Online simulation for flexible robotic manufacturing

Christina Petschnigg, Guido Breitenhuber, Benjamin Breiling, Bernhard Dieber, Mathias Brandstötter
JOANNEUM RESEARCH
Institute for Robotics and Mechatronics
Klagenfurt, Austria
{firstname.lastname}@joanneum.at

***Abstract*— From cutting shapes out of metal or wood to welding, screwing or lifting heavy objects, the application and versatility of robots in industry is vast. Their programs however are typically very static and only used in pre-defined, static settings, thus they cannot deal with changing production tasks. Whenever a special problem has to be solved several times with different parametrizations, transferring simulation results to real-world applications is desirable in order to reduce the time investment necessary for program adjustments. The current paper discusses the major challenges in including powerful simulations as a tool to improve robot programs at runtime. Further, an evaluation of the Virtual Robot Experimentation Platform (V-REP) with respect to its applicability to direct robot control via simulation is presented.**

Keywords- Robot Simulation; Flexible production;Robot Control; Path Planning;

I. INTRODUCTION

In today's industry the trend shifts from manufacturing mass products towards more individualized offers, where consumers have the possibility to customize the products they want to buy. Hence, production systems have to be flexible and easy to adjust to varying needs during the manufacturing process. Imagine a robot cutting metal or wooden pieces of different shape. Planning paths dynamically typically exceeds the capability of normal industry robots. First, movements are typically hard-coded. Second, the workplaces may lack the necessary sensors required to perceive the environment in a way to dynamically detect work pieces and obstacles. Third, the description of the next task to perform must be transmitted and processed online. Thus, the typical way is to outsource those calculations to an external system like a manufacturing execution system (MES). Still, this MES needs to perform complex planning tasks for robot trajectories including obstacle and collision avoidance depending on the current task and workspace setting. Thus, extending the MES with a dynamic simulation component bears many advantages. In order to keep the time investment for switching between different shapes low, a simulation approach that generates welding or cutting trajectories automatically based on a CAD model of the corresponding workpiece is desirable. Deploying the movement instructions directly from the simulation on the robot drastically reduces switching times. Further advantages of simulation tools may be exploited like faster design of scenes, easier testing of use cases and above all the incorporated trajectory planning features. The applicability of the simulated trajectory to actual problems, however, has to be checked carefully. In this context NIST proposed standardized test methods in order to verify the validity of the robot model [1].

Trajectory planning, i.e. the description of robot motion through space with respect to time, is a mathematically well-solved problem. The major steps of the process are computing a sequence of positions, velocities and accelerations. There are several algorithms ready to use for path planning, collision detection and associating time with the path. The Virtual Robot Experimentation Platform (V-REP) [2] is a robot simulator providing trajectory planning functionalities. In the context of this paper, we present an approach to access a V-REP motion planning scene from outside using a remote API and directly apply the simulation results to the control of a real-world robot.

The remainder of this paper is organized in the following way. Section II covers related work on the topics of robot simulation and robot control. In Section III we point out major challenges when using simulation results in control applications. Section IV discusses our approach to overcome the challenges. In Section V we demonstrate the applicability of the presented approach to the control of a UR10 and Section VI concludes this paper.

II. RELATED WORK

Simulation has played an important role in robotics, industry and research. It is a quick, easy and especially safe method for testing new algorithms and concepts. Basically every robot manufacturer provides its own simulation tool for offline planning (OLP) that is supplied alongside the robots, however, there are also manufacturer independent ones. While many works focus on the description of different simulation tools and using them in production planning, to our best knowledge there is hardly any information on using V-REP [3] for this purpose.

RoboDK [4] is a manufacturer independent programming and simulation environment that allows transferring the simulations directly onto the dedicated robot after the code has passed a series of tests. Applications can be implemented on a virtual robot controller that may be used for simulation. Robot drivers are available which allow for online programming, i.e. the robot is moved while being simulated. However, one serious downside of RoboDK is the missing path and trajectory planning functionality. This makes simulated scenes inflexible. In [5] the behavior of industrial robot manipulators is simulated with the help of RoboDK. Gazebo is another 3D simulation tool that is presented in [6]. It provides a virtual framework to evaluate the behavior of multi-robot teams. Simulations are based on customizable models of the robots' actual workspace. Another open source simulation platform is the Urban Search and Rescue Simulation (USARSim) which is described in [7]. This simulator may also be used for complex scenes like simulating a humanoid robot [8].

In [9] however, a programming system based on the simulation tool SolidWorks is presented. It includes aspects of robot and work space simulation, kinematics, dynamics, motion planning and automatic code generation for a welding application. Virtual Manufacturing [10] is an approach to reduce companies' time to market and improve overall production processes by building a synthetic environment. This virtual workspace is generated and managed with simulation and virtual reality tools. It allows following all the activities a product undergoes during its lifecycle and hence facilitates the prediction of problems and inefficiencies. A similar approach is the digital factory concept [11] where simulation of various planning levels (i.e. discrete event or motion simulation) plays a major role in enhancing production processes and product lifecycle management. It enables integration of CAD designs and the establishment of reusable product configurations. The notion of a Digital Twin [12] has been used by the U.S. military to describe the runtime co-simulation of cyber-physical systems.

III. Major Challenges

When designing such a simulation scheme there are several obstacles to overcome on the way. The biggest challenges when using simulation outputs in an automated fashion for robot control are described below.

1. In case of simulated path generation, a dynamically changing environment poses a stringent problem especially for collision detection. As these models do not take into account humans or other objects suddenly crossing the robot workspace, collisions may occur which endangers human safety. Hence, some form of feedback from the environment to the simulation would be necessary. However, for this paper we assume that the robot is placed within a dedicated cell that prevents human co-workers from entering the workspace. In any case, a simulation can never replace proper safety mechanisms as required according to safety standards.
2. Dynamic re-parametrization of the workpiece is difficult to realize during runtime. Due to a change in the underlying CAD model of the item to be manufactured, high flexibility is necessary to change simulations dynamically.
3. The result of a simulation can only be as good as the models used in their calculations. Thus, simulations must be built with the utmost care to according details relevant to the use-case.
4. Sometimes tasks demand fewer degrees of freedom than what are provided by the robot system. This redundancy must be resolved in some meaningful way. In case that the degree of redundancy is 1, a two-dimensional visualization helps to find a proper trajectory taking, e.g. static obstacles, joint limits and performance criteria into account, see Figure 1. This illustration is inspired by Robotmaster and the proposed method in [13]. The gradient depicted as the grey background symbolizes the value of a cost function.
5. When physically following a trajectory with the robot, the joint angles, velocities, accelerations and jerks must be within certain limits that are given by the manipulator itself. Due to the nonlinear mapping between the end effector and the joints the behavior of the end effector is challenging to pre-calculate.

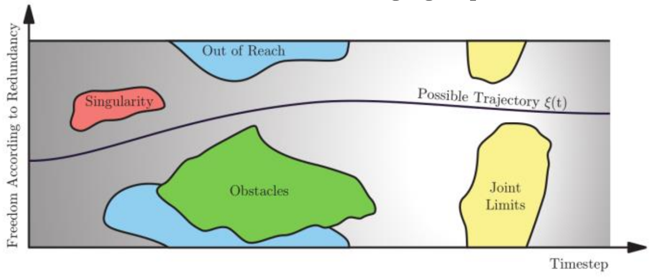

Figure 1 Qualitative representation of a graphical optimization tool (e.g. Robotmaster)

IV. Approach

In order to achieve a better understanding of how our simulation scheme works, this section describes the overall API architecture, used libraries and integration as well as dependency with external libraries. Further, a brief introduction to the functionality of the API client and the software design are presented.

Our goal is to online connect a dynamic production environment with a parametrizable simulation of this environment which can be used to calculate robot trajectories that are adapted to the current task and state of the environment. Thus, we represent the task essentials of the use-case in the simulation. The parameters to describe the specific instance of the scene are then dynamically fed into the simulation before performing path planning. This includes the size and pose of obstacles and workpieces, the current joint configuration of the robot and further elements which can be foreseen to be changing during production. It has to be noted however, that unforeseen circumstances will typically not be modeled a-priori. We assume that the basic settings of the environment and task to be performed are known at time of simulation creation.

A. *Software Architecture*

The simulation environment V-REP includes 4 different physics engines, out of which one may be chosen by the user for simulation. Additionally, it incorporates the OMPL path planning library [14] and the Reflexxes Motion trajectory planning library [15], which make it a convenient means for generating collision-free robot trajectories. Therefore, we chose V-REP to build our simulation scheme. The trajectory planning functionalities are realized on top of V-REP. For accessing simulation scenes from outside, there exists a remote API and a ROS (Robot Operating System) [16] interface. Further, it offers different types of integration possibilities, several of which are used for our trajectory planning functionalities.

The scene considered in this work consists of the robot, its environment and workpieces and is modeled within V-REP. The scene is extended with an embedded client script which implements the path planning functionalities. In order to access the path planning from outside V-REP an API client implementation calls the planning methods of the embedded client script. It uses the V-REP client API for communicating with V-REP. We implemented a Python version of the client API, which can easily be ported to other languages if needed. Figure 2 shows the V-REP integration parts.

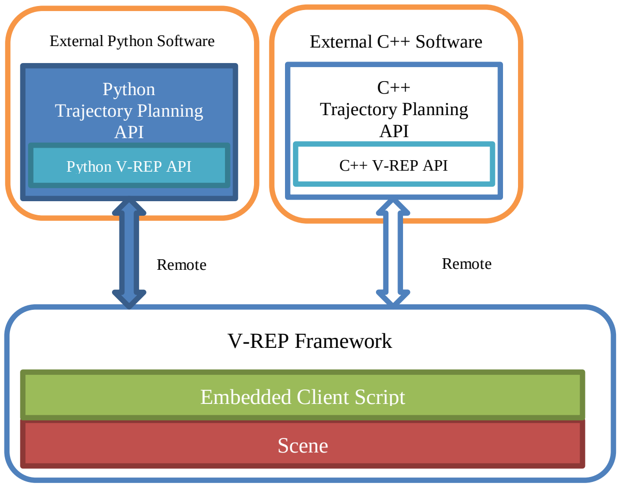

Figure 2 Layers of used V-REP integration possibilities

B. API Client Functions

The Python API is based on the standard V-REP API client provided by the simulation tool itself but extends its functionalities for trajectory planning on the one hand and simplifies access to existing functions valuable for path planning on the other hand. The Python API client's major capabilities are the following:

1) *Start and Connect to V-REP:* In order to plan a trajectory V-REP must be started and the API client connected. There are two possibilities how this may work. Either V-REP is started manually or the V-REP process is managed by the API Client. In the former case the API client simply connects to a running instance with a loaded simulation scene. In the latter case the API client starts a V-REP instance, loads a scene and connects to its simulator instance. When the simulation ends the client closes the simulator.

2) *Simulation Control:* Once connected, the API client can control the simulation state, i.e. start, stop and pause the simulation. In addition, the current simulation state and the status of the simulation's real-time mode can be retrieved.

3) *Object Manipulation:* There are methods that provide functionalities to retrieve existing objects from the scene and read or edit position as well as orientation properties. These methods are useful for retrieving and manipulating planning targets and obstacles in the scene.

4) *Path planning:* For path and trajectory planning, a so called "planning target" can be enqueued to a FIFO queue. Each planning target is a point in the scene consisting of Cartesian coordinates and an orientation representing the desired target pose of the TCP. The path planning engine is able to plan a path from the current TCP pose to the desired planning target considering all scene obstacles and the robot's joint limitations. Since multiple targets can be enqueued, a path containing several intermediate points may be generated. The API uses a state machine to represent the different steps required in path planning. Simplified, the process can be divided into three steps: enqueue a planning target, plan a path and move the TCP. A more detailed representation of the states and their transitions is shown in Figure 3. With no planning targets in the queue the API is in state "NoTarget" otherwise the state machine moves to "Ready". When calling the search path function the state moves to "PathSearching". In case a path is found the API is moving to state "PathFound" otherwise the new state is "PathNotFound". When a path was found and the move function is called the new state is "Moving" and the TCP is moved. After reaching the target the state is either "NoTarget" or "Ready" if another planning target waits to be processed. In any state except "PathPlanning" and "Moving" the next planning target can be dequeued.

5) *Further Functions:* The Trajectory Planning API Client is based on the official V-REP API Client. It is fully compatible with the standard API and all functions provided by the default API client can be used as well.

C. Detailed Design

V-REP's trajectory planning functionalities are residing in three different layers. A scene resides at the bottom, at least one client script in the scene and an API client for use in external software components on top. We decided to split the overall realization to these three parts to exploit the strength of each component.

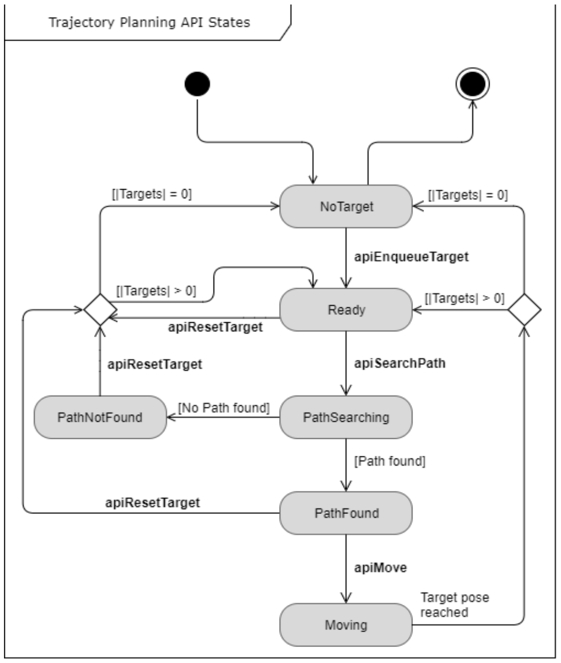

Figure 3 Trajectory Planning API States

1) V-REP Scene: A V-REP scene defines a model of the whole static physical simulation environment and its settings. This includes the robot, gripper, obstacles and other objects. It also defines properties of these objects, e.g. max angle or torque for robot joints. There is a vast amount of predefined object and robot models which can be used inside a scene. The user can also create new models which fit differing requirements if no predefined model exists.

2) V-REP Client Script: When properties of a V-REP scene cannot be expressed via the V-REP user interface and settings one can extend a scene using client scripts. We use this possibility to add the path and motion planning functionalities on the one hand and entry points for the API client on the other hand. These methods do not need to be modified for each scene but can mostly be reused in new scenes. One could modify the path planning script for tuning the calculation of the "best" path. This could be optimized with respect to the shortest path or any other optimization criteria. For our implementation we define the "best" path as the path with the shortest distance with respect to TCP movement.

3) API Entry Points: For controlling the planning functionalities from outside of V-REP via the API Client several API entry points are defined in the client script. They are responsible for calling the planning methods. Due to the fact, that calling a function using the API blocks the execution of other parallel running functions in the script the API Entry points are implemented using a lightweight state machine model. This state machine was described before in Figure 3. Each method tries to modify the state machine and returns immediately after doing so. A main threaded function listens to state changes and executes the internal long running procedures.

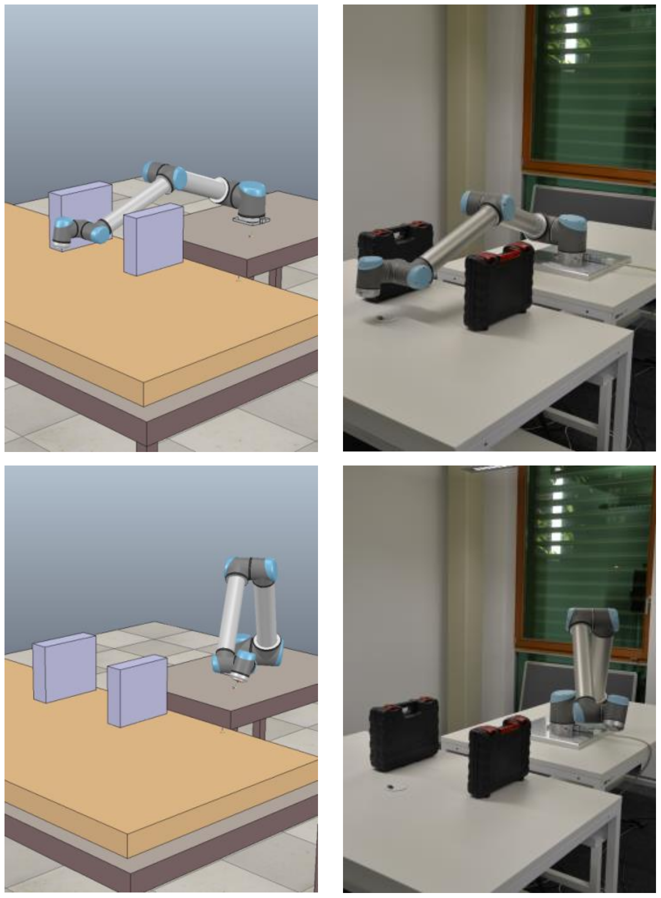

Figure 4 Correspondence of simulation and real application.

V. FINDINGS

For evaluation purposes we connect a Universal Robots UR10 robot to our online simulation environment. Depending on the current robot joint configuration we plan collision-free paths through a workspace with moving obstacles. Trajectory planning in the same context will be discussed in future work. Figure 4 displays the correspondence of the V-REP simulation and the real application. The pictures on the top show the robot in its initial position while the illustrations at the bottom show the robot in its target position. Note, that in this setting, no linear path exists between initial and target pose and thus, dynamic path planning is needed whenever the target or obstacles change in location.

In general, our API allows for full parametrization of the simulation at run time. Robot configurations, obstacle and workpiece poses as well as other pre-defined elements can be configured remotely. Our architecture also allows for distributed deployment of robot controller, computational nodes and simulation and thus, the application parts can be flexible deployed on different physical machines. We can use the full power of a robot simulation tool at run time and thus enable flexible production down to lot size one.

When configuring the simulation and its parameters, care needs to be taken to incorporate the properties of path planning algorithms. As we configured V-REP to use the RRT-Connect algorithm [17] which incorporates the notion of Rapidly-exploring Random Trees and therefore randomness to determine a collision-free path through the workspace, the time to find a path is not constant. Further, not every run of the simulation yields a feasible path due to the random nature of the path planning algorithm. Therefore, this approach is not suitable for real-time planning problems.

Additionally, simulation scenes have to be designed very carefully in order to ensure compatibility with the real workspace. The size and position of obstacles needs to be captured accurately in order to avoid collisions.

VI. Conclusion

Using simulation for direct robot control has been adopted in many different works of research over the years. In the present paper, we describe how the tight integration of a production system and a simulation and its path planning functionalities for the purpose of robot control can be used to dynamically adapt a robot's behavior to changing tasks. Our evaluation shows the effectiveness of using the internal scenic and path planning functions and sending the commands directly to the robot.

In future work we want to extract whole trajectories from the V-REP scene, as so far we only extracted path objects. This further facilitates robot control as robots' internal joint angles, velocities, accelerations and jerk limits are already taken into account by the planning tool. In addition, we will explore how scenes can be dynamically composed from various types of sensor data.

Acknowledgment

The work reported in this article has been supported by the Austrian Research Promotion Agency under grant nr. 849896.

References

[1] Pepper C., Balakirsky, S. & Scrapper, C. 2007. Robot simulation physics validation. In Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems (PerMIS '07). ACM, New York, NY, USA, 97-104. DOI=http://dx.doi.org/10.1145/1660877.1660890

[2] Virtual Robot Experimentation Platform, http://www.coppeliarobotics.com.

[3] Rohmer, E., Singh, S. P. N. , & Freese, M. "V-REP: A versatile and scalable robot simulation framework," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, 2013, pp. 1321-1326. doi: 10.1109/IROS.2013.6696520,

[4] RoboDk Software S.L: RoboDK Simulation and OLP for Robots; https://robodk.com/

[5] Erdei, T. I., Molnar, Z., Obianna, N.C., Gönczi T. & Husi G.; "Industrial KUKA Robot Manipulator in Simulation Environment and Position Read-back." https://www.researchgate.net/publication/311970929_Industrial_KUKA_Robot_Manipulator_in_Simulation_Environment_and_Position_Read-back

[6] Koenig, N. & Howard, A.; "Design and use paradigms for gazebo, an open-source multi-robot simulator." Intelligent Robots and Systems, 2004.(IROS 2004).Proceedings. 2004 IEEE/RSJ International Conference on Vol. 3. IEEE, 2004.

[7] Carpin, S., Lewis, M., Wang, J., Balakirsky, S. & Scrapper, C. "USARSim: a robot simulator for research and education." Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007.

[8] Greggio, N., Silvestri, G., Menegatti, E., & Pagello, E. (2007, March). A realistic simulation of a humanoid robot in usarsim. In Proceeding of the 4th International Symposium on Mechatronics and its Applications (ISMA07), Sharjah, UAE.

[9] Mitsi, S., Bouzakis, K. D., Mansour, G., Sagris, D., & Maliaris, G. (2005). Off-line programming of an industrial robot for manufacturing. The International Journal of Advanced Manufacturing Technology, 26(3), 262-267.

[10] Souza, M. C. F., Sacco, M., & Porto, A. J. V. (2006). Virtual manufacturing as a way for the factory of the future. *Journal of Intelligent Manufacturing*, *17*(6), 725-735.

[11] Kuhn, W. "Digital Factory - Simulation Enhancing the Product and Production Engineering Process," Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, 2006, pp. 1899-1906. doi: 10.1109/WSC.2006.322972

[12] Glaessgen, E. & Stargel, D. "The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles", 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Structures, Structural Dynamics, and Materials and Co-located Conferences, 2012.

[13] Zargarbashi S., Khan, W. & J. Angeles. The jacobian condition number as a dexterity index in 6R machining robots. Robotics and Computer-Integrated Manufacturing, 28(6):694–699, 2012.

[14] Şucan, I. A., Moll, M. & Kavraki, L. E. (n.d.). The Open Motion Planning Library. Retrieved November 09, 2017, from http://ompl.kavrakilab.org/

[15] Kroeger, T. (n.d.). Reflexxes – Online Trajectory Generation – Robotics- Home. Retrieved November 08, 2017, from http://www.reflexxes.ws/

[16] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software(Vol. 3, No. 3.2, p. 5).

[17] Kuffner, J. J. & LaValle, S. M. "RRT-connect: An efficient approach to single-query path planning," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, 2000, pp. 995-1001 vol.2. doi: 10.1109/ROBOT.2000.844730