Object Manipulation and Assembly in Modelica

Robert Reiser[1]

[1]Institute of System Dynamics and Control, German Aerospace Center (DLR), Germany,
{firstname.lastname}@dlr.de

Abstract

This paper introduces a new library for the manipulation and assembly of 3D objects using Modelica. The method is based on collision detection, contact dynamics, position and orientation control as well as states for object manipulation. The aim is a stable and efficient simulation of these processes close to real physics. 3D objects (both elementary shapes and CAD objects) can be added from the library browser to a model by drag-and-drop and are directly capable of being manipulated by multiple grippers and conveyors or being assembled to an assembly. This allows a high degree of flexibility and the modeling effort can be decreased significantly.

Keywords: manipulation, grasping, gripper, assembly, collision detection, contact dynamics, state machine

1 Introduction

The manipulation and assembly of objects is widely used in multiple domains (e.g. in the production industry). This leads to a high relevance for the modeling and simulation of these processes. However, current simulation tools are either specialized standalone software with restricted flexibility (Miller and Allen 2004; León et al. 2010) or embedded in plant planning software and therefore limited to this specific domain (Kühn 2006). On the other hand, the object-oriented modeling language Modelica (Modelica Association 2017) offers cross-domain flexibility.

In the work of Ferretti et al. (2006), a gripper was modeled and simulated using the Modelica Multibody Library (Otter, Elmqvist, and Mattsson 2003). The Modelica Contact Library of Oestersötebier, Wang, and Trächtler (2014) aims at the idealized simulation of contacts with non-central contact blocks. In both works, the contact pairs are pre-defined in the model. Therefore, an object cannot be grasped by multiple grippers and can only be placed on the same ground. Elmqvist et al. (2015) introduced a framework for multibody contacts in Modelica using the discrete element method where objects interact without predefined connections. However, due to their complexity, all models are not well suited for real time simulations.

The new solution presented in this paper aims at the stable simulation of manipulation and assembly processes in real-time close to real physics. The contact dynamics model is therefore simplified and only used for the contact between object and ground or conveyor and the initial contact between object and gripper jaws. The stable contact between object and gripper or within assemblies is achieved by a "fixed connection" (nstead of contact forces and torques (see subsection 3.3). 3D objects can be added to a model without any pre-defined connection and are directly capable of being manipulated or assembled.

The developed library (which is currently only used internally) combines the flexibility of a non-causal, object-oriented Modelica multi-body environment with the performance of a collision detection algorithm in a linked C library. Precise simulations of the contact and frictional forces or multipoint contacts are out of the scope of this work. However, the library is modular and can be extended in multiple ways, e.g. adding precise physical models for object contact.

The following section gives an overview of the library including the blocks which can be used for modeling and information about the usage of the library. Section three deals with the models and algorithms including contact detection, contact dynamics, hold control and manipulation logic. The next section presents the concepts of object manipulation and assembly.

In addition, two applications are shown, one for the object manipulation and one for the assembly of an object. In conclusion, the advantages and disadvantages of the solution as well as future developments are discussed.

2 Overview

The first part of this section is about the available blocks and the second part about the usage of the library. Figure 1 shows the structure of the library.

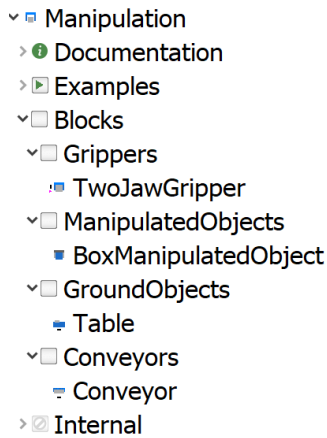

Figure 1. Overview of the library structure.

2.1 Blocks

The available blocks are shown in Figure 2 (visualization) and 3 (model) and introduced in the following subsections.

2.1.1 Grippers

The **`TwoJawGripper`** is a simple gripper with two jaws (it is possible to model grippers with more than two jaws). It can be attached to a robot. The following parameters can be set by the user. `vMax` is the maximum speed of the jaws for closing and opening. `jawPenMax` defines the maximum penetration depth for the jaws. Once the jaws have contact to the manipulated object, the velocity is reduced in relation to the penetration until the given maximum penetration is reached. The `graspDuration` is the time period after which the grasping transitions from jaw contact to stable grasping and therefore the jaws will stop the movement. There are two input states: close (jaws move inwards with the maximum velocity) and open (jaws move outwards with the same velocity). The gripper only contains a kinematic model, i.e. the gripper and its jaws are free of forces and torques.

2.1.2 Manipulated objects

The main object is the **`BoxManipulatedObject`**. It is a cuboid whose dimensions `dim` can be defined by the user. Additional parameters are the start position `rStart`, the start orientation `anglesStart`, the mass `m` and the inertia tensor as well as visualization properties. If `useJawContact` is true, the contact forces and torques caused by the jaws are used for the duration defined in `graspDuration` (only for the object and not for the gripper). If false, the object is directly attached to the gripper (close to real physics, see section 3).

The object can be used both as cuboid and as CAD object. For the latter, the CAD object is only used for visualization, i.e. the cuboid is still used for contact. Therefore, it is possible to manually set the surface for grasping (e.g. only a part of the CAD object is covered by the cuboid).

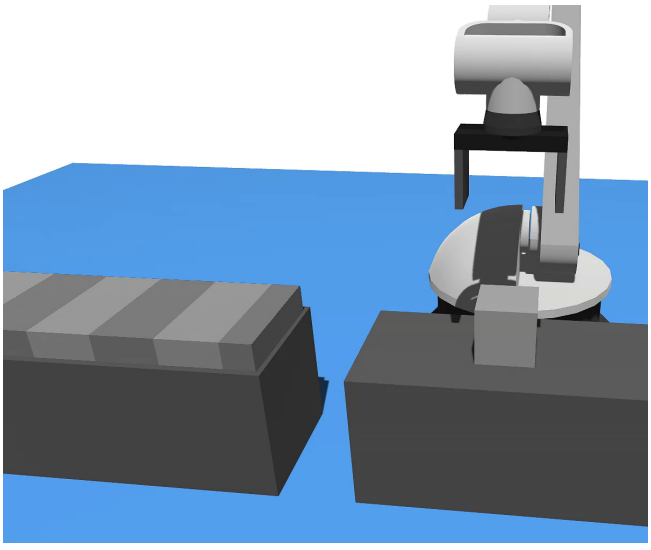

Figure 2. Visualization of Modelica blocks for object manipulation and assembly: `TwoJawGripper` (attached to robot), `Table` (bottom right), `BoxManipulatedObject` (on top of the table) and `Conveyor` (left).

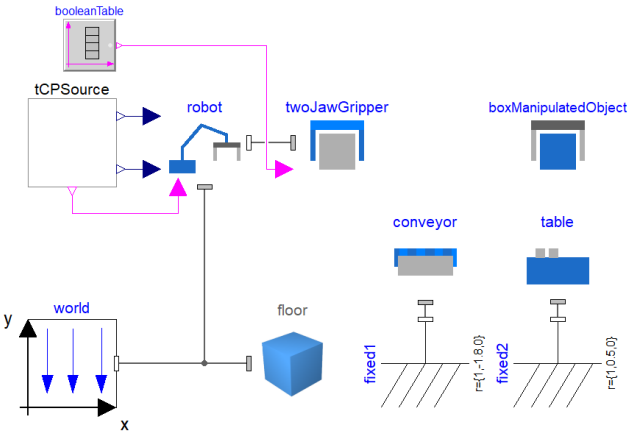

Figure 3. Modelica model with robot, `TwoJawGripper`, `Table`, `BoxManipulatedObject` and `Conveyor`.

2.1.3 Ground objects

The **`Table`** can be used as a ground object and is parameterized by the dimensions `dim` and visualization properties. The `BoxManipulatedObject` can be placed on a `Table`, which does not use forces and torques.

2.1.4 Conveyors

For the movement of objects, the **`Conveyor`** model can be used. It is also free of forces and torques and represents a moving belt which leads to a relative velocity of the `BoxManipulatedObject`. The resulting friction forces and torques move the object. The user can set the parameters `units` (number of belt elements), `velocity`, the belt dimensions, the frame dimensions and visualization properties. It can be placed horizontally or oblique to the ground. It is capable of moving objects upwards within a angle which depends on the contact parameters set for the `BoxManipulatedObject`.

2.2 Usage

Blocks can be added from the library browser to a model by drag-and-drop. There are no connections between the objects in the model because contacts are not pre-defined. All objects are directly capable for manipulation or assembly. For the `BoxManipulatedObject`, the start position and the start angles have to be defined. The initial position must be on top of a `Table` or `Conveyor`. An initialization within a closed gripper is planned, but not yet implemented. The parameter `useJawContact` is true by default.

The `TwoJawGripper` has to be attached to a robot. If the `graspDuration` (i.e. the period after the activity transitions from jaw contact forces to stable grasping) is changed, it must be changed both for the gripper and the `BoxManipulatedObject`.

DOI 10.3384/ecp21181433

3 Models and algorithms

This section shows the main models and algorithms used for the manipulation and assembly of objects. At first, the contact detection concept is introduced. In addition, the models for object forces and hold control are explained. The last subsection contains the manipulation logic.

3.1 Contact detection with libccd

The contact detection of the presented Manipulation library is done by **libccd**, a library for the calculation of collisions and penetrations between objects (Fiser 2018). It is written in C, open source under the 3-clause BSD license (Open Source Initiative 2020) and implements the following contact detection algorithms:

- Gilbert-Johnson-Keerthi (GJK)
- Expand-Polytope-Algorithm (EPA)
- Minkowski Portal Refinement (MPR)

For the purpose of this paper, the MPR algorithm is used because it is more stable.

Minkowski Portal Refinement is a collision detection algorithm developed by Gary Snethen (Snethen 2008). It is similar to the GJK algorithm.

Both are limited to convex shapes and use the Minkowski difference which can be described as *"the region swept by Object A translated to every point negated in Object B"* (Serrano 2016). It is widely used in collision detection because the location of the origin indicates if both objects collide. If the Minkowski difference includes the origin, there is a collision. (Serrano 2016)

The MPR algorithm also uses the Support Function, which is a *"function that takes a convex hull (or convex polygon) and a support vector representing a direction of search. The function returns the point on the body that is farthest in the sense of a plane sweep using the support vector"* (Newth 2013).

The MPR algorithm in 2D can be described as follows (Snethen 2008; Newth 2013) and is shown in Figure 4. In general, it consists of two phases: portal discovery and portal refinement.

The **portal discovery** starts from the Minkowski difference between two objects and the origin `O` (Figure 4a).

- The phase begins with the definition of an interior point `V0` within the Minkowski difference (e.g. the geometric center). The origin ray is then defined as the ray from `V0` to `O` (Figure 4b).
- With the origin ray, the first support point `V1` is determined based on the support function. The second support point `V2` is discovered by a ray perpendicular to the vector from `V0` to `V1` (Figure 4c, 4d).
- The support points `V1` and `V2` are now defining a portal. If the origin passes through this portal, the algorithm continues with the second phase. If not, the process above continues with finding a new support point in normal direction to the portal (Figure 4e, 4f).

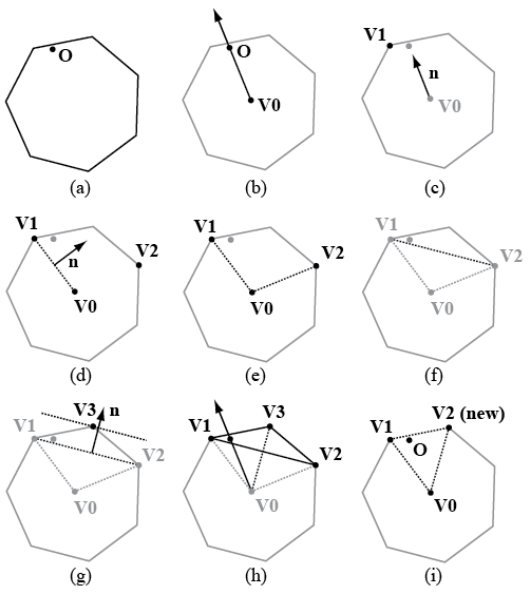

Figure 4. Phases of the MPR algorithm (Snethen 2008). The initial portal is discovered so that the ray from the interior point V0 to the origin O passes through (V1 and V2 define the initial portal). This portal is then refined until the origin is on the inside of the portal (V1 and V2 (new) define the final portal). The shape represents the Minkowski difference between two objects.

The **portal refinement** starts from an initial portal.

- It is checked, if `O` is inside the portal. If this is not the case (Figure 4f), the portal will be refined.
- A normal perpendicular to the current portal is created to find the next support point `V3` and a new portal is created (Figure 4g, 4h).
- This step is repeated until `O` is inside the portal (Figure 4i).

The MPR algorithm in 3D is similar with the portal becoming a triangle instead of a line segment. Based on the MPR algorithm, libccd is able to determine if two objects collide and if so to calculate the position, direction and maximum depth of the collision.

The interface between libccd and Modelica is based on unpublished work of Hellerer (2019). With this library it is possible to define the mentioned object types in Modelica and perform collision detection in Modelica based on libccd.

In addition, the unpublished library of Buse (2021) adds collision groups and the memory handling on C, which enables the usage of collision objects in Modelica models by drag-and-drop.

In the context of object manipulation, the contact detection is limited to a single contact, i.e. an object can collide

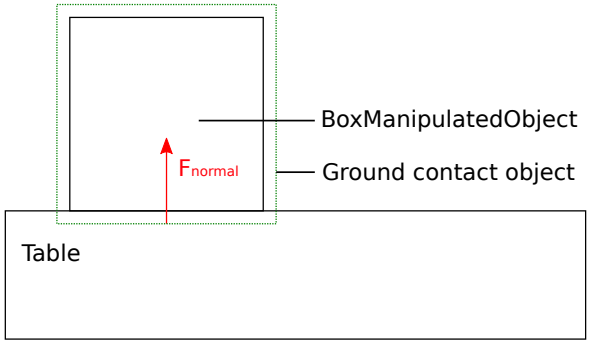

Figure 5. Contact between `BoxManipulatedObject` and `Table` based on the contact dynamics model. The `BoxManipulatedObject` contains an object for ground contact (green dotted, invisible in simulation visualization) with larger dimensions to compensate the lower stiffness necessary for fast simulations.

with multiple other objects, but only the contact with the maximum depth is transferred to Modelica.

3.2 Contact dynamics model

This model is used for forces and torques between objects, both for ground contact (object is placed on a table or conveyor) and jaw contact (interaction between jaws and the grasped object).

The libccd uses support functions which are defined by the geometry of the shapes during the initialization of the simulation. During simulation, the current position, orientation and velocity of all collision objects are constantly transferred to the libccd. In return, information about the collision are transferred to Modelica. These are the position, direction and depth of a collision.

The contact dynamics model is simplified so that a fast and stable simulation close to real physics is possible. The normal force is based on a spring-damper-model:

$$F_{normal} = k \cdot s_{penetration} + d \cdot v_{penetration} \tag{1}$$

where $s_{penetration}$ is the penetration depth and $v_{penetration}$ the velocity of the penetration. The spring constant k and the damping constant d are set by the user. For the calculation of the friction force, the simplified Coulomb friction model from (Andersson, Söderberg, and Björklund 2007) is used:

$$F_{friction} = \mu \cdot F_{normal} \cdot \text{sign}(v_{tangential}) \tag{2}$$

which is simplified to

$$F_{friction} = \mu \cdot F_{normal} \cdot \tanh(k_{tanh} \cdot v_{tangential}) \tag{3}$$

where μ is the coefficient of friction and $v_{tangential}$ the tangential velocity. For simulation performance, sign() is replaced by tanh(), scaled by the coefficient k_{tanh}.

Since hard contacts decrease the performance of the simulation significantly, it is possible to use soft contact parameters, i.e. lower stiffness (e.g. for ground contact). To prevent objects from sinking in the ground when soft contacts are used, the dimensions for the contact object can be extended which compensates for the lower stiffness and leads to visualizations close to reality. The dimension for the ground contact object dim_{ground} is calculated by:

$$dim_{ground} = dim + \frac{m \cdot g}{k} \cdot \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \tag{4}$$

where dim is the dimension, m the mass and k the the spring constant of the `BoxManipulatedObject`.

The `BoxManipulatedObject` contains three contact dynamics models for ground and both jaw contacts, each with its own collision object. Therefore, the contact parameters can be set specifically for each contact. Figure 5 shows the `BoxManipulatedObject` and its collision object for ground contact. The larger dimension of the ground contact object compensates the lower stiffness of the contact. The jaw contact objects are similar to the ground contact object (see Figure 6). Each contact object generates a force and torque which are summarized in the `BoxManipulatedObject`.

3.3 Hold control

The hold control generates a "fixed connection" between two objects and is used for stable grasping and stable assembly. The concept consists of two objects:

- A low-level object (LLO) which is held by the top-level object. It contains a controller for position and orientation control.
- The top-level object (TLO) which is only necessary for collision detection.

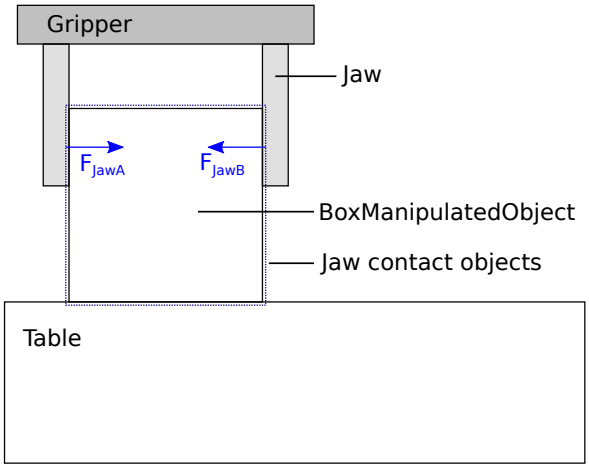

Figure 6. Contact between `BoxManipulatedObject` and `TwoJawGripper` based on the contact dynamics model. The `BoxManipulatedObject` contains one object for each jaw contact (blue dotted, invisible in simulation visualization).

DOI 10.3384/ecp21181433

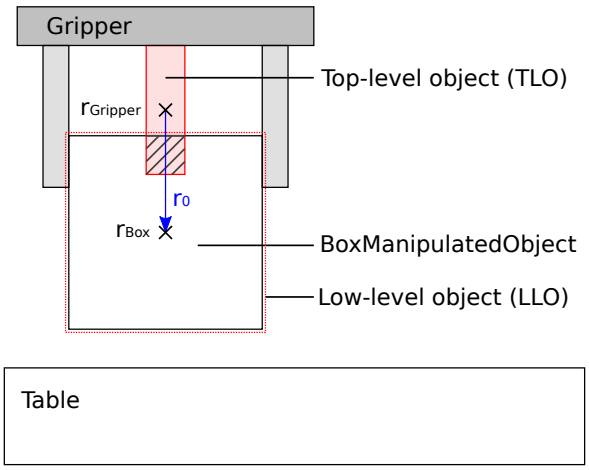

Figure 7. Hold control used for the gripper. The LLO (invisible in simulation visualization) gets the position and orientation from the TLO (invisible in simulation visualization). Based on the initial difference (r_0), the target position and orientation is calculated and a PI-controller is used to keep the object there.

The procedure can be described as follows:

At first, the collision detection is disabled for the TLO. Then it is moved into the LLO to get a penetration. From the moment the LLO object should be held in place by the TLO (i.e. the gripper closes), the ability of the TLO to collide is activated. The LLO recognizes a collision and stores the initial difference in position and orientation between both objects:

$$r_0 = \mathrm{T}_{\mathrm{Gripper}} \cdot (r_{Box} - r_{Gripper}) \tag{5}$$

$$\mathrm{T}_0 = \mathrm{T}_{\mathrm{Box}} \cdot {\mathrm{T}_{\mathrm{Gripper}}}^T \tag{6}$$

where r_{Box} is the position and $\mathrm{T}_{\mathrm{Box}}$ the rotation matrix of the box. From the moment the gripper begins to close, the LLO recieves continuously the position ($r_{Gripper}$) and orientation ($\mathrm{T}_{\mathrm{Gripper}}$) of the TLO. Based on this information and the stored initial difference, it is possible to calculate the target position and orientation for the object:

$$r_{Target} = r_{Gripper} + {\mathrm{T}_{\mathrm{Gripper}}}^T \cdot r_0 \tag{7}$$

$$\mathrm{T}_{\mathrm{Target}} = \mathrm{T}_0 \cdot \mathrm{T}_{\mathrm{Gripper}} \tag{8}$$

A PI-controller and a gravity compensation for the weight are finally used to keep the LLO in the target position and orientation.

The forces and torques are only present in the low-level object: in the grasped object (not in the gripper) and in the sub-assembly (not in the assembly).

Figure 7 shows the application of the hold control model for a gripper. The `TwoJawGripper` contains a TLO. It is invisible in the visualization and penetrates the `BoxManipulatedObject`, which contains the LLO.

3.4 Manipulation logic

The manipulation logic is used to switch between the states of a `BoxManipulatedObject`. There are the following states (see Figure 8):

1. `GroundContact` (initial state): The object can be placed on a ground object (e.g. `Table`) or a `Conveyor`.
2. `JawContact` (grasping part with jaw interaction until hold control is used): Both ground contact forces and jaw contact forces are used. This state is only used if `useJawContact` is true.
3. `Grasping` (based on hold control): The object is attached to the `TwoJawGripper` based on hold control for position and orientation. The transition from state 2 to state 3 is done if the `graspingTime` is over or if the grasping is stable (minimum relative rotation and movement between jaw and object).
4. `Assembly`: Is used if the object is placed in an assembly and grasping is over. `Grasping` is always higher prioritized than `Assembly`.
5. `Delay` (between `Assembly` and `Grasping`): This is necessary for the following condition:
   - The object is part of an assembly.
   - Assembly and object are grasped by a gripper.
   - `useJawContact` is enabled.

   In this case, the object would normally be instantly controlled by the gripper. This delay holds the object in `Assembly` state until `JawContact` for the assembly is done. This state is necessary, if e.g. the basis of an assembly is slightly rotated by the jaws: all parts of the assembly have to be rotated as well.

The transitions between the states (shown in Figure 8) are defined as follows:

```
a := graspingActive and useJawContact and
     not stableGrasping and not
     assemblyActive
b, c, f, g := graspingActive and (not
     useJawContact or useJawContact and ((
     stableGrasping or (time >=
     graspInitTime + graspDuration))))
```

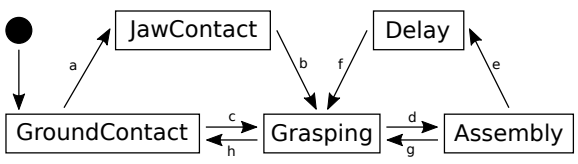

Figure 8. Manipulation states and their transitions.

```
d := not graspingActive and assemblyActive
e := graspingActive and useJawContact and
    not stableGrasping and assemblyActive
h := not graspingActive and not
    assemblyActive
```

4 Object manipulation and assembly

In this section, the concepts of object manipulation and assembly are shown. This includes the placement of objects on ground and on conveyors, grasping objects with grippers and the assembly of objects.

4.1 Object placement on ground

A `BoxManipulatedObject` contains a ground contact object with the underlying contact dynamics model. Based on this, the object can be placed on a `Table` as mentioned in subsection 3.2 and shown in Figure 5.

4.2 Object transport with conveyors

The same model is used if the object is placed on a `Conveyor`. The only difference is the velocity of the `Conveyor` which leads to a movement of the `BoxManipulatedObject` (see subsubsection 2.1.4).

`Conveyors` can be attached together. The subsequent conveyor has to be slightly lower relative to the previous one to avoid the `BoxManipulatedObject` from being stuck between both `Conveyors`.

4.3 Object grasping with grippers

An object can be grasped by all `TwoJawGrippers`. The `BoxManipulatedObject` contains two jaw contact objects to apply forces and torques from both gripper jaws (see subsection 3.2).

In addition, hold control is used for a stable connection between object and gripper as explained in subsection 3.3.

A transfer of a `BoxManipulatedObject` from one `TwoJawGripper` to another is possible to cover a wide range of applications.

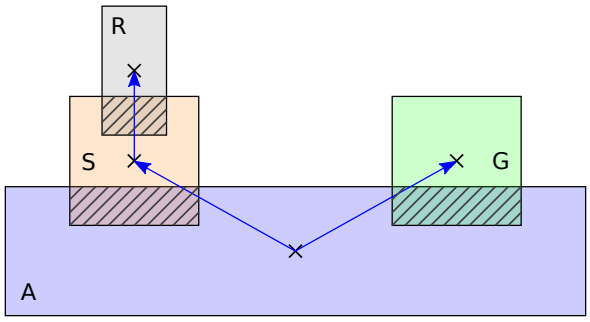

Figure 9. Assembly with sub-assembly. R is assembled to S and both S and G are assembled to A. The vectors are showing the position differences between the object origins.

4.4 Object assembly

The assembly of objects is based on the same principle as grasping, namely the hold control concept introduced in subsection 3.3. The `BoxManipulatedObject` contains a TLO and a LLO for assembly.

If two objects are assembled, one object must be defined as base and the other one as module. The user has to enable the TLO for the base and the LLO for the module. Then it is possible to "attach" the module to the base.

The same applies if two modules are assembled to a base. Both modules need an activated LLO and the base represents the TLO for the hold control model.

For the representation of sub-assembly processes, the definition of the TLO and LLO is more complicated. Therefore, collision groups are used. For each collision object, `assemblyCollisionGroups` and `assemblyIncludedGroups` can be defined.

The concept is demonstrated with an example containing four objects: gray rod (R), orange sub-assembly box (S), green box (G) and main assembly (A) (see Figure 9).

The following collision groups are defined:

```
R.assemblyCollisionGroups = {""}
R.assemblyIncludedGroups = {"SubAssembly"}
S.assemblyCollisionGroups = {"SubAssembly"}
S.assemblyIncludedGroups = {"Assembly"}
G.assemblyCollisionGroups = {""}
G.assemblyIncludedGroups = {"Assembly"}
A.assemblyCollisionGroups = {"Assembly"}
A.assemblyIncludedGroups = {""}
```

This means:

- R is a LLO in relation to S
- S is a TLO in relation to R and a LLO regarding A
- G is a LLO in relation to A
- A is a TLO in relation to S and G

Figure 9 shows the assembly for the example. The vectors are showing the initial position differences used for the hold control model. To manipulate an entire assembly, the `TwoJawGripper` can only grasp the base object and not the sub-assemblies. If a `TwoJawGripper` grasps a sub-assembly, it can be disassembled.

5 Applications

In this section one example for object manipulation and one for assembly are shown. The simulations are visualized with the DLR Visualization 2 Library (Kümper, Hellerer, and Bellmann 2021).

5.1 Object manipulation

In this example, objects are manipulated with grippers (attached to robots) and `Conveyors`. The model is shown in Figure 10 and the visualization illustrated in Figure 11. There are the following objects:

- Orange box on front table (O)

DOI 10.3384/ecp21181433

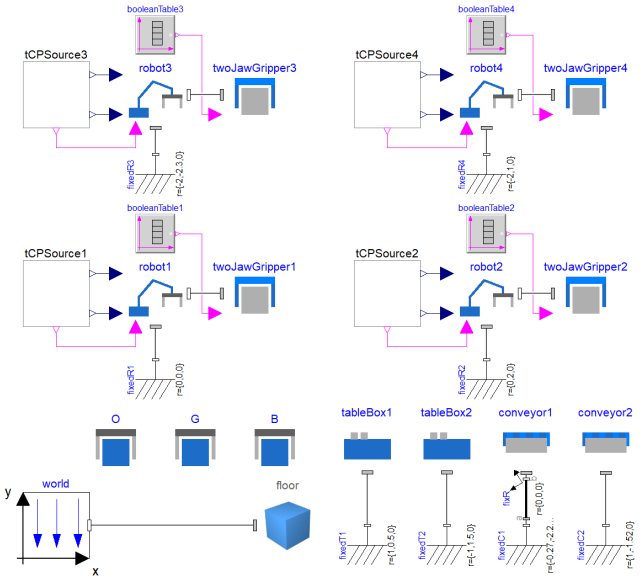

Figure 10. Modelica model for the manipulation of three objects (O, G and B) with robots, grippers and conveyors.

- Green box on front table (G)
- Blue box on back table (B)

At first, the robot in the front grasps object G and places it on the right of the same `Table`. The initial orientation of G is rotated 8 degrees to the lateral surface of the `Table`. Since the gripper is held in an orthogonal orientation above the object, the jaws force the object into the orientation of the gripper. Therefore, G will be placed in a perfect orientation on the `Table`. Meanwhile, the robot in the back moves B to the position between the two robots in the back. The object is then handed over to another robot (Figure 11 (middle)), while G is moved to the back `Table` and O to the right of the `Table`. Object O is rotated in the beginning as well. Finally, B is placed on the `Conveyor` and moves back to the front, onto the other `Conveyor` and then to the `Table`. The end of the sequence is shown in Figure 11 (bottom).

The model was tested in Dymola 2020 (64-bit) on Windows 10 on a Intel® Xeon® W-2135 workstation. A Rkfix2 solver with the fixed step of 0.001 was used. The "CPU-time for integration" for the 40 s simulation was 19.1 s. Therefore the model is real-time capable.

5.2 Assembly of an object

This example is the same as mentioned in subsection 4.4. It shows the assembly of a module cosisting of multiple `BoxManipulatedObjects` with `TwoJawGrippers` and robots including a subassembly. The model is shown in Figure 12. The initial setting is visualized in Figure 13 (top). There are the four objects R, S, G and A.

The first step is to assemble the rod into the subassembly box. This is done by the left robot. Then both S and G are assembled to the main assembly by both robots. Finally, the right robot is able to grasp the entire assembly as shown in Figure 13 (bottom) and to move it to the second `Table` in the back.

In this example, the hold control model is used for grasping and for assembly. R is "attached" to S in the same way as a box is "attached" to a gripper. S and G are connected to A similarly.

The same settings as in subsection 5.1 are used to simulate the model. The 40 s simulation took 14.5 s for integration, i.e. the model is real-time capable as well.

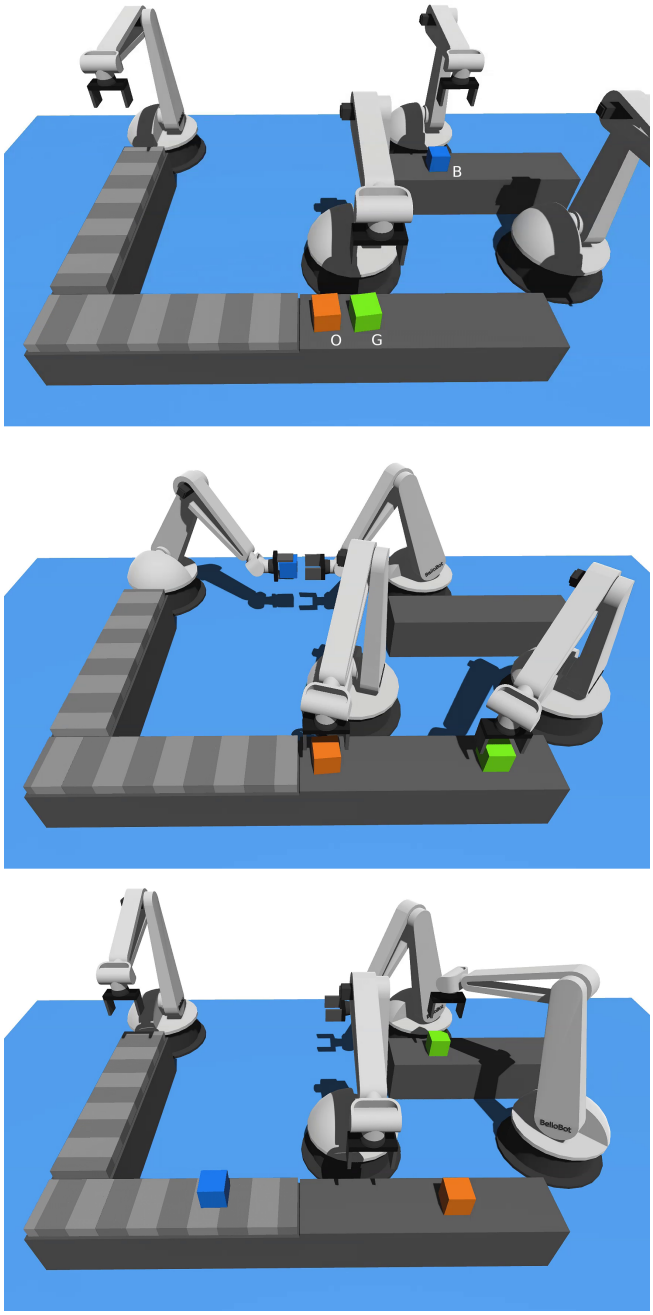

Figure 11. Manipulation of multiple objects with `Grippers` and `Conveyors` with start (top), object handed over from one robot to another (middle) and end of sequence (bottom).

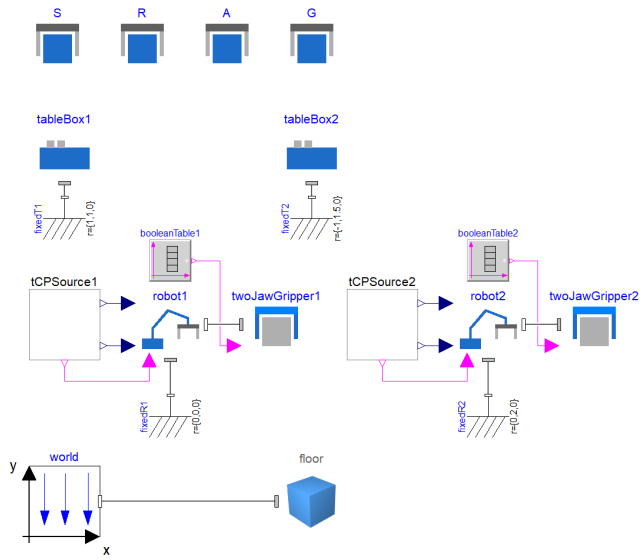

Figure 12. Model for the assembly of the objects S, R, A and G.

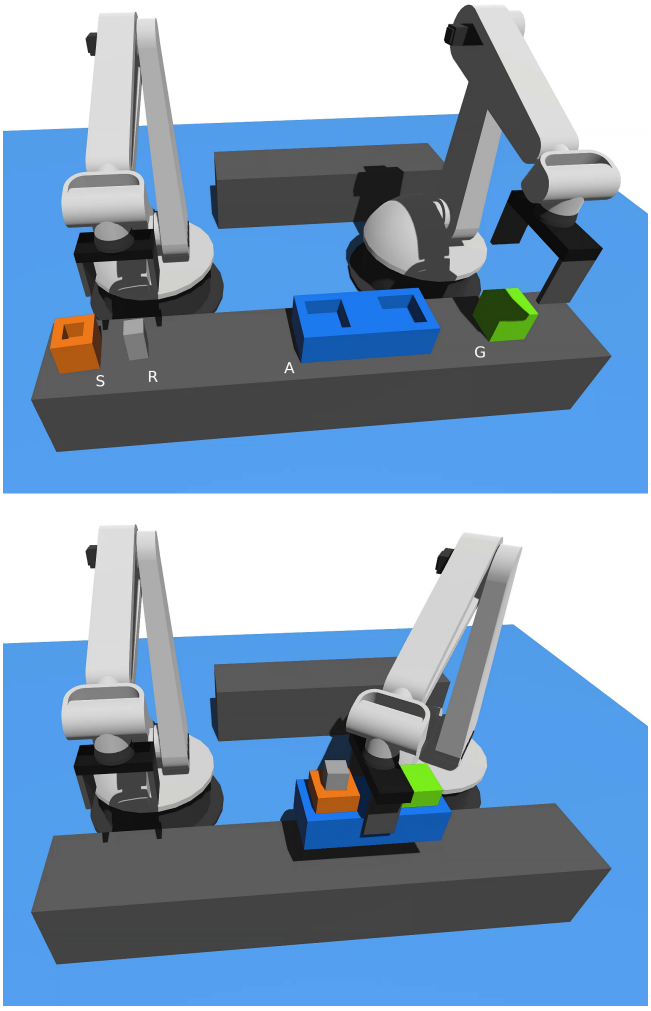

Figure 13. Assembly of an object including a sub-assembly. The grey rod is assembled to the orange box and both the sub-assembly and green box are assembled to the blue base box.

6 Discussion

The successful application of the library for object manipulation and assembly was demonstrated in the previous section. Both mentioned models (Figure 10, 12) are real-time capable. The simulation speed depends on the number of manipulated objects in the model. Real-time simulations are currently possible for models with up to 20 objects (depending on the number of robots, grippers, conveyors and tables in the model). However, the library still has restrictions and limitations:

- The contact detection algorithm is limited to one contact, therefore an object can be placed on a table but it is not possible to place another object on top of the former object.
- If an assembly is grasped by a gripper, both the gripper and the base-object of the assembly must have contact (i.e. it is not possible to grasp an assembly by grasping a sub-assembly of the assembly).
- There are no forces and torques on top level. Therefore, it is not possible to use the library for robot dynamic simulations.

Possible future developments are:

- Forces and torques for the top-level object (e.g. gripper) in the hold control model.
- Maximum hold force in the gripper. If the force exceeds the maximum, the connection will be disabled.

7 Conclusion

The combination of the non-causal, object-oriented Modelica multi-body environment and the performant MPR collision detection algorithm in C are the basis for a new solution to the modeling and simulation of manipulation and assembly processes. The real-time capability and stability were demonstrated in two use cases.

Blocks for manipulated objects, grippers, conveyors and tables can be added from the library browser to a model by drag-and-drop. This allows a low modeling effort with a high flexibility at the same time. The blocks can be combined with all libraries in the Modelica environment. Therefore, this library lays the foundation for plant simulation on multiple levels of detail.

Acknowledgements

I would like to thank Tobias Bellmann for fruitful discussions regarding the library, Fabian Buse for valuable discussions about and providing models for contact detection and contact dynamics, Matthias Hellerer and Bernhard Thiele for help with Modelica, Patrick Weber for providing the basis for the conveyor and Sebastian Kümper for help with the visualization (all DLR).

This work benefited from the MFlex 2025 project founded by the German Federal Ministry for Economic Affairs and Energy (BMWi).

References

Andersson, Sören, Anders Söderberg, and Stefan Björklund (2007). "Friction models for sliding dry, boundary and mixed lubricated contacts". In: *Tribology International* 40.4. NORDTRIB 2004, pp. 580–587. DOI: 10.1016/j.triboint.2005.11.014.

Buse, Fabian (2021). *ContactDynamics. Modelica Library. Unpublished work.*

Elmqvist, Hilding et al. (2015). "Generic Modelica Framework for MultiBody Contacts and Discrete Element Method". In: *Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015*. The 11th International Modelica Conference. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 427–440. DOI: 10.3384/ecp15118427.

Ferretti, Gianni et al. (2006). "Modelling and simulation of a gripper with Dymola". In: *Mathematical and Computer Modelling of Dynamical Systems* 12.1, pp. 89–102. DOI: 10.1080/13873950500071405.

Fiser, Daniel (2018). *libccd. Library for collision detection between two convex shapes*. URL: https://github.com/danfis/libccd (visited on 2020-04-28).

Hellerer, Matthias (2019). *CollisionDetection. Modelica Library. Unpublished work.*

Kühn, Wolfgang (2006). *Digitale Fabrik. Fabriksimulation für Produktionsplaner*. 1. Aufl. München and Wien: Carl Hanser Fachbuchverlag. 495 pp. ISBN: 978-3-446-40619-3.

Kümper, Sebastian, Matthias Hellerer, and Tobias Bellmann (2021). "DLR Visualization 2 Library - Real-Time Graphical Environments for Virtual Commissioning". In: *14th International Modelica Conference 2021*. Ed. by Martin Sjölund et al.

León, Beatriz et al. (2010). "OpenGRASP: A Toolkit for Robot Grasping Simulation". In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, pp. 109–120. DOI: 10.1007/978-3-642-17319-6_13.

Miller, Andrew T. and Peter K. Allen (2004). "Graspit! A Versatile Simulator for Robotic Grasping". In: *IEEE Robotics & Automation Magazine* 11.4, pp. 110–122. DOI: 10.1109/MRA.2004.1371616.

Modelica Association (2017). *Modelica – A Unified Object-Oriented Language for Systems Modeling. Language Specification Version 3.4. Tech. Rep.* Linköping: Modelica Association. URL: https://modelica.org/documents/ModelicaSpec34.pdf.

Newth, Joshua (2013). "Minkowski Portal Refinement and Speculative Contacts in Box2D". Master's Thesis. San Jose State University. URL: http://www.cs.sjsu.edu/faculty/pollett/masters/Semesters/Spring12/josh/joshua_newth.pdf.

Oestersötebier, Felix, Peng Wang, and Ansgar Trächtler (2014). "A Modelica Contact Library for Idealized Simulation of Independently Defined Contact Surfaces". In: *Proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden*. the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden. Linköping Electronic Conference Proceedings. Linköping University Electronic Press, pp. 929–937. DOI: 10.3384/ecp14096929.

Open Source Initiative (2020). *The 3-Clause BSD License*. URL: https://opensource.org/licenses/BSD-3-Clause.

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003). "The New Modelica MultiBody Library". In: *Proceedings of the 3th International Modelica Conference* (Linköping, Sweden). Ed. by Peter Fritzson, pp. 311–330. URL: https://modelica.org/events/Conference2003/papers/h37_Otter_multibody.pdf.

Serrano, Harold (2016). *Visualizing the GJK Collision detection algorithm*. URL: https://www.haroldserrano.com/blog/visualizing-the-gjk-collision-algorithm.

Snethen, Gary (2008). *Minkowski Portal Refinement in 2D. XenoCollide*. URL: http://xenocollide.snethen.com/mpr2d.html.