PyBullet Industrial: A process-aware robot simulation

Jan Baumgärtner [1], **Malte Hansjosten**[1]**, Dominik Schönhofen**[2]**, and Prof. Dr.-Ing. Jürgen Fleischer** [1]

1 wbk Institute of Production Science **2** Independent Researcher

Summary

The trend towards individualized products and the increasing demand for a greater number of variants require a rethinking in the production engineering environment. In the context of this transformation, we see robots taking on more and more manufacturing tasks (Mühlbeier et al., 2021). The development of this field is hampered by a toolchain gap: While there are a large number of robot simulations and process simulations there is not yet a simple simulation environment that combines the two and allows the user to investigate the interplay of both. Process simulation in this case refers to the simulation of manufacturing processes which according to (Mourtzis et al., 2014) is defined as the use of one or more physical mechanisms to transform the shape and/or form of a workpiece. While process simulation is a vast field that studies different levels of detail, we focus on the simulation of how the process affects the environment and the robot.

To meet this challenge we developed PyBullet Industrial. This Python package extends the open-source multi-body physics package PyBullet with manufacturing process models to simulate manufacturing applications that add material, remove material or simply move material. A sample of concrete manufacturing applications in each category can be seen in Figure 3.

The package not only simulates the environmental effect of the processes but also the forces imparted on the robot. It also allows the dynamic switching of processes with the same robot corresponding to tool changes during the manufacturing process. The package also contains utility functions such as path builder classes which are based on G-code (also called RS274) interpolation schemes (Kramer et al., 2000) or a variety of drawing and visualization functions. A sample screenshot of a simulation using PyBullet Industrial can be seen in Figure 1.
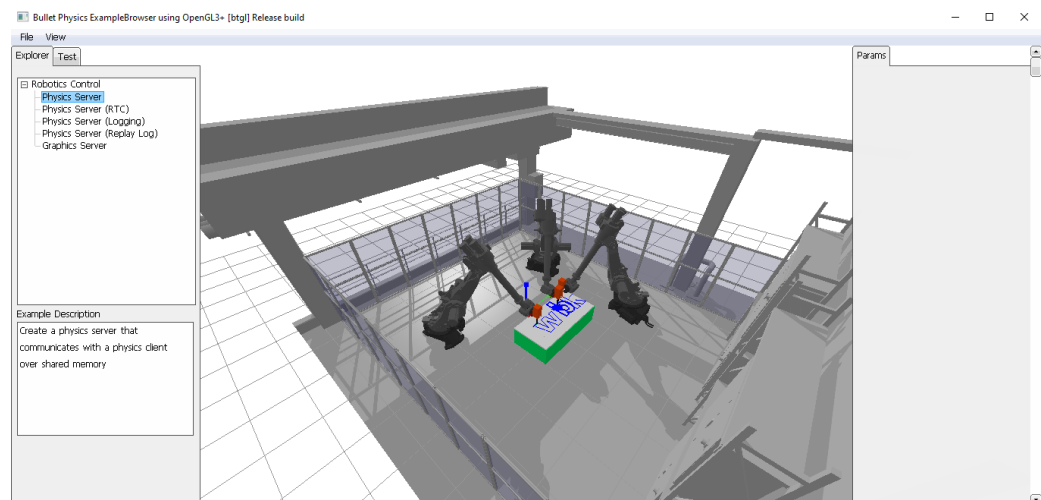
Figure 1: Sample view of Pybullet Industrial

Statement of need

PyBullet Industrial was developed for the interdisciplinary field of robot manufacturing. While there are a large number of simulation tools for robotics research such as Gazebo (Koenig & Howard, n.d.), CoppeliaSim (Rohmer et al., 2013), or webots (Michel, 2004), their capabilities all end at the robot end effector as they are unable to simulate manufacturing processes. In the same vein, there are several popular FE simulation tools such as Abaqus (Smith, 2009) capable of simulating process behavior. These simulations end at the tool as they are not meant to simulate the systems that move the tool. Since robots are now performing more and more manufacturing tasks studying and accounting for the interaction between robots and processes becomes ever more important. This requires a simulation that can simulate robots and processes.

PyBullet Industrial closes this gap by taking classical robot multibody simulations and extending them using simple process simulations which impact the environment. PyBullet Industrial is thus the first process-aware robot simulation platform built for research. Note that PyBullet Industrial neither aims to develop perfect process simulations nor robot simulations, it focuses on the interplay of both. Example applications of PyBullet Industrial are:

- Designing joint controllers that compensate for the large process forces during milling.
- Designing path planning algorithms for 3D printing that can detect if a robot combines with a previously printed object.
- Checking the coating of an object in complex scenarios where the object is moved by a robot while another one is spraying paint.

Overview

Robot simulations typically start at the base and stop at the end effector while process simulations typically start at the process and end where the tool is connected to the machine. PyBullet Industrial divides functionality similarly by employing a `RobotBase` class simulating the multibody dynamics of a Robot manipulator and an `EndeffectorTool` class capable of simulating processes. A sample simulation view with both objects can be seen in Figure 2.
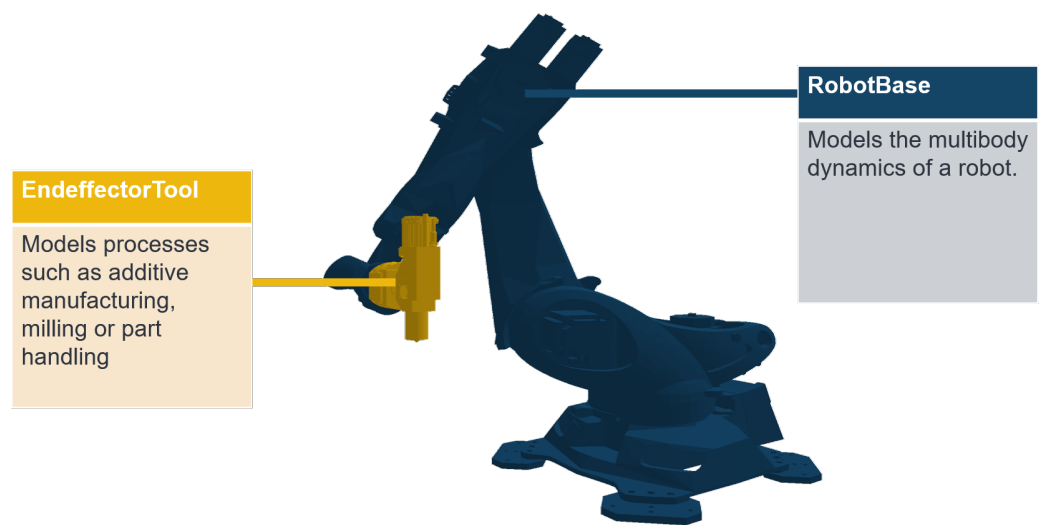
Figure 2: Overview over the two main Objects

These objects can be deployed into a standard PyBullet simulation environment and used to build manufacturing scenarios.

Robot objects

The RobotBase class builds upon PyBullet's URDF (Universal Robot Description Format) (Tola & Corke, 2023) import feature which allows the loading of dynamic multibody robot models. The class adds several convenient interfaces which allow the setting and measuring of joint and end effector states. This latter allows the user to reposition the end effector without worrying about the underlying kinematics. The list of interfaces includes:

- A joint state interface that allows the user to set and read joint positions, velocities, and torques.
- A end effector state interface that allows the user to set and read the end effector position, orientation, and velocity. The inverse kinematics is in this case calculated using PyBullets built-in inverse kinematics solver.
- A world state interface that allows the user to set and read the world position and orientation of the robot base.

End effector Tools

End effector tools are the main novelty of this library and implement various process models. An EndeffectorTool object can be coupled with a robot attaching it to the flange of the end effector. The tool provides a positioning interface that automatically calls the end effector interface of a coupled robot making it easy to reposition the tool center point in space.

Note that coupling and decoupling of tools can be done during runtime to simulate tool quick changes common in complex manufacturing cells. The geometry of a specific tools is defined by a URDF file where the last link is the default tool center point although this can be changed by the user.

While the base object implements the main interfaces and structure of the class, different process models are implemented as children of the EndeffectorTool object. These models can be grouped into three different categories according to how they interact with material as seen in Figure 3.

Figure 3: Classes of Manufacturing processes that can be simulated using this package

The adding of material is done using the `Extruder` class which uses raycasts (Roth, 1982) to spawn objects either on the surface of another object or at the end of the raycast. This is indicated in Figure 4, which also shows the extruder parameters that can be set.
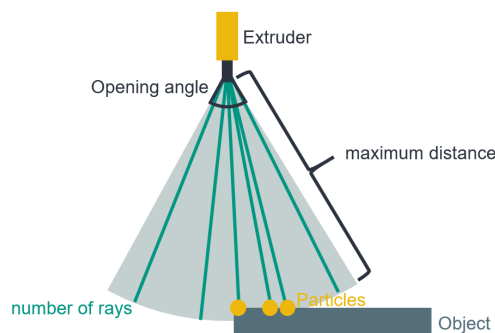
Figure 4: Extruder parameters visualization

These objects are implemented as Materials that can have different properties from massless particles sticking to surfaces (such as paint) to physical bodies like 3D printing plastic. By default, no force is imparted during such processes although custom force models can be added by implementing the `calculate_process_force` function.

Removing of material is either done using the `MillingTool` which uses the Kienzle force model (Kienzle, 1952) for planar milling or the `Remover` which is the twin of the `Extruder` and can be used to simulate ablative processes such as sandblasting or waterjet cutting. The process force model for milling can be seen in Figure 5. Here the chip thickness exponent and the material-specific force are material-dependent.
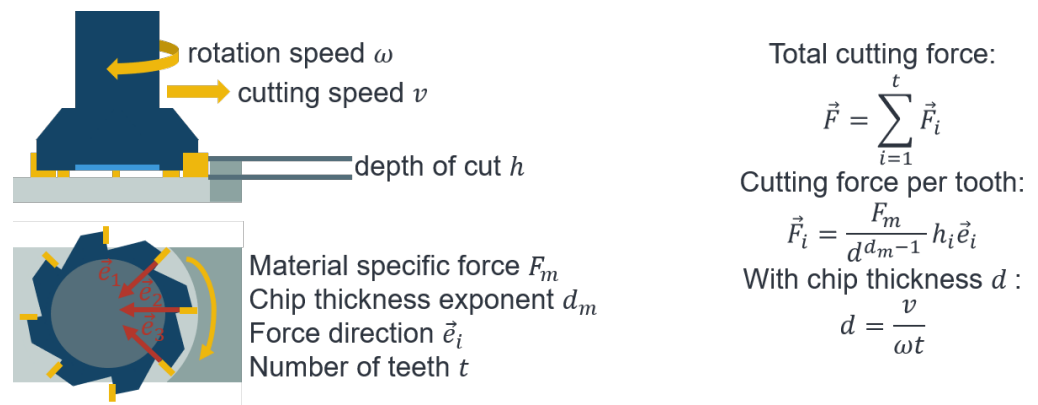
rotation speed ω
cutting speed v
depth of cut h

Material specific force F_m
Chip thickness exponent d_m
Force direction $\vec{e}_i$
Number of teeth t

Total cutting force:
$$\vec{F} = \sum_{i=1}^{t} \vec{F}_i$$
Cutting force per tooth:
$$\vec{F}_i = \frac{F_m}{d^{d_m-1}} h_i \vec{e}_i$$
With chip thickness d :
$$d = \frac{v}{\omega t}$$

Figure 5: Cutting Force calculation as described by the Kienzle Model (Kienzle, 1952)

The moving of material is achieved using grippers. PyBullet Industrial supports both finger grippers and suction grippers for this purpose.

For camera-based applications, the library also contains a camera sensor tool that can be used to simulate process inspection tasks.

Utility

To make development easier, the library has several utility functions. This includes the `ToolPath` class which has a custom iterator making it easy for tools and robots to follow predetermined paths. These paths can be built using different interpolation functions such as linear interpolation, spline interpolation, or circular interpolation. Path positions and orientations can be visualized using drawing functions. These underlying functions can also be used to visualize arbitrary coordinate systems or robot link poses.

Example Research Applications

Apart from the example applications mentioned in the statement of need, the library is also already been used in a number of ongoing research projects. These include:

- A study on VR-based robot programming where a welding task was simulated in VR using PyBullet Industrial. The resulting Project Demonstrator can be seen at the Hannover Messe 2023.
- A project on automated Electromotor disassembly where the simulation is used to validate a given disassembly plan including for example the milling away of rusted screws.

Conclusion

PyBullet Industrial is a novel simulation platform for robot manufacturing research. It allows the simulation of robots and processes in a single environment. While the library offers the basic functionality to simulate robots and processes, these blocks need to be parameterized and combined to create a meaningful simulation. Future work will focus on the development of such parameterization and combination methods. For deployment on real robots, the library will also be extended to directly parse g-code files and convert them into tool paths. For direct control, a ROS interface will be added to allow the use of ROS controllers.

References

Kienzle, O. (1952). Die bestimmung von kräften und leistungen an spanenden werkzeugen und werkzeugmaschinen. *Zeitschrift Des Vereins Deutscher Ingenieure*, *94*, 299–305.

Koenig, N., & Howard, A. (n.d.). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, *3*, 2149–2154.

Kramer, T., Proctor, F., & Messina, E. (2000). *The NIST RS274NGC interpreter - version 3*. NIST Interagency/Internal Report (NISTIR), National Institute of Standards; Technology, Gaithersburg, MD. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=823374

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, *1*(1), 39–42. http://www.ars-journal.com/International-Journal-of-Advanced-Robotic-Systems/Volume-1/39-42.pdf

Mourtzis, D., Doukas, M., & Bernidaki, D. (2014). Simulation in manufacturing: Review and challenges. *Procedia CIRP*, *25*, 213–229. https://doi.org/https://doi.org/10.1016/j.procir.2014.10.032

Mühlbeier, E., Gönnheimer, P., Hausmann, L., & Fleischer, J. (2021). Value stream kinematics. Production at the leading edge of technology. *WGP 2020. Lecture Notes in Production Engineering*, 409–418. https://doi.org/10.1007/978-3-662-62138-7_41

Rohmer, E., Singh, S. P. N., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1321–1326. https://doi.org/10.1109/IROS.2013.6696520

Roth, S. D. (1982). Ray casting for modeling solids. *Computer Graphics and Image Processing*, *18*(2), 109–144. https://doi.org/https://doi.org/10.1016/0146-664X(82)90169-1

Smith, M. (2009). *ABAQUS/standard user's manual, version 6.9*. Dassault Systèmes Simulia Corp.

Tola, D., & Corke, P. (2023). *Understanding URDF: A survey based on user experience*. https://doi.org/10.48550/arXiv.2302.13442