Virtual Bin Picking
A Generic Framework to Overcome the Bin Picking Complexity by the Use of a Virtual Environment

Adrian Schyja[1] and Bernd Kuhlenkötter[2]
[1]*Institute for Research and Transfer, Dortmund, Germany*
[2]*Institute of Production Systems, TU Dortmund University, Dortmund, Germany*

Keywords: Bin Picking, Simulation, DirectControl 3, AutomationML, SmartComponents.

Abstract: Bin Picking is a very popular topic in the scope of robotic applications. For many years, R&D facilities as well as the industry work on Bin Picking solutions. However, it is challenging to bring such systems into industrial shop floors mainly due to the design and economical calculability accompanied by the acceptance of stable Bin Picking systems without any downtime. This paper presents a versatile interface-based framework for the planning, designing and in particular for the simulation of various Bin Picking applications. For that, the term 'Virtual Bin Picking' has been introduced, which associates the simulation of Bin Picking scenarios in a virtual environment without the need for hardware components. Thus, it enables the design of Bin Picking work cells and it allows to predict the quality of such cells in an early virtual commissioning stage.

1 INTRODUCTION

In today's automated factories, robot based grasping tasks are essential and the associated constraints need to be overcome in different situations. Once the work object or its position is not known exactly, the grasping system – consisting of a robot and its gripper – must be extended by suitable (vision) sensors and appropriate software. The latter includes, in particular, the recognition and localization of the work pieces with strategies to compute collision-free paths needed for the movement of the robot. A typical example of such applications is the removal of disordered components from a bin, well known as Bin Picking. Such handling processes are used in metal working industries, in the field of logistics for commissioning or in food industry. In many cases, an early conclusion on the feasibility of a Bin Picking solution, especially the achieved cycle times and process reliability without costly and time-consuming real experiments in advance, are impossible. As a result, such designed and optimized systems are often specifically tailored to one or just a few specific tasks making them and thus very inflexible in response to changing requirements.

Comparing Bin Picking systems to conventional solutions such as huge feeding systems, one advantage is the adaptability and hence the ability to handle different work pieces with only one system. The set up times are reduced when changing the product, since there is no reconstruction of the overall system necessary, only a reconfiguration, which in turn minimizes costs. With such systems, heavy or harmful components can be handled as it is currently performed by manual operators consequently exposing them to dangerous situations.

This paper is structured as follows. The next subsection describes the current challenges and the need for a virtual Bin Picking system. Section II outlines both, the architecture and the implementation of the presented framework within a simulation system. The subsequent section illustrates the concept of virtual Bin Picking. Section IV presents the idea of smart, virtual components which may be shared across heterogeneous engineering tools. Section V continues with the use of the developed framework and exemplary results are presented. Finally, section VI points out possibilities for further developments.

1.1 Related Work

Both research facilities and industry have been working on the Bin Picking topic for several decades. Developments during recent years have shown that a technical realization of Bin Picking for specific families of work pieces is achievable. Nevertheless, no

Schyja A. and Kuhlenkötter B..
Virtual Bin Picking - A Generic Framework to Overcome the Bin Picking Complexity by the Use of a Virtual Environment.
DOI: 10.5220/0005011401330140
In *Proceedings of the 4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2014), pages 133-140
ISBN: 978-989-758-038-3

prevalence rate of Bin Picking systems in industrial environments has succeeded so far. One reason for this circumstance is that the realized cycle times of available systems do not meet industrial requirements for economically feasible Bin Picking systems. Furthermore, such systems are often very complex, so that particularly commissioning and configuration require a lot of expertise on one hand. On the other hand, it takes a disproportionate amount of time limiting the economic benefit when it comes to a change of work piece settings.

In the past, many research institutions have been dealing with the development of Bin Picking systems. First attempts to detect and unload work pieces by assistance of robots were made in the mid '70s. Here, however, no bins were used as the components were located on conveyor belts (Tsuboi and Inoue, 1976; McKee and Aggarwal, 1977). In the following decades, more sophisticated technology concerning sensors, grippers and robots was available and the subject of Bin Picking has been addressed in various works (Ghita and Whelan, 2003; Schraft and Ledermann, 2003; Kirkegaard and Moeslund, 2006; Leonard et al., 2007; Böhnke, 2007; Ghita and Whelan, 2008). Most of the named papers present their work in the context of Bin Picking systems and reflect specific subdomains, e.g. object recognition and pose detection (Palzkill et al., 2010; Böhnke and Gottscheber, 2010; Wurdemann et al., 2011), path planning (Schyja et al., 2012; Johnson, 2013), gripper optimization and determination of best grasping positions (Francois et al., 1991), vision sensors (Pochyly et al., 2012), among others. Since they are part of research activities most of the presented solutions are shown on laboratory or prototype level. There are even some commercial systems available off the shelf, however, testing for suitability and configuration is often time consuming and requires availability of real components.

1.2 Motivation

Currently, there are no software tools available, which may be used for designing and verifying entire Bin Picking systems. Nor are there any for systematic determination of necessary system components in an early engineering phase complicating the system's planning. Even the selection of suitable components in terms of vision sensors, robots or grippers requires a high level of knowledge. Due to the complexity and amount of possible solutions this results in a major challenge. Additionally, the selected solutions may actually only be verified on a real implementation, which is time consuming and sconsequently not efficient. For this reason, it might be too soon to reach a definite conclusion about general general suitability of required components, potential cycle times or the degree of emptying the bin.

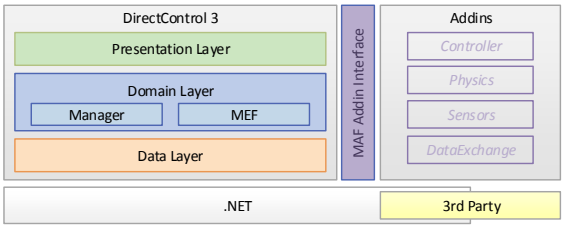


Figure 1: Simplified representation of the three-tier architecture of the DirectControl 3 framework.

Due to these shortcomings, we present an engineering tool, which enables virtual planning and simulation of various Bin Picking scenarios allowing optimization of such a system before real commissioning.

2 SIMULATION FRAMEWORK

The DirectControl 3 (DC 3) application framework, a successor of (Kneupner, 2004), has integrated the approach of simulating Bin Picking. DC 3 is a modular framework focusing on the simulation of industrial robot-based processes and applications (Schyja et al., 2012). It provides an application programming interface (API) which offers access to internal functionalities and is based on the Microsoft .NET framework. A simplified architecture of the software is depicted in figure 1. This framework follows a rich client platform (RCP) paradigm and provides a core with lots of reusable software modules and mechanisms. These modules are quite generic and offer important key functionalities, such as user interface, project management, neutral data structures and manager classes in terms of singletons making them accessible at any time. The complete design is based on the Microsoft Windows Presentation Foundation Framework (WPF), which offers an efficient separation of model and view, known as the Model-View-Controller pattern. The framework is designed to be completely independent of manufacturer, which is realized by the use of generic interfaces and an appropriate inheritance hierarchy of data structures. The latter is a primary key feature of this framework and offers the capability to adapt into different application domains.

The fundamental idea of DC 3 is to offer a core framework, the DC 3 runtime platform, which then may be extended via additional software components. To achieve that, DC 3 provides techniques for ex-

tending the framework in two ways. Both utilize a .NET based extensibility framework to support custom components as well as user defined Addins.

The former technique provides functions to automatically register and discover any type of components within the core application. Primarily it is based on the Managed Extensibility Framework (MEF) and consists of two major sections: the components, which are exported and any remote station, which discovers and imports the exported parts. To simplify the usage of MEF a generalized composing manager is provided which may be inherited and reused for user defined parts. This is usually performed using interfaces. As an example, the core framework provides a scripting manager, which imports any published or exported script based on a specific interface class. This registration is done completely in the background, while the programmer only has to provide some additional metadata about the components. If needed, they will be added to the user interface as well, depending on their metadata. In this way, it becomes very easy to enrich the core with additional components and functionalities. This technique works even if the exported components are located in outsourced assemblies, which yields to the second mechanism to extend the core framework. Based on the Managed Add-in Framework (MAF), a robust Addin architecture is provided. Using this custom extensions may be loaded or unloaded at runtime, which have access to internal structures of DC 3 at runtime. Addins may reference – among core functionalities – other Addins and may specify the order of loading sequence, which is quite important within the scope of dependency. The overall lifetime of any Addin is managed by a responsible part of the core framework.

Although the framework is intended for simulation purposes, it allows integrating custom project types as well. Each project type may specify its custom user interface arrangement, any views and windows, and so on. In this manner we extended the framework by a Bin Picking project, which offers a toolbox for doing Bin Picking simulation.

3 SIMULATION OF BIN PICKING

With DC 3 a general extensible framework with a lot of basic functionalities exists. Additionally, an application programming interface (API) is provided for the development of any types of extensions. For enabling Bin Picking simulation, various additional extensions by means of Addins are needed. The concepts and implementation will be presented in the next sections. The result will be a configurable Bin Picking simulation framework.

3.1 The Environment Model

The first significant extension developed is a simulation framework that enables a complete environment simulation including path and motion planning. A common way of most 3D applications for spatial representation of the graphic components is a strong coupled integration of a scene graph. To keep the general idea of a neutral implementation, we pursue an alternative approach by separating the structure of the environment into a so-called environment model and a loosely coupled graphical representation. The environment model represents the scene without a graphical representation. This strongly follows the separation of concerns paradigm and enables running a model-based simulation including all features like collision detection without the need of visualization, which saves resources and computational time. Another advantage of the separation between the environment and the graphical visualization by means of a scene graph is the way of the internal organization of data structures. While a scene graph may use a nested tree representation, e.g. in case of a kinematic chain consisting of links and joints, the environment model may use a flat tree structure, which simplifies the presentation of the model within the user interface. Within the environment model any object like geometries or robots including joints and links are derived from a generalized frame object which represents a coordinate system. Each of these objects has an interface which may be used to connect a graphical representation.

For enabling a vendor independent robot simulation modelling of kinematic chains and the design of data structures for enabling a general path planning are matters of special importance. In the current development stage the framework supports different types of kinematic representations. First, they may be coded and integrated into specific extensions. Secondly there is a general representation based on a customized XML format. Both types use the DH-convention (Denavit and Hartenberg, 1955) to represent the frame relationship of the links. Much more complex kinematic chains are enabled utilizing the COLLADA (version 1.5) file format, which is supported as well. The latter format becomes more and more important within the engineering tool chain since it is currently the only standardized file format which supports kinematics (Kuhlenkötter et al., 2010). Any of the representation types is integrated via a universal interface and passed through a kine-

matic compiler, which is responsible for the transformation into an internal representation based on the environment model. Thus, different robots are supported without any binding to a specific manufacturer.

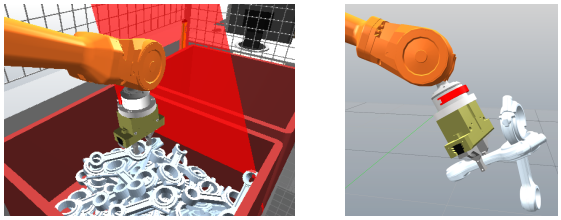
3.2 Multi-Engine 3D Visualization

Like many other engineering tools a 3D visualization of a simulation is essential, not only to get a visual feedback but also for the verification. Based on the environment model an arbitrary visualization engine may be connected to the model. We introduced a general interface, which allows integrating different rendering engines or even powerful geometry libraries like graphical kernels. Each engine may register itself via the MEF technique within the DC 3 core framework and the user may select and switch between different engines. Additionally, each engine can register their supported file formats. Based on this conception, three different engines were realized. The former uses the MOGRE rendering framework, a .NET port of the open source Object-Oriented Graphics Rendering Engine. The internal scene organization is divided into two sub-graphs, the internal and the visual one. The internal scene is used to render temporary items which are not part of the environmental model. Any environment related object is part of the visual scene, which mainly consists of transformable nodes and leaves with attached triangular meshes. Each node is connected and synchronized with the environment item by the use of the mentioned interface.

In addition, we integrated a pure WPF based rendering engine as well as a commercial graphical kernel using the same procedure by means of connecting to the environment model. The latter provides a rendering engine among common 3D CAD file formats, so that the range of functions of the developed framework is expanded enormously. The complete integration of 3D visualization is realized in a multithreaded and thread safe manner to achieve maximum performance.

3.3 Path Planning and Collision Avoidance

Unloading parts from the bin requires computation of valid path starting from a home position moving into the bin. After grasping the work piece the movement towards a release position must be determined. During path planning, apart from kinematic reachability, the collision avoidance is an important factor. For guaranteeing a collision-free path, it is crucial to simulate the steps of grasping a work piece and its movement throughout the execution of a path (figure 2). Furthermore, physics simulation comes in once a realistic grasping simulation of work pieces is required to determine, for example, cycle time, throughput, the degree of emptying or any undesirable situations as depicted in figure 2.



(a) (b)

Figure 2: (a) Simulation of grasping parts out of a bin, (b) simulation and analysis of failures, e.g. nested parts using physic simulation.

In a similar manner compared to the embedding of rendering engines, an interface for physics simulation was integrated. Since the most physics engines already entail a collision detection processing unit, we concentrate on the specification of a physics engine interface. Of course, individual collision detection libraries without support for physics simulation may be used that way, however, without physics. The interface is designed to enable a connection to the environment model. Thus, it is synchronized with the simulation engine. Compared with the integration of visualization, in this case, bidirectional synchronization is necessary. Using this interface in combination with the environment model, different loosely coupled physics and collision libraries were integrated, including Open Dynamics Engine (Drumwright et al., 2010), RAPID (Gottschalk et al., 1996) and D-Collide (Project Group 510, 2008).

3.4 Realistic Simulation

The target poses of a robot path are connected through many interpolated poses in between, which then must be reached by the end effector of the robot. During path planning the robots motion must be considered, which means that for each interpolated pose joint parameter needs to be calculated. Latter refers to the inverse kinematics computation. For that, several approaches exists, such as algorithms based on geometrical constraints, Jacobian matrices, or evolutionary algorithms. A considerable development in the past was the specification of an interface for realistic motion control of robots, namely the RRS-Interface (Realistic Robot Simulation) (RRS-Owners, 1991).

Based on this specification, a neutral interface was developed enabling the integration of different, man-

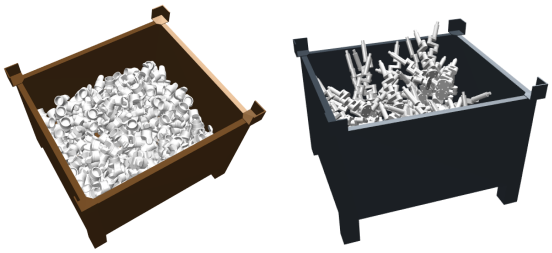
ufacturer specific motion controller. Through the concept of this interface it is possible to integrate different third party motion algorithms such as RRS modules, frameworks like Robot Operating System (ROS) (Martinez and Fernndez, 2013), OpenRave (Diankov, 2010) or even to develop custom motion controller to solve at least the forward and backward calculation. Each implementation of a motion controller may be outsourced into separate assemblies which then are registered via the MEF framework into the core of DC 3. By the use of additional metadata any motion controller informs the core framework which type of robot it supports. While the interface consists of low level programming calls, there is additionally a high level implementation, the virtual controller object. Each virtual controller has a relation to a single so-called *MotionController* interface which in turn controls motion of one or more robots. Through the concept of separation we may switch between different controllers and use a more precise or realistic motion controller via RRS or a less precise but faster in computation via a custom implementation.

3.5 Virtual Bin-Filling

In a real world scenario, initially there are empty bins, which are filled with work pieces for any further processing. For a nearly realistic simulation of Bin Picking the virtual bin must be filled within the simulation environment. The virtually filled-up bins serve as input information for a subsequent simulation of Bin Picking and, in particular, for virtual part localization. To be able to optimize for instance path planning strategies, a realistic behaviour is essential. Therefore the bin must be filled randomly with work pieces as precise as possible.

One solution may be a brute-force algorithm based on the bounding box of the bin and the dropped parts itself. The first part will be randomly positioned within the bin and in case of any collisions between the part and the bin including parts it will be randomly repositioned. Then next part will be added to the collision set and a new collision-free position within the bin will be calculated. These steps are repeated until a predefined number of parts is reached. The results are suboptimal, since gaps between parts may occur which leads to a less realistic placement. A more promising approach is reflected in the utilization of physics engines, as described above.

For the randomized, virtual bin filling a further extension was developed, which enables a more realistic simulation. Through a graphical user interface, the CAD models of the bin and the work piece as well as additional parameters such as the maximum number



(a) (b)

Figure 3: Virtually filled bins with different automotive parts using physics simulation.

of components or physical properties are configured in a first step, followed by the simulation of dropping parts into the bin. Based on the integrated 3D simulation as well as the developed modules for physics simulation, instances of the work pieces are created and placed at a predefined height over the bin. Next, they are falling into the bin using physics simulation, until a specified maximum amount is reached. Then, the scenario and particularly the positions of the work pieces may be exported into a file or database and used for further simulation of Bin Picking. This way a lot of different scenarios may be set up and used for virtual path planning as well as optimization. A further advantage is the possibility to determine cycle times or the degree of emptying. Figure 3 shows some results of filled bin with different work pieces based on physics simulation.

In addition to the virtual bin filling, we are also able to replicate real world scenarios. Within the research project AGMASS a Bin Picking cell was developed (Bücker et al., 2011), which uses the developed framework for doing the path planning and for the communication and synchronization of all hardware components. While the process is being executed, a database collects all relevant information for later investigations in the virtual environment. Thus it is possible to consider deadlock situations or situations, where parts interlock. To prevent those situations they may be optimized within the virtual environment.

4 REUSABLE VIRTUAL COMPONENTS

A work cell such as Bin Picking consists of different system components. For a realistic simulation it is important to have information about the components available in the virtual environment. Currently, it is common to exchange CAD models through different standardized file formats, but in general much more information is required, in terms of physical or dy-

namical properties, electrical information, kinematics, etc. In the past, there was no comprehensive file format, which was able to cover all needed information. With the release of AutomationML (Automation Markup Language) a data exchange format based on XML exists for the first time, which is able to store any relevant engineering information. AutomationML is an open and standardized specification for characterization and exchange of information on plant topology, geometry, kinematics, electrical signalling and further information in an object oriented way, making it very powerful. Since it has been already used in different engineering tools, like the robotics programming and simulation system ABB RobotStudio (Kuhlenkötter et al., 2010), its importance increases more and more.

In order to simplify the exchange of information and in particular product data we develop the so-called *SmartComponents* within the conexing research project (Schyja et al., 2014). Through SmartComponents, a concept is presented for exchanging virtual representations of real components between heterogeneous engineering tools based on AutomationML. Beyond CAD data they include all engineering related information. However, for a consistent simulation, the functionality of such components is also necessary. In case of a gripper, the functionality reflects the simulation of opening, closing or grasping any object. Another example is a sensor like a light barrier, detects obstacles. Since it is not desirable to implement these features in every engineering tool from scratch, we are currently working on a generic approach to integrate black box capabilities into SmartComponents in terms of secured third party libraries. For this we are developing formal definitions of methods and extend therefore the AutomationML specification. Additionally, to access i. a. black box functionality and its data across different tools a development kit is realized.

This approach will result in two possible scenarios: if the target software provides a native support of a given component or its functionality, this native features will be used and only missing information from the SmartComponent are extracted. Conversely, the software may have no native support. In the latter case the information and functionality contained in the SmartComponent is used for the simulation. This enables applications that even do not have native support for a component to use that exported functionality. To prevent an unwanted access to any sensitive data or licensed functionalities, a possibility of encryption will be offered to SmartComponents.

In the current state of development, we customized a 3D CAD design software using the development kit and developed interfaces for importing and exporting SmartComponents within that tool. This enables the export of work pieces including physical information needed to perform a realistic physics simulation within DC 3. This has also been extended for loading and saving SmartComponents. Thereby it is easy to do a Bin Picking simulation on almost any work piece.

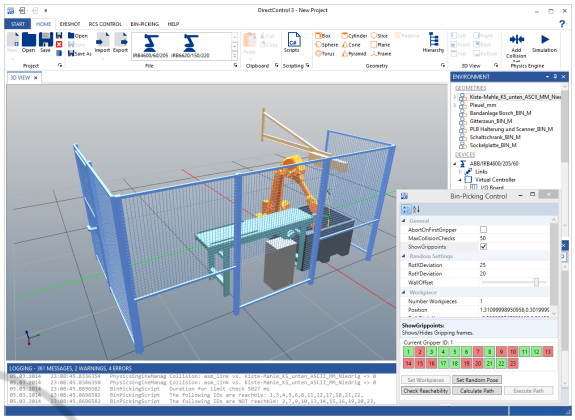


Figure 4: Virtual Bin Picking scenario based on a real work cell, which is used for simulation.

5 SIMULATION RESULTS

Simulation and verification of different Bin Picking scenarios has become possible due to the presented framework. What has been challenging so far is now optimized with respect to Bin Picking as in the design and layout a work cell.

With the real Bin Picking cell, a lot of data has been recorded having been feed into the developed virtual Bin Picking system. Figure 4 depicts a virtual copy of the existing Bin Picking cell. This application is the result of different cooperating Addins as described in the previous sections and was used for different experiments. Valid paths are calculated on configured gripping frames as they are in a real setup. Any peripheral equipment is considered when calculating paths and, in case of any collision, alternative paths are calculated. The complete computational and the simulated time is stored for analysis purpose as well as for cycle time detection. Since RRS is used, the determined times are close to the real ones qualifying them to be reliable.

In fact, data from deadlock situations were particularly interesting. While the layout of this cell was fixed, we used this data to optimize both path planning and the design of the gripper. At the current stage, we do not provide any sensor simulation needed for virtual object localization. Hence, we need

Algorithm 1: Simulation of Bin Picking

```
1. fill virtual bin by physics or real data
2. compute grasping positions or import real one
3. sort parts by center of gravity in Z direction
while parts in bin do
   take n top most parts (n is a random number)
   choose one of them randomly
   for all grasp id do
      if reachable and no collision then
         compute motion
         generate path
         execute and simulate unloading part
         remove part from pile
      else
         reject part (remember for statistics)
         continue
      end if
   end for
end while
```

to emulate the part localization for simulating Bin Picking. The complete procedure for simulation is presented in algorithm 1. Firstly, the bin is filled using virtual or real data. Instead of calculating any grasping positions real ones may also be used. To emulate localization, the parts are sorted by their center of gravity along the Z axis. Thus, the top most parts are accessible. As long as there is either a part in the bin or no parts to be unloaded, any top most part will be considered. Next a path is generated for each reachable and collision free grasping position.

The overall simulation takes both the simulation of the gripper's behavior, opening, closing and attaching the work piece into account. This is realized based on real conditions. The virtual controller is connected with a virtual I/O module, which provides digital and analog signals. The gripper attached to the robot is connected through a signal to the I/O module and appropriate instructions are added to a path generated for each part. The resulting robot program is close to a program, which may run on a dedicated robot, including motion and action instructions. As a result, lots of information are acquired. These include duration times of calculations and collision checks as well as the (real) elapsed simulated time, which is provided by the RRS module. We are able to analyse individual parts and determine e. g. cycle times per part up to a prediction for a complete bin, although no times for part localization are currently considered. Hence the resulting time is the minimum duration for computation of valid paths and for motion of the robot, which is quite a step forward to an overall simulation of Bin Picking.

6 CONCLUSIONS

In this paper an approach for virtual planning of Bin Picking applications was presented. Since there are currently no tools available, which allow an overall virtual planning and process simulation as well as optimization of the entire design of such a robot based work cell, it is verified nowadays in a real setup, leading to inacceptable downtimes. To overcome this, a framework which allows process simulation and optimization of various Bin Picking scenarios without the need of a real setup was developed. The developed solution was realized in form of a DirectControl 3 extension. In future works, we seek to achieve a complete Bin Picking simulation including vision sensors and virtual object localization. Therefore, we want to extensively use SmartComponents in order to intensify integration as well as the exchange of virtual components like virtual controllers or robots with other engineering tools, as described in (Bartelt et al., 2013). Once virtual vision sensors will be made available through SmartComponents, they will be integrated into the overall framework. In this way, localization algorithms may be tested in advance. Using the presented framework, it is possible to calculate various configurations in a short time.

ACKNOWLEDGEMENTS

The research and development project conexing is funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept "Research for Tomorrows Production" and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication. The authors would like to thank the anonymous reviewers for their helpful comments and suggestions.

REFERENCES

Bartelt, M., Schyja, A., Kuhlenktter, B., and Benkner, T. (2013). Interdisziplinre zusammenarbeit in einer heterogenen cax-software-landschaft. *PRODUCTIVITY Management*, 3/2013.

Böhnke, K. (2007). Object localization in range data for robotic bin picking. In *International Conference on Automation Science and Engineering*, CASE, pages 572–577.

Böhnke, K. and Gottscheber, A. (2010). Fast object registration and robotic bin picking. In Gottscheber, A., Obdrzalek, D., and Schmidt, C., editors, *Research and Education in Robotics – EUROBOT 2009*, volume 82

of *Communications in Computer and Information Science*, pages 23–37. Springer Berlin Heidelberg.

Bücker, M., Krewet, C., Schyja, A., and Kuhlenkötter, B. (2011). Autonome Roboter mit sensorbasierter Bahnplanung. *Industrie Management*, 1:21–24.

Denavit, J. and Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Transactions of the ASME. Journal of Applied Mechanics*, 22:215–221.

Diankov, R. (2010). *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute.

Drumwright, E., Hsu, J., Koenig, N., and Shell, D. (2010). Extending open dynamics engine for robotics simulation. In *Proceedings of the Second International Conference on Simulation, Modeling, and Programming for Autonomous robots*, SIMPAR'10, pages 38–50, Berlin, Heidelberg. Springer-Verlag.

Francois, C., Hebert, M., and Ikeuchi, K. (1991). A three-finger gripper for manipulation in unstructured environments. In *IEEE International Conference on Robotics and Automation*.

Ghita, O. and Whelan, P. F. (2003). A bin picking system based on depth from defocus. *Machine Vision and Applications*, 13(4):234–244.

Ghita, O. and Whelan, P. F. (2008). A systems engineering approach to robotic bin picking. In Bhatti, A., editor, *Stereo Vision*, chapter 4, pages 59–72. InTech.

Gottschalk, S., Lin, M. C., and Manocha, D. (1996). OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA. ACM.

Johnson, M. (2013). Flexible Path Planning For Bin-Picking Applications. Masterarbeit, Technische Universitt Dortmund.

Kirkegaard, J. and Moeslund, T. B. (2006). Bin-picking based on harmonic shape contexts and graph-based matching. In *18th International Conference on Pattern Recognition*, volume 2 of *ICPR*, pages 581–584.

Kneupner, K. (2004). *Entwicklung eines Programmier- und Steuerungskonzepts für Robotersysteme auf der Basis eines Umweltmodells*. PhD thesis, TU Dortmund University. ISBN 3-18-338920-7 VDI Verlag, Düsseldorf.

Kuhlenkötter, B., Hypki, A., Schyja, A., and Miegel, V. (2010). Robot Workcell Simulation with AutomationML Support - An Element of the CAx-Tool Chain in Industrial Automation. In *Proceedings for the joint conference of ISR 2010 (41st International Symposium on Robotics) und ROBOTIK 2010 (6th German Conference on Robotics)*. VDE Verlag GmbH.

Leonard, S., Chan, A., Little, J. J., and Croft, E. A. (2007). Robust motion generation for vision-guided robot bin-picking. *ASME Conference Proceedings*, 2007(43033):651–658.

Martinez, A. and Fernndez, E. (2013). *Learning ROS for Robotics Programming*. Packt Publishing.

McKee, J. W. and Aggarwal, J. K. (1977). Computer recognition of partial views of curved objects. *IEEE Transactions on Computers*, 26(8):790–800.

Palzkill, M., Ledermann, T., and Verl, A. (2010). Anticipation-preprocessing for object pose detection. In *41st International Symposium on Robotics*, ISR 2010, pages 440–445.

Pochyly, A., Kubela, T., Singule, V., and Cihak, P. (2012). 3d vision systems for industrial bin-picking applications. In *15th International Conference on Mechatronics*, MECHATRONIKA, pages 1–6. IEEE.

Project Group 510 (2008). Entwicklung einer echtzeitfhigen Kollisionsbehandlung fr die physikalische Simulation in virtuellen Umgebungen. Project Group, TU Dortmund University, Computer Science VII.

RRS-Owners (1991). Realistic Robot Simulation Interface Specification, version 1.3. Technical report, Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), Berlin.

Schraft, R. D. and Ledermann, T. (2003). Intelligent picking of chaotically stored objects. *Assembly Automation*, 23(1):38–42.

Schyja, A., Bartelt, M., and Kuhlenkötter, B. (2014). From conception phase up to virtual verification using automationml. In *5th CATS 2014 – CIRP Conference on Assembly Systems and Technologies*. Status: accepted.

Schyja, A., Hypki, A., and Kuhlenkötter, B. (2012). A modular and extensible framework for real and virtual bin-picking environments. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5246 –5251.

Tsuboi, Y. and Inoue, T. (1976). Robot assembly system using tv camera. *Industrial Robot: An International Journal*, 3(2):67–72.

Wurdemann, H. A., Aminzadeh, V., Cui, L., and Dai, J. S. (2011). Feature extraction of non-uniform food products using rgb and rgb-d data combined with shape models. In *International Conference on Robotics and Biomimetics*, ROBIO, pages 1652–1657. IEEE.