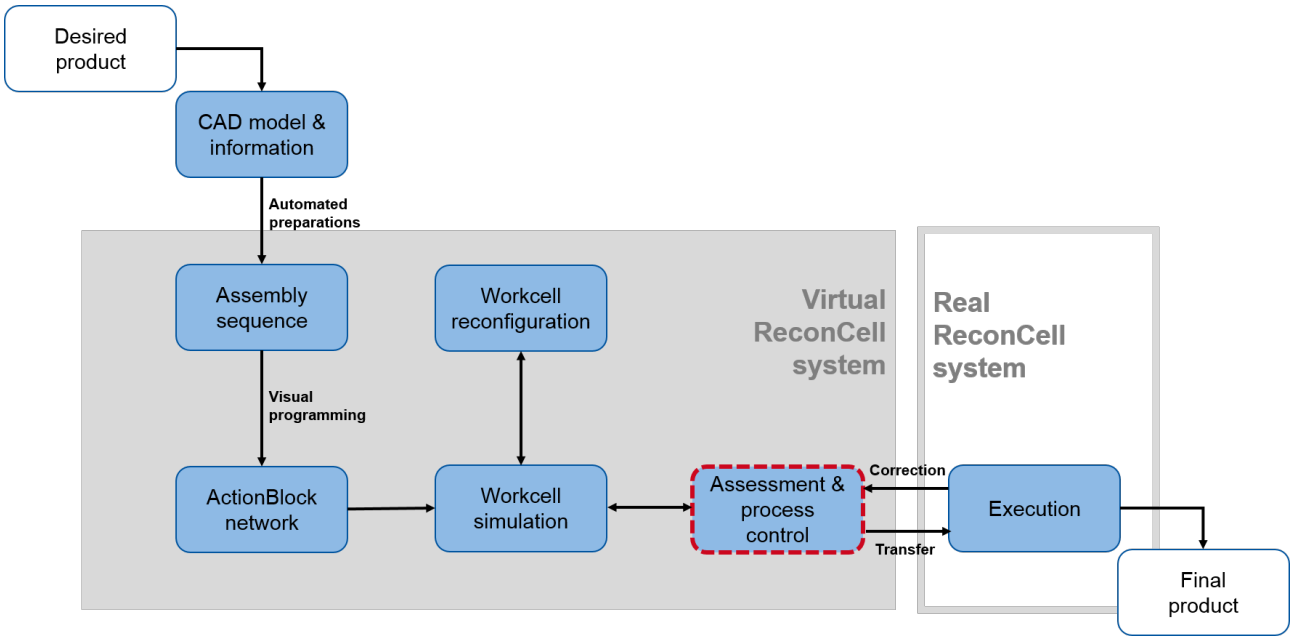


Figure 2: Intended workflow of the ReconCell system (left to right).

2 The VEROSIM platform

The major prerequisite for realizing eRobotics concepts on the tool level is the use of one single but comprehensive and integrated 3D simulation framework which is able to implement the methods and support the processes outlined above.

2.1 Requirements

The major advantage of such an integrated framework is the ability to simulate all components within single Virtual Testbeds, while minimizing conversion tasks between various subsystems. This leads to various requirements of the underlying 3D simulation framework:

- Overall Flexibility: The simulation system must support a broad range of applications and usage scenarios (see Figure 3), by the way enabling to be used a) as an engineering tool on the desktop, b) as an interactive system to realize complex user interfaces and c) as a tool for realizing control algorithms on real-time capable systems. Hence, it has to separate simulation algorithms from user interface implementation.

- Freely Configurable Database: In order to be able to address the manifold of assembly scenarios at SMEs, the underlying data model has to be freely adaptable to new components and products. To allow all methods to be based on the same model which contains (on an equal level) geometric information as well as e.g. sensor configurations or controller programs, a meta data system and a reflection API are necessary,

- Calibrated Simulation Algorithms: In order to obtain valid and reliable results, the simulation algorithms have to be calibrated against real systems.

- Seamless Transition from Simulation to Reality: The use of block-oriented data models and subsequent code generation for controller implementation is a quasi standard in "Rapid Control Prototyping. This well-established workflow should also be available in the 3D simulation system, such that data processing algorithms developed with and integrated into the system can be used on the real hardware.

The state of the art in simulation technology lists various general approaches to simulation: Discrete event simulation systems [1], block-oriented simulation approaches like the Matlab/Simulink framework [14] or the "Modelica" modeling language [6] as well as various FEM-based simulation tools (e.g. COMSOL [3]) are probably the most well-known ones. Regarding quasi-continuous 3D simulation technologies in robotics and automation, available approaches are development frameworks (e.g. ROS [16] and GAZEBO [7]) or generic mechatronic systems (e.g. Simscape [24]).

In summary, most approaches focus on dedicated application areas, dedicated disciplines (electronics, mechanics, electronics, thermodynamics, etc.), or are restricted to the development of single components. Still missing is a holistic and encompassing approach, which enables and encourages the synergetic use of simulation methods on a single database throughout the entire lifecycles of technical system.

2.2 Simulation System architecture

VEROSIM strives to overcome these limitations and to fulfill the requirements listed above.The key idea is the introduction of a micro kernel, the “Versatile Simulation Database” (VSD, see Figure 3). Basically, the VSD is an object-oriented real-time database holding a description of the underlying simulation model. Fully implemented in C++, it provides the central building blocks for data management, meta information, communication, persistence and user interaction. The VSD is called "active" as it is not simply a static data container, but also contains the algorithms and interfaces to manipulate the data. Furthermore, the VSD incorporates an intelligent messaging system that informs interested listeners of data creation, change and deletion events.

The functionality of the micro kernel is extended by various plugins implementing simulation or data processing algorithms, interfaces to hard- or software systems, user interfaces, etc. Using the VSD, the plugins can communicate with the database as well as establish directed communications between themselves.

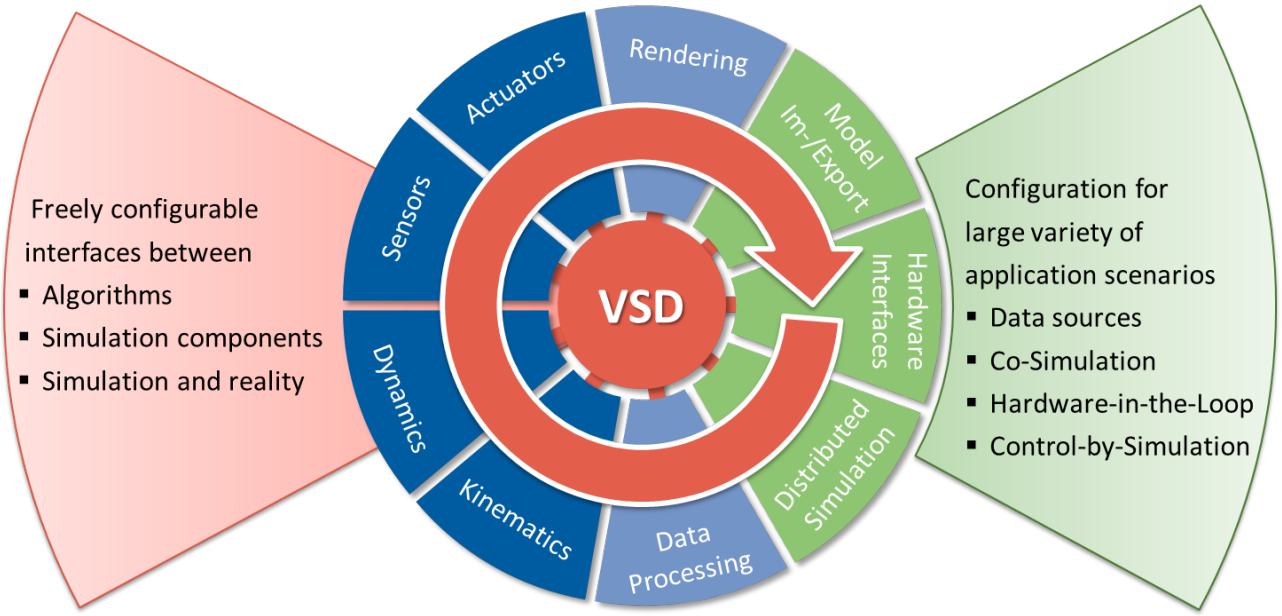


Figure 3: The VEROSIM microkernel architecture.

3 Robot simulation and control

Due to its design and structure, the VSD allows for the fast development of challenging automation solutions. This development is made possible by the integration capabilities emphasized in Figure 3 and messaging advantages pointed out in Sec. 2.2 which, when combined, offer ideal tools upon which ready-to-use robotics solutions can be provided. Toward this end, basic robotics functionalities are intuitively put at disposal for a seamless integration into and successful employment in multidisciplinary higher level applications in Virtual Testbeds. Each of these functionalities captures and reflects a fundamental level of abstraction of the behavior of a physical robot in association with intermittent interactions with its workspace. While the kinematic abstraction focuses on a geometric description of the behavior of objects to which a robot and its surrounding environment belong, the dynamic abstraction considers the inertia properties of these objects. Both abstractions are loosely coupled through a third abstraction that delivers control forces to the dynamics in order to meet desired motions specified by the kinematics at runtime.

3.1 Kinematics

Kinematics in VEROSIM allow for defining and programming the motion of individual objects as well as kinematic chains, solely based on their position and orientation, velocity and acceleration. Kinematics are often used as a first step to model robots by defining joints between articulated links. In some applications, series of joints can be controlled as individual paths in kinematic trees along trajectories in configuration space or Cartesian coordinates [20]. In the ReconCell context, these trajectories are the desired motions the robot must follow by using additional motion control components for a successful completion of targeted manipulation objectives, as shown in Figure 4.

3.2 Dynamics

As an exchange of mechanical work in terms of contact force and velocity between a robot and its operational environment becomes a stringent requirement, considering the inertia properties of objects that populate a scene becomes mandatory. For the sake of generality, a uniform treatment of these interactive dynamics has been adopted in Cartesian coordinates. While these coordinates facilitate the modeling of challenging configurations of single bodies that form a robot as well as its workspace, constraint forces are injected, when required, between pairs of bodies for two important goals.

- The first goal is a systematic mechanical assembly of multi-body systems. In the case of a manipulator, this assembly is done by inserting torque-controlled joints between its links for articulation purposes. The same joint functionality also enables a translatory motion of the base of a robot along a linear axis, as is the case with a kinematic abstraction of the same multi-body systems. The choice of the underlying abstraction level is left to the user and will be influenced by the objectives of the task at hand in the ReconCell system.
- The second goal is the rendering of a natural interactive dynamic behavior of multi-body systems. More specifically, a frictional contact dynamics is enforced between each pair of physically interacting bodies by employing constraint forces which prevent these bodies to penetrate each other.

In order to determine the next state of multi-body systems constituting a scene, the physics engine detailed in [9] computes constraint forces by solving a linear complementary problem that characterizes the constraints at hand. These forces are then introduced into an expression of the law of Newton that apprehends the scene as a set on single independent bodies. The Cartesian velocities of the center of mass of each bodies in the scene follow from this expression. Given this set of velocities, the configuration of a particular robot manipulator is easily obtained through an integration of those Cartesian body velocities related to its links.

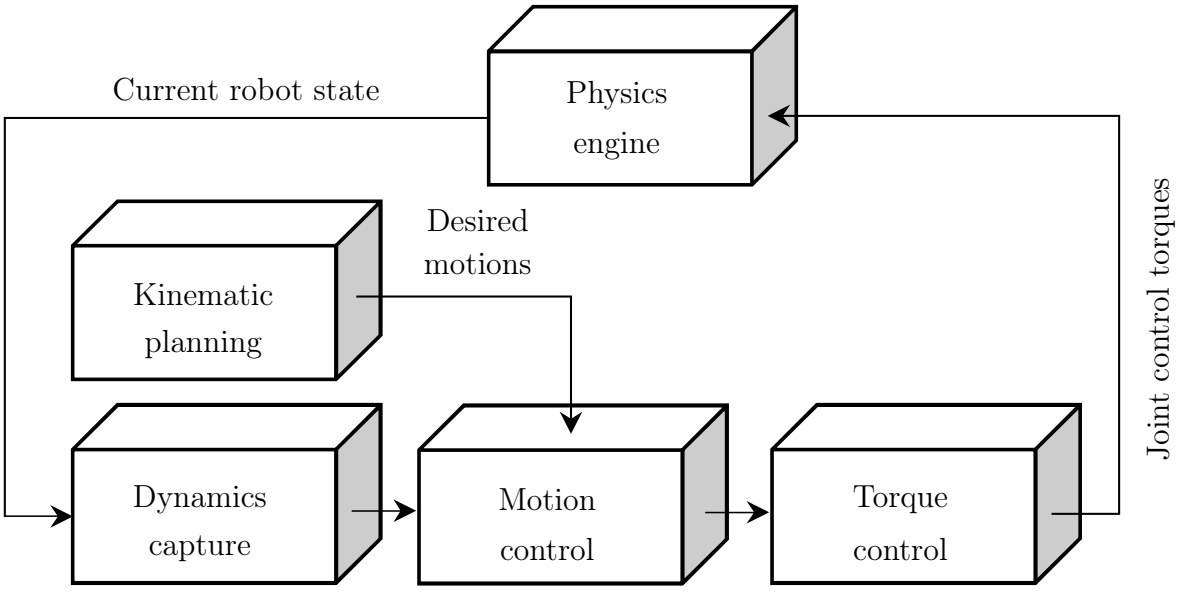


Figure 4: Simplified architecture of the overall approach for robot simulation.

It is worth to note that at this stage of pure multi-body simulation, a robot modeled in the simulator is not controlled. The robot will sink down under gravity load unless joint control torques are suitably provided as shown in Figure 4.

3.3 Control

A model-based trajectory following and interaction control provides the necessary tracking accuracy and skillful disturbance response to external forces to any simulated robot. As shown in Figure 4, the component that captures the current robot dynamics in joint coordinates, receives the current robot state made of the vectors of joint positions and velocities. On this basis [10], the inertia, Coriolis and gravity disturbances are systematically captured. This nominal dynamics provides an advantageous insight into the robot behavior. Indeed, this dynamics will be beneficial for the construction of advanced performance indexes that uncover the best possible motion efficiency potentials for the robots employed in ReconCell. Since the far reaching impact of this insight is not restricted to any robot type or application, the unique information flow on the robot dynamics can be interfaced by any high-lever application for specific purposes. These include different forms of supporting decision taking in business analytics as well as the rationalization of automation processes that rely on the usage of intelligent robots.

In fact, the nominal dynamics of the robot facilitates a subsequent development of numerous skillful motion control approaches, such as joint admittance control. In this compliance control scheme, the goal is to absorb the impact energy in the transient phase of contact and render a desired stiffness at steady state. For this, commanded joint control torques are delivered by the torque control component. These torques are finally fed into the physics engines in order to actuate the robot joints, which leads to a new robot state (see Figure 4).

3.4 Dynamics customization

Though concepts behind ReconCell are not tailored to specific types of manipulator, the Universal Robot 10 (UR10) has been chosen as reference manipulator. A major reason is the fact that UR10 brings advantageous servicing capabilities that are beneficial to ReconCell. These include light-weight, fast reconfiguration, force sensitiveness and almost system-wide monitoring. With the aim of offering similar features in VEROSIM, it was necessary to endow the UR10 model with dynamic parameters like kinematic data, inertia properties of links as well as the power train parameters which are highlighted on the right hand side of Figure 5. For this purpose, modules in charge of capturing the multi-body dynamics and motion control of digitized manipulators have been extended with interfaces (called *extension* in VEROSIM) that facilitate the specification of dynamic parameters. The customization is done on a graphical unit interface. In this respect, care has been taken in ensuring an intuitive and user-friendly usage of these interfaces. For instance, selecting a component of the robot on the *explorer*-view, as displayed on the left hand side of Figure 5, triggers on the *properties*-view the related interface on which parameters of the selected element (in our case a power train) can be assigned. Users can therefore take advantage of this modular and contextual customization in order to quickly setting up manipulators and quantitatively monitoring their pertinent emerging properties, as shown in the *oscilloscope*-view on Figure 5 that reflects the corresponding joint torques.

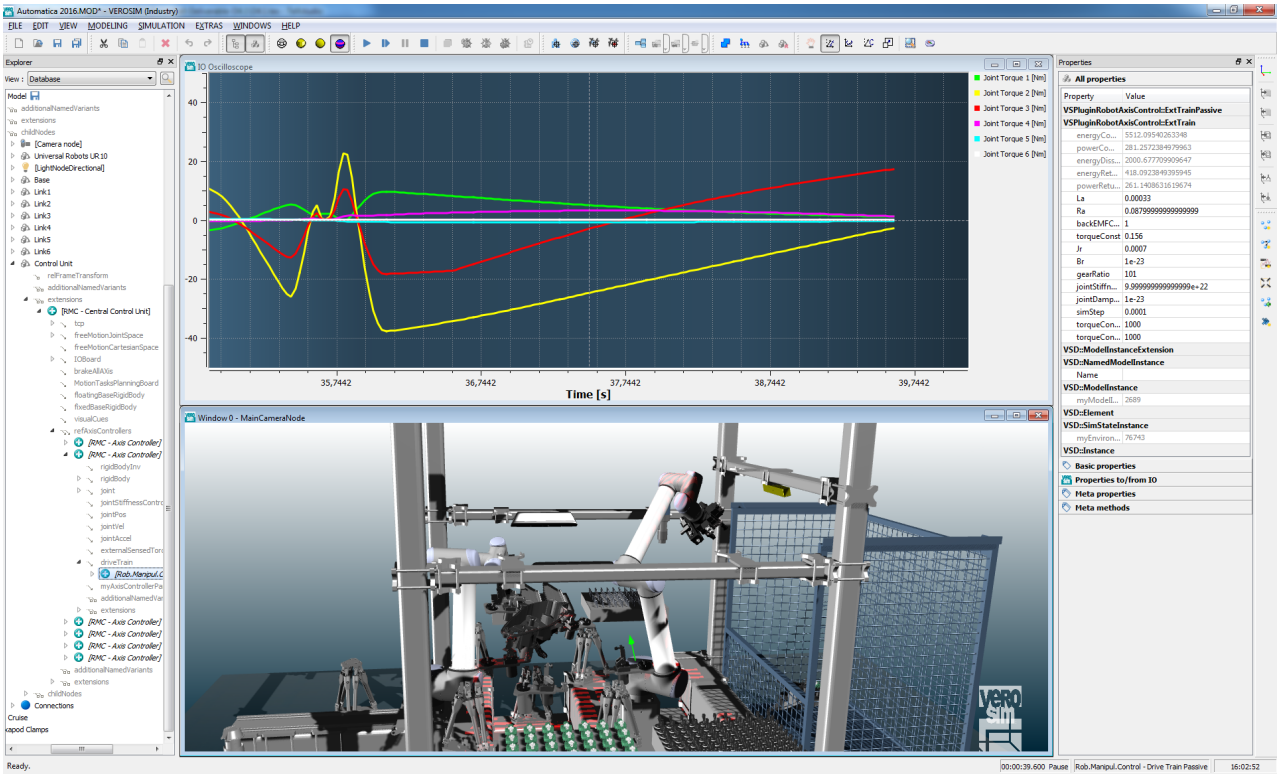


Figure 5: Robot dynamics has been customized by using identified parameters.

4 Physical robot control

The robots are modelled as kinematic chains and moved by a kinematic controller, as described in section 3.1 and [19]. The desired motions are programmed as point-to-point movements (PTP) in joint coordinates or continous path movements (CP) in cartesian coordinates. These motion paths are combined with velocity profiles to compile a trajectory as a function of time. If a PTP movement is desired, the joint increments are given directly to the kinematic. In the case of a CP movement, the compatible joint increments for the axes of the robot are calculated based on the inverse kinematics. The desired joint positions are then sent to the physical robot in each communication time step. An experimental setup was build to test VEROSIM's capabilities to control an UR10 robot (see Figure 6). The control mechanisms can be directly transferred to the ReconCell. To achieve this, different methods were implemented to communicate with the ReconCell technical infrastructure.

The interface to the robot allows different modes, which define the underlying protocols and physical connections. Figure 7 shows different components involved in controlling the ReconCell. VEROSIMcan take different roles in this setup. These include monitoring functions, exclusivly providing status information about the ongoing process. Alternatively, VEROSIMcan be used to control the robot in two different ways: On the one hand, direct control, bypassing the whole infrastructure, can be utilized (Figure 7, green connection between VEROSIMand UR10 robot). This allows VEROSIMto handle an ongoing process in the ReconCell entirely. On the other hand VEROSIMcan be integrated in a ROS (Robot Operating System [15]) network. It will appear as a ROS node providing functionality and services by publishing ROS topics (Figure 7, VEROSIMintegrated in ROS network).

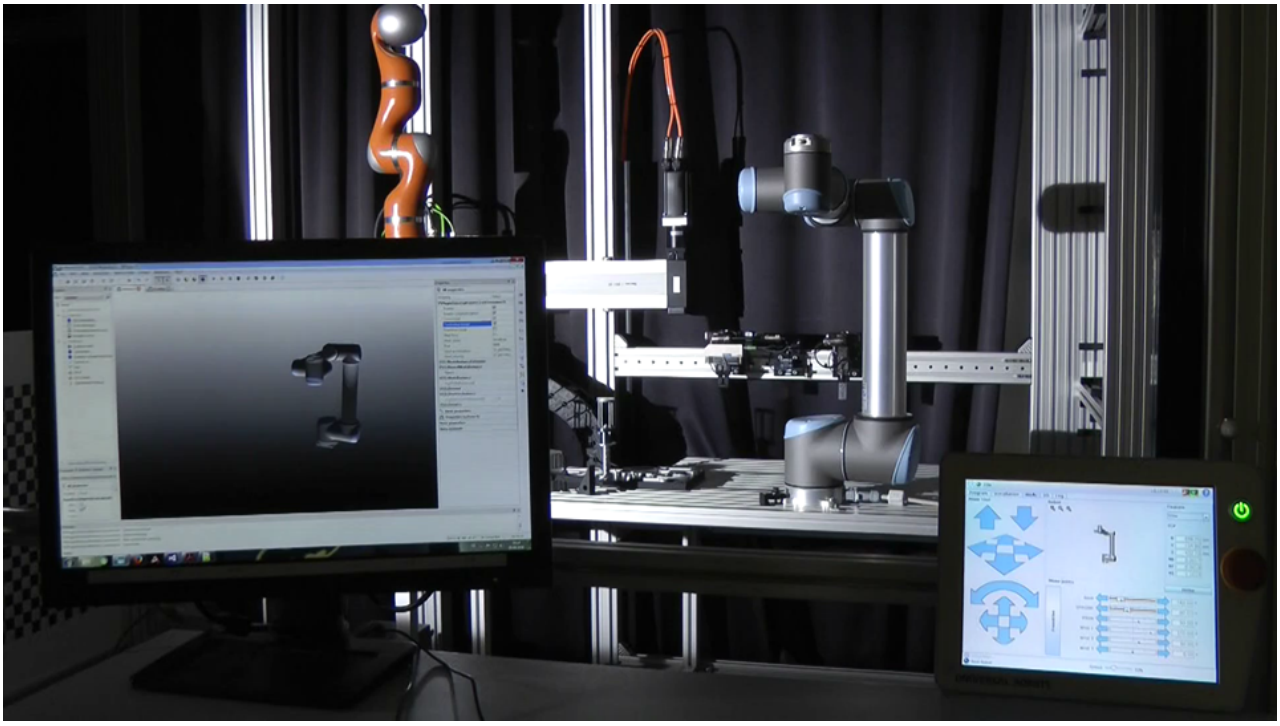


Figure 6: Experimental setup for the verification of UR10 movements.

5 Assembly execution and simulation

Complex industrial assembly sequences in ReconCell are composed of elementary tasks. Such elementary tasks are enabled in the robot controllers by so-called skill primitives, which represent high-level control functions resp. stategies to solve specific tasks. At MMI, we focus on the development and validation of skill primitives to handle peg and hole operations, since most industrial assemblies can be described in terms of peg and hole connections. For the development skill primitives to address peg and hole operations, we used KUKA LWR robots so far, but will switch to Universal Robots UR10 in future experiments for ReconCell (see Figure 5 and 1).

During an assembly execution, the robots grasp and bring the peg and hole parts in contact with each other according to strategies described in the skill primitive, while reacting on the situation and and adapting its strategy based on the data from the internal and external sensors. For the simulation of peg and hole operations as part of the ReconCell user interfaces, we model the given assembly scenario in VEROSIMbased on detailed virtual representations of the robots, the sensors and the peg and hole parts. This functional 3D virtual model of the assembly scenario then allows to parameterize, optimize and test specific combinations of assembly components and peg and hole parts using methods of Visual Programming.

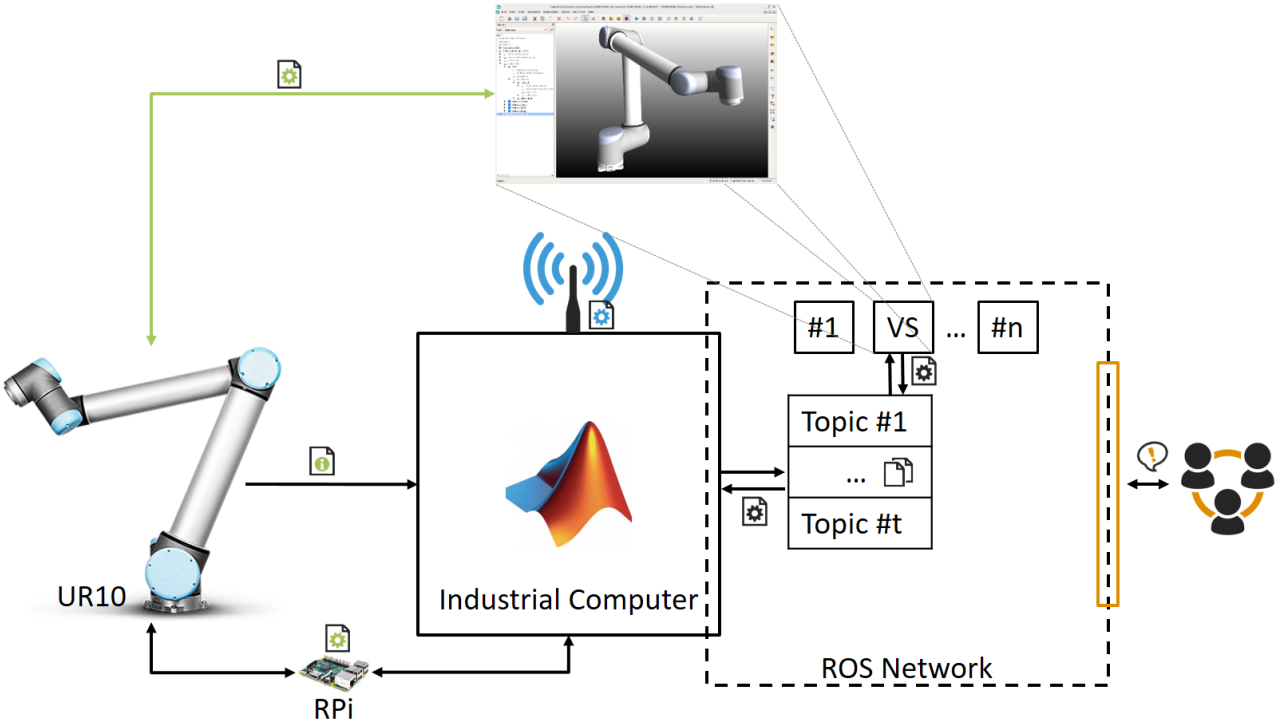


Figure 7: Communication layout between VEROSIM and UR10 robot. In the diagram, VS is an abbreviation for VEROSIM, and RPi stands for Raspberry Pi. Regarding the icons, gear wheels indicate the flow of command and status data, the info icon stands for status data only. The exclamation mark marks data provided for users.

6 3D simulation-based user interface

6.1 3D rendering

It is still a challenge to develop a comprehensive multidomain VR simulation system that has the capabilities to serve as a development basis in a wide range of application areas. The combination of accurate simulations and a modern rendering framework is a key feature of eRobotics systems. Modern software design patterns, as well as the concepts of semantic world models and graph databases provide new opportunities for the development of a new class of modern multi-domain VR simulation systems. As illustrated in the right part of Figure 8, the key idea is to base simulation and rendering components on a central, active and extendible object-oriented graph database. This approach grants the close interplay between all modules involved. Each module can define its own view to the database and integrate subgraphs to meet its specific data structure requirements. The semantic database also acts as connecting element between the simulation and rendering framework and grants the inter-communication between them. A multigraph can be created by adding references to other elements, thus defining additional arbitrary sets of edges within the database.

In accordance with the flexibility requirements to realize a sustainable and manageable system, a modular architecture is vital. The micro-kernel pattern is a popular approach to address the complexity problem. Originally developed for operating systems and embedded systems, it can also be applied to simulation systems that have to adapt to changing conditions. The main idea is to separate a minimal functional core from extended functionalities and project-

specific parts. As illustrated in Figure 3, the micro-kernel defines the very basic structure of the framework and also serves as a socket for plugging in these extensions and coordinating their collaboration.

The VSD described in section 2.2 provides the ideal basis to realize the Model View Controller (MVC) pattern for rendering. Plugins can integrate transformation rules that transform the semantic data graph into domain-specific data structures. For example, the 3D geometry plugin defines a set of semantic nodes (VSD3D), which contain basic nodes like geometric elements, texture nodes, material nodes, etc. that can be used to model the scene in a "natural" fashion. A set of rules (controller) is also defined by the VSD3D that transform the semantic nodes into a renderingoptimized data structure (view) automatically, using the data monitoring possibilities provided by the VSD. The used models are completely decoupled from system-specific requirements. This technique enables ease modelling and the efficient integration of modern rendering techniques, ranging from realistic ground vegetation over weather effects up to advanced data visualization techniques [8].

Modern rendering techniques and attractive virtual worlds not only ease the development in virtual testbeds and support the understanding of simulation results, they are also beneficial to improve the performance and accuracy of specific simulations tasks. As illustrated in the Figure 8, all components in an eRobotics-capable multi-domain VR simulation system are interconnected. The rendering framework can actively support specific simulation processes. Considering the simulation of optical sensors, the arising advantages are twofold. Firstly, a realistic looking virtual environment directly enhances realism when using the rendered images as input for digital camera simulations. Secondly, advanced rendering techniques can be applied to simulate other optical sensors like time-of-fight (ToF) cameras or laser range scanners (LiDAR) with high accuracy.

Figure 9 illustrates how an application can request a sensor data stream through a sensor

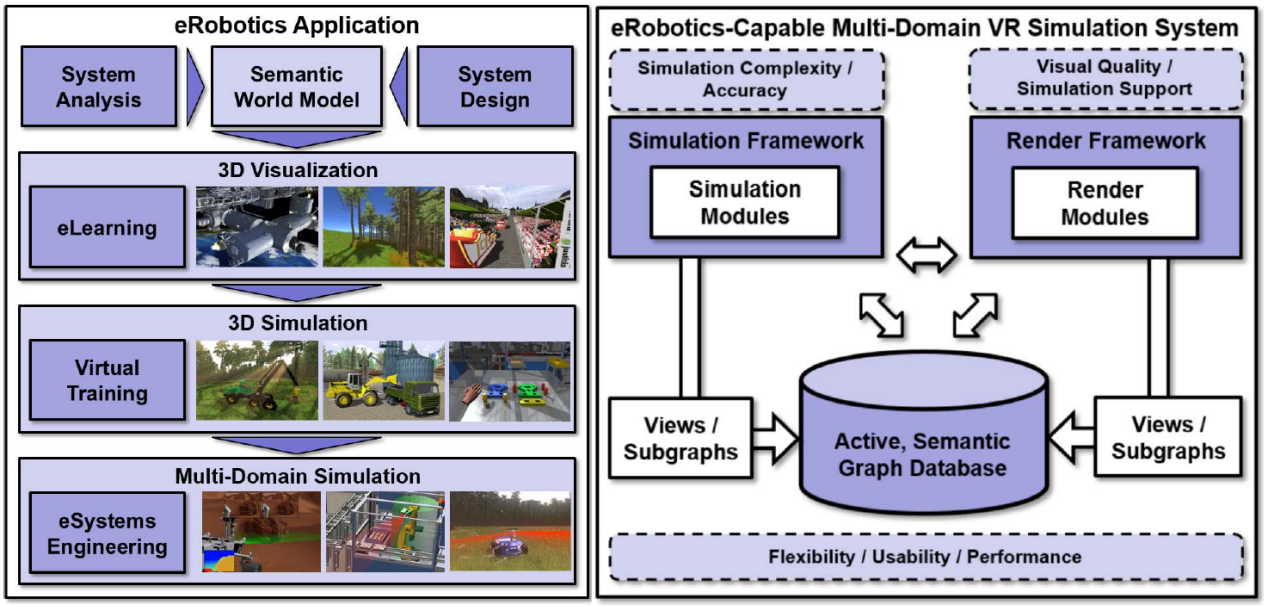


Figure 8: Left: Model descriptions are based on semantic world models. Right: System implementation with the close interplay between a central semantic graph database and a modular simulation and rendering framework.

abstraction layer. This layer triggers a simulated or a real world sensor and provides the acquired data to the application for further processing. In an ideal case, the application performs similarly with the simulated and the real world data. The general simulation process consists of a simulation plugin that can apply rendering techniques and data provided by the rendering framework to support accurate real-time sensor simulations. Figure 10 illustrates the structure of a rendering-supported camera simulation plugin.

6.2 Visual programming and ActionBlocks

Visual Programming is a programming paradigm that aims to replace or augment conventional, text-based programming. Since visual approaches are in general more concrete, direct, explicit and allowing for direct visual feedback [2], they target inexperienced robot programmers and process engineers utilizing a prototypical development approach. Examples for Visual Programming include the block- and icon-based development of algorithms for LEGO Mindstorms [11] or service robotics [4].

In previous projects we developed the concept of "ActionBlocks" to utilize the Visual Programming paradigm in the development and Virtual Commissioning micro-optical assembly processes [22, 21] and to simulate the results of symbolic planning algorithms that create optimized action sequences and processes [25]. The concept is used to visually program processes for agent-based control architectures: An ActionBlock represents a parameterized action that is executed by an agent – either in simulation or in the physical realm. ActionBlocks can represent actions of arbitrary complexity, from the simple activation of grippers to the execution of move-and-glue sequences that alternate kinematic movements and glue dosage. Figure 11a depicts the structure of an ActionBlock.

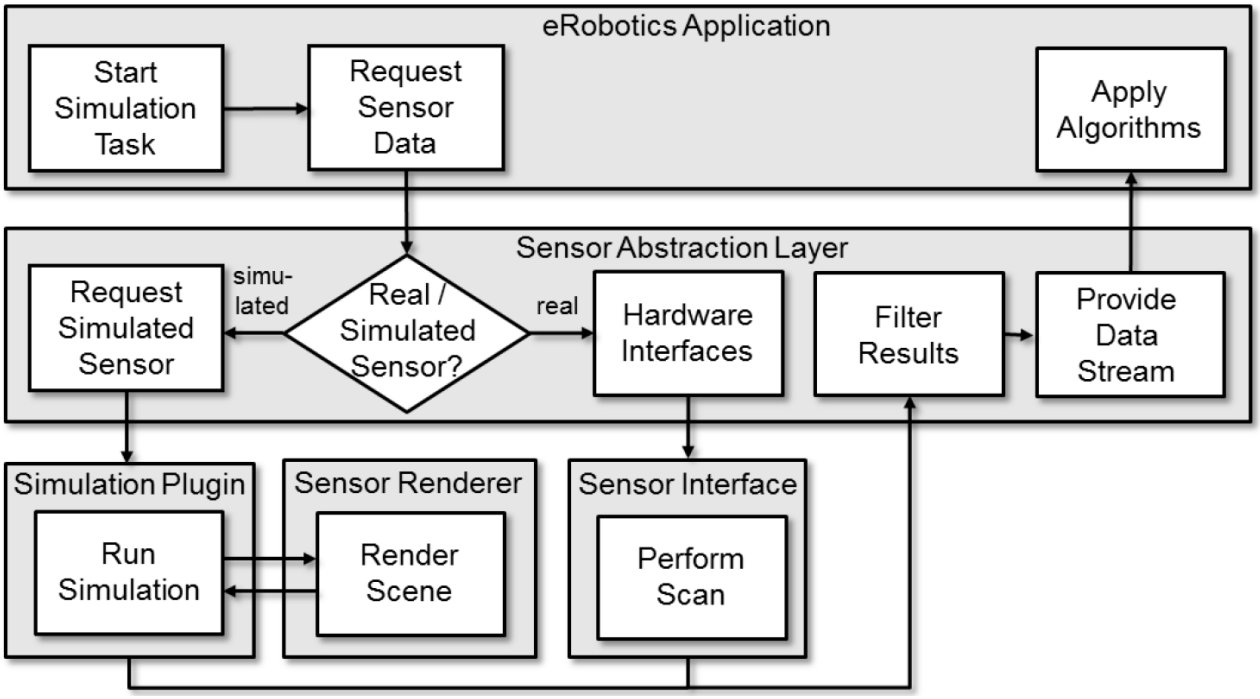


Figure 9: A hardware abstraction layer decouples application development from the applied sensors, regardless if real world or simulated ones.

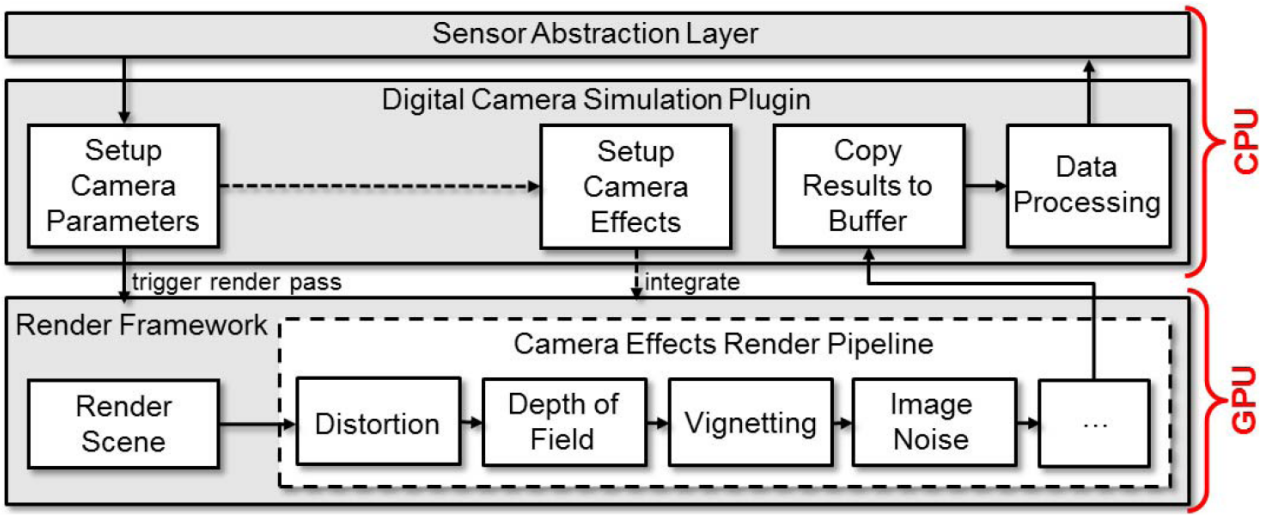
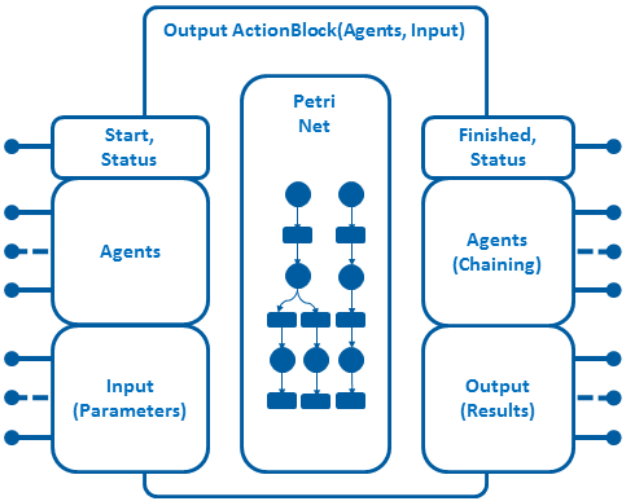


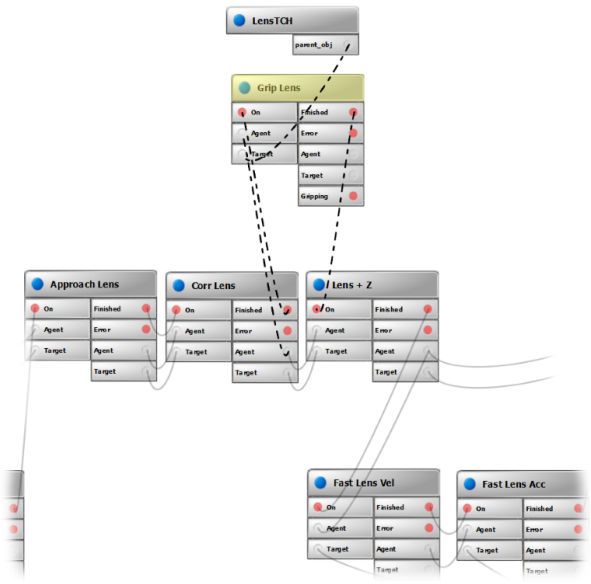
Figure 10: Structure of the camera simulation plugin, which applies a chain of post-processing effects to properly simulate typical camera effects.

ActionBlocks can be used to model a robotic work process on different layers of abstraction - from a high-level view of different process steps to a low-level view of the implementation of single process steps. Figure 12 visualizes this layered modelling concept.

Visual Programming with ActionBlocks is primarily carried out in a 2D data flow editor integrated into our simulation system. The data flow editor is used to visually connect ActionBlocks to a sequence of process steps and to assign agents and additional parameters to the ActionBlocks. Figure 11b shows an excerpt from an assembly sequence that was visually



(a) Basic structure of an ActionBlock.



(b) Exemplary network of Action Blocks as part of an assembly process.

Figure 11: ActionBlocks for Visual Programming in ReconCell.

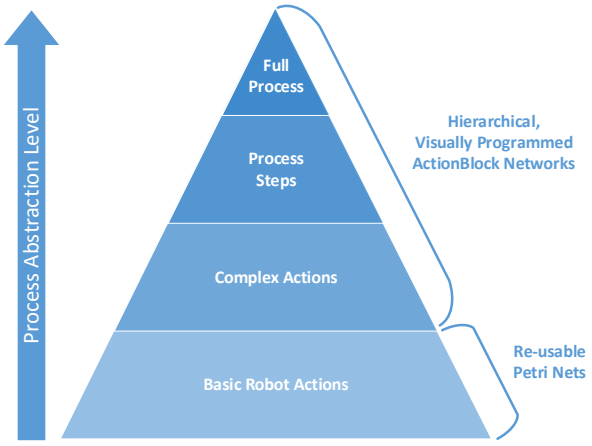


Figure 12: Layered process modelling with ActionBlocks.

programmed with ActionBlocks.

6.3 Sensor simulation and data visualization

The 3D simulation-based user interfaces in ReconCell depend on a realistic simulation of optical sensors, since only the application of sensors allows to bridge the gap between the ideal, virtual model and the real ReconCell system. The basis of the optical sensor simulation is the sensor framework in VEROSIM [5]. It provides methods for the modeling, simulation and visualization of a wide range of sensors and offers a consistent data interchange within the simulation environment as well as between real sensors and simulation algorithms in hardware-in-the-loop scenarios. Logging and playback mechanisms allow for an efficient offline development for real sensors while the introduction of various error models enable the detailed analysis of sensor data processing algorithms under different boundary conditions [18].

In detail, the sensor framework supports the parallel integration of real and simulated sensors into the system and provides a smooth transition between simulation and real world setups. The design of the sensor framework allows for easy setups of 3D virtual models using a generic communication concept for the interaction of all components offering a focused view on every component of the system to analyze and optimize its behavior. Figure 13 shows the sensor framework's system structure.

Due to the fact that simulated sensors are implemented to deliver or work on ideal data, specific error models need to be applied to grant close-to-reality simulations required for Virtual Testbed scenarios. As described in section 6.1, the system allows for a realistic simulation of a wide range of optical effects and errors. Electronic and optical effects with a huge impact on computer vision based algorithms are distortion, chromatic aberration, depth-of-field, noise and sensor saturation. On modern, programmable graphics hardware it is possible to map the characteristics of an optical sensor to a camera model of the render engine of the underlying simulation system by using shaders. As sensors like cameras deliver data consisting of several millions of measurement values, it is even more important to move the calculations from the CPU to the GPU to fulfill real-time simulation criteria. In contrast to other approaches, our approach takes place as an additional render pass which mostly utilizes data already loaded to the GPU [17].

Our approach allows multiple noise functions and supports the test and verification of computer vision algorithms. The different noise functions are hot-pixel noise, color noise and monochrome noise [18]. The noise characteristics are taken from real cameras. The simplest form of noise, hot-pixel noise, is obtained by taking images in complete darkness, but different temperatures. These images are used as noise textures and are added to the rendered image in a post-processing step. The reproducibility of highly dynamic noise effects can be accomplished through the active simulation time as seeds for distribution of semi-random noise values.

Distortion values for optical systems are provided by the optics manufacturer or can be measured or computed using standard computer vision algorithms as provided by OpenCV, MatLab Camera Calibration Toolbox or Zemax. Approaches to simulate distortion besides other effects are presented in [13] and [12].

Furthermore, the sensor framework supports sensors to directly gather depth information, for example time of flight (TOF) sensors. These send out a light beam towards the scene, that is being reflected and partly returned to the sensor. The TOF sensor calculates the distance to an object using the light's time of travel, that can simply be calculated. Another method is the measurement of the phase shift. A continuous wave laser continuously emits light with a modulated phase. Thus, the distance of an object can be calculated regarding the phase-shift of the received signal. From a simulation point of view, TOF sensors can be regarded as perfect pinhole cameras like the camera models in real-time rendering. Thus, they can easily be mapped to the graphics hardware. Various error models implemented for camera simulation can be applied to the output image of the simulated TOF sensor. Figure 14 shows an image gathered by a simulated TOF sensor.

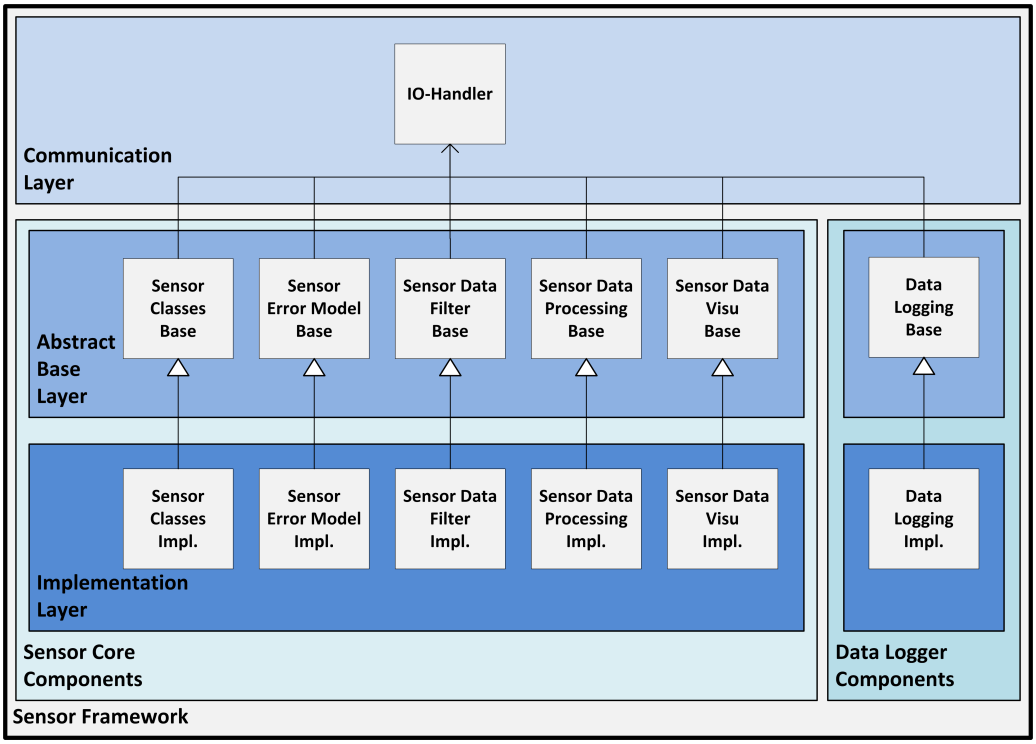


Figure 13: Structure of the sensor framework.

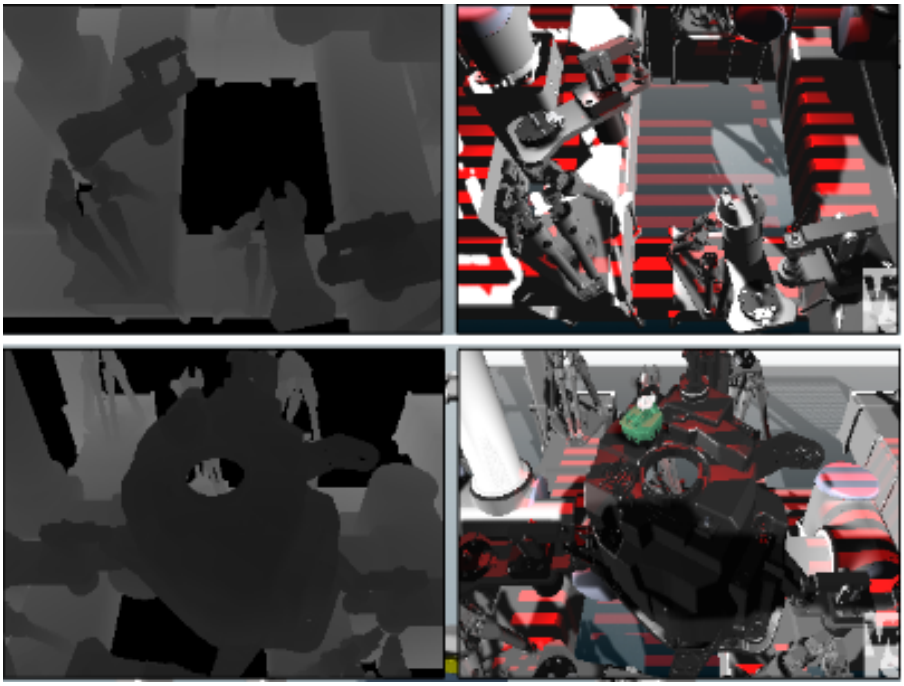


Figure 14: Image gathered by simulated RGB-D sensor in a 3D ReconCell. In particular, the simulated RGB-D sensors allow for the parameterization and optimization of algorithms for object recognition and pose estimation early on in the project.

References

[1] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation (3rd Edition)*. Prentice Hall, 2000.

[2] M. Burnett. "Software engineering for visual programming languages". In: *Handbook of Software Engineering and Knowledge Engineering*. Vol. 2. River Edge, NJ, USA: World Scientific, 2001, pp. 77–92.

[3] *COMSOL Multiphysics*. Comsol Group. URL: `www.comsol.com`.

[4] C. Datta, C. Jayawardena, I. Kuo, and B. A. MacDonald. "RoboStudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 2352–2357.

[5] M. Emde, J. Rossmann, B. Sondermann, and N. Hempe. "Advanced Sensor Simulation in Virtual Testbeds: A Cost-Efficient Way to Develop and Verify Space Applications". In: *AIAA Space 2011 American Institute of Aeronautics and Astronautics (AIAA) Conference and Exposition, Long Beach, California*. AIAA. 2011, pp. 1–6.

[6] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Computer Society Pr, 2003. ISBN: 0471471631.

[7] *GAZEBO*. Open Source Robotics Foundation. URL: `gazebosim.org`.

[8] N. Hempe and J. Rossmann. "A semantics-based, active render framework to realize complex eRobotics applications with realistic virtual testing environments". In: *Modelling Symposium (EMS), 2013 European*. IEEE. 2013, pp. 733–738.

[9] T. Jung. "Methoden der Mehrkörperdynamiksimulation als Grundlage realitätsnaher Virtueller Welten". PhD thesis. RWTH-Aachen University, Aachen, Germany, 2011.

[10] E. G. Kaigom and J. Roßmann. "Developing virtual testbeds for intelligent robot manipulators-An eRobotics approach". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 1589–1594.

[11] S. H. Kim and J. W. Jeon. "Programming LEGO mindstorms NXT with visual programming". In: *Proc. International Conference on Control, Automation and Systems (ICCAS)*. 2007, pp. 2468–2472.

[12] M. Kučiš, P. Zemčík, O. Zendel, and W. Herzner. "Overview of Simulation of Video-Camera Effects for Robotic Systems in R3-COP". In: *SAFECOMP 2013 - Workshop DECS (ERCIM/EWICS Workshop on Dependable Embedded and Cyber-physical Systems) of the 32nd International Conference on Computer Safety, Reliability and Security*. Ed. by M. ROY. Toulouse, France: Springer Science+Business Media B.V., 2013, pp. 1 –8. URL: `https://hal.archives-ouvertes.fr/hal-00848628`.

[13] M. Kučiš and P. Zemčík. "Simulation of Camera Features". In: *Proceedings of the 16th Central European Seminar on Computer Graphics*. Smolenice, SK: Technical University Wien, 2012, pp. 117 –123. ISBN: 978-3-9502533-4-4. URL: `http://www.fit.vutbr.cz/research/view_pub.php?id=10161`.

[14] *Matlab Simulink*. The MathWorks, Inc. URL: `www.mathworks.de/products/simulink`.

[15] M. Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[16] *Robot Operating System*. Open Source Robotics Foundation. URL: `www.ros.org`.

[17] J. Rossmann, N. Hempe, and M. Emde. "New Methods of Render-supported Sensor Simulation in Modern Real-time VR-simulation Systems". In: *Proceedings of the 15th WSEAS International Conference on Computers*. Corfu Island, Greece: World Scientific, Engineering Academy, and Society (WSEAS), 2011, pp. 358 –364. ISBN: 978-1-61804-019-0.

[18] J. Rossmann, N. Hempe, M. Emde, and T. Steil. "A real-time optical sensor simulation framework for development and testing of industrial and mobile robot applications". In: *Proceedings of 7th German Conference on Robotics (ROBOTIK 2012)*. VDE. 2012, pp. 1–6.

[19] J. Rossmann, C. Schlette, and M. Springer. "Kinematic robot control for a planetary landing mockup". In: *Proceedings of the Twentieth IASTED International Conference on Applied Simulation and Modelling ASM (accepted)*. 2012.

[20] C. Schlette. *Anthropomorphe Multi-Agentensysteme: Simulation, Analyse und Steuerung*. Springer-Verlag, 2013.

[21] C. Schlette, D. Losch, S. Haag, D. Zontar, J. Roßmann, and C. Brecher. "Virtual commissioning of automated micro-optical assembly". In: *Proc. SPIE Laser Technology and Industrial Laser Conference, Part of Photonics West (LASE)*. dx.doi.org/10.1117/12.2080742. 2015.

[22] C. Schlette, D. Losch, and J. Roßmann. "A visual programming framework for complex robotic systems in micro-optical assembly". In: *Proc. International Symposium on Robotics (ISR/Robotik)*. 2014, pp. 750–755.

[23] M. Schluse, C. Schlette, R. Waspe, and J. Roßmann. "Advanced 3D Simulation Technology for eRobotics". In: *Proc. IEEE International Conference on Developments in eSystems Engineering (DeSE)*. 2013, pp. 151–156.

[24] *Simscape Multibody*. The MathWorks, Inc. URL: `www . mathworks . de / products / simmechanics`.

[25] N. Wantia, D. Losch, and J. Roßmann. "Virtual commissioning and symbolic planning of micro-optical assembly processes". In: *Proc. Computer Science and Electronic Engineering Conference (CEEC)*. 2015.