# *OpenMRH*: a Modular Robotic Hand Generator Plugin for *OpenRAVE*

Filippo Sanfilippo[1] and Kristin Ytterstad Pettersen[2]

***Abstract*— In this work, the open-source plugin *OpenMRH* is presented for the *Open Robotics Automation Virtual Environment* (*OpenRAVE*), a simulation environment for testing, developing and deploying motion planning algorithms. The proposed plugin allows for a fast and automated generation of different modular hand models. *OpenMRH* combines virtual-prototyping and modular concepts. Each modular model is generated by applying a dynamically generated code, which is consistent with the standard syntax expected by *OpenRAVE* for the simulated models. In this way, once the desired model is generated, an instance of *OpenRAVE* can be launched and the model can be visualised. Alternatively, the modular models can be generated from a user-defined input specified via a graphical user interface (GUI). The generated models can be used for testing, developing and deploying grasp or motion planning algorithms. Two case studies are considered to validate the efficiency of the proposed model generator. In the first case study, a modular robotic hand model is generated with *OpenMRH* by using user-defined input parameters. In the second case study, another hand model is generated with *OpenMRH* by using algorithmic defined input parameters.**

## I. Introduction

The human hand is capable of grasping an astounding variety of objects of different shapes, textures, weights and spatial orientations. One of the most ambitious steps in getting a robot to fully mimic the movement of the human hand consists of building a robotic hand with sufficient dexterity. However, the development of such a hand is challenging because a high number of degrees of freedom (DOF) is required to be controlled.

A promising solution for getting such flexibility consists of using a modular approach [1]. Modular grasping makes it possible to use only the necessary number of DOF to accomplish a specific task. As such, a trade-off between simple grippers and more complex human-like manipulators can be reached. Modularity offers robustness to hardware failures considering that the robot parts are interchangeable. The production cost can also be considerably reduced and the weight of the manipulator can be minimised. Moreover, modularity is advantageous in terms of versatility since the robotic hand can be disassembled and reassembled to form new morphologies that are suitable for new tasks.

To fully exploit the concept of modular grasping, an iterative algorithm for a bio-inspired design of modular grippers was introduced in [2]. The proposed method is a simulation based approach that allows for determining effective modular hand configurations to get efficient grasps of given objects. An iterative procedure is adopted: starting from the simplest modular structure, different configurations are tested, with a greater number of DOF at each iteration. The goal of this procedure is to obtain a modular configuration that reaches a predetermined performance in terms of grasp quality using the least amount of modules possible. The resulting modular hand configurations are able to perform effective grasps that a human would consider stable. Nonetheless, in this preliminary work, all the different configurations were manually built with the *Open Robotics Automation Virtual Environment* (*OpenRAVE*) [3], a simulation environment that allows for testing, developing and deploying motion planning algorithms. Even though the effectiveness of the proposed design algorithm has been proved, the manual process of generating different modular configurations was a tedious and time-consuming task, which required a significant effort for the designer. This challenge also affects most of the existing simulation environments. Even though several simulation environments are openly available to researchers, the manipulator models must be manually built.

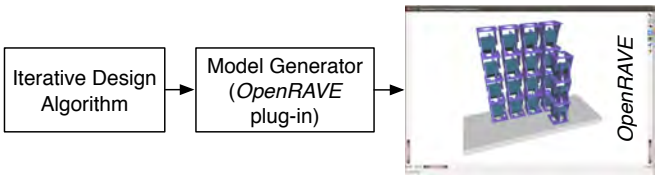



Fig. 1: The idea of realising a plugin for *OpenRAVE* that allows for a fast and automated generation of different modular hand models.

To overcome this challenge, *OpenMRH* – a model generator for modular robotic hands – is presented in this article as an *OpenRAVE* plugin. This plugin may work as *middleware* between the previously proposed iterative design algorithm and the *OpenRAVE* simulation environment, as shown in Fig. 1. Alternatively, the modular models can be generated from a user-defined input from a graphical user interface (GUI). This idea makes it possible to combine the virtual-prototyping approach with the modular concept. Each modular model can be automatically built by applying a dynamic code-generation process. The self-generated models are consistent with the standard guidelines expected by *OpenRAVE* for simulated robots. In this fashion, once the model is generated, an instance of *OpenRAVE* can be initialised and the model can be visualised within the selected simulation environment. The generated models can be used for testing, developing and deploying grasp or motion

[1]Filippo Sanfilippo is with the Dept. of Maritime Technology and Operations, Aalesund University College, Postboks 1517, 6025 Aalesund, Norway. fisa@hials.no

[2]Kristin Ytterstad Pettersen is with the Centre for Autonomous Marine Operations and Systems, Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway. kristin.y.pettersen@itk.ntnu.no

planning algorithms. *OpenMRH* is an open-source project and it is available online at `https://github.com/aauc-mechlab/openMRH`. To validate the efficiency of the proposed model generator, related simulations are carried out in this paper.

The paper is organised as follows. A review of the related research work is given in Sec. II. In Sec. III, we describe a generalised modular model. Based on this model, we summarise the previously presented iterative design algorithm emphasising the need for an automated model generation process. Following this, we focus on the description of the proposed model generator plugin. Two case studies are described in Sec. IV. In the first case study, a modular robotic hand model is generated with *OpenMRH* by using user-defined input parameters. In the second case study, another hand model is generated with *OpenMRH* by using algorithmic defined input parameters. In Sec. V, conclusions and future works are outlined.

## II. Related Research Work

Previous attempts to deal with automatic model generation for modular robots are very limited in literature. From a design point of view, a simulation based prototyping can be beneficial when developing robots with different configurations. Development time can be significantly reduced, the system properties can be analysed and the corresponding performance can be assessed. Therefore, simulation and virtual prototyping are necessary steps to validate the design before making a physical prototype.

Different robotic simulation environments have been presented in the last few years. For instance, *OpenRAVE* is a common tool and is adopted by several researchers in this field. *OpenRAVE* provides a flexible developing environment with a seamless integration of 3-D simulation, visualisation, planning, scripting and control. The plugin-based architecture of *OpenRAVE* allows researchers to easily write custom made controllers or add new control features. Any planning algorithm, robot controller or sensing subsystem can be distributed and dynamically loaded at runtime thanks to *OpenRAVE* plugins. This frees developers from struggling with enormous code-bases. *OpenRAVE* users can focus their attention on the development, planning and scripting aspects of a problem without having to explicitly manage the intricacies of the underlying architecture such as kinematics and dynamics details, collision detection, world updates and robot control. The *OpenRAVE* architecture has a flexible interface that is possible to implement in conjunction with other popular robotics packages such as *ROS* [4]. This flexibility is possible due to the priority *OpenRAVE* gives to autonomous motion planning and high-level scripting rather than low-level control and message protocols. *OpenRAVE* also supports a robust network scripting environment, thereby simplifying robot control and modification of execution flow at run-time. Furthermore, the use of open component architecture such as *OpenRAVE* gives the robotics research community the freedom to easily share and compare algorithms. However, *OpenRAVE* does not provide any tools for an automated generation of robotic models. The onus is put on the designer, who must manually define the robot structure according to the guidelines expected by *OpenRAVE*. The manual design process may require an extensive effort for the designer in terms of time. This can be a significant disadvantage when considering the use of rapid virtual prototyping methods.

To tackle this problem, a flexible graphical tool for generating modular configurations, which can be simulated with *OpenRAVE*, was presented in [5]. The proposed tool consists of a GUI, which is based on the *OpenMR* plugin [6], an *OpenRAVE* modular robots extension that allows for simulating the locomotion of modular robots. The proposed GUI allows for easily defining the desired configuration of the robot to be simulated. This enables the user to get simulation results very quickly because only some basic knowledge about the application space of modular robotics is needed. Therefore, this interface is suitable for both educational and research purposes. However, the presented GUI is restricted to snake-like modular robots and it only allows locomotion simulation purposes.

To the best of our knowledge, a model generator tool for robotic modular hands or human like manipulators has not been released. The main contribution of this paper is to propose such a design tool.

## III. *OpenMRH*

In this section, a generalised modular model for modular robotic hands is first considered. Based on this model, we then summarise our previously presented design algorithm for the sake of clarity. Finally, a detailed overview of *OpenMRH* is provided.

### *A. A Generalised Modular Model for Modular Robotic Hands*

The same generalised modular model as proposed in [2] is adopted in this work. The concept is a modular device whose structure can be adapted to the object to grasp or to the task to be executed. To this end, the simplest mechanical structure was chosen, with the minimum number of actuators, simplest set of sensors, etc. These guidelines brought forth the *Y1* modular robot [7]. This low-cost, versatile and robust robot, which has one DOF and fast-prototyping features, was chosen as the fundamental element of the proposed modular device. Docking blocks allow for easy and flexible connection or disconnection. A standard servo motor actuates each joint and each module has the same assembly selection, making the modular structure as simple as possible. As shown in Fig. 1, the generalised modular model consists of one or more kinematic chains of modules fixed to a base, in which each module is a chain link. When compared to a human hand, each chain represents a finger, each module represents a phalanx and the base represents the palm. The base of the model is also modular. Each finger is attached to its own base plate module, which can be connected to the other base plates with the existing slots and hooks, forming a unique base. Three possible base configurations are defined:

*linear base* when there is no finger opposition, *circular base* when equidistant fingers are set in a circular configuration and *opposable-fingers base* when one or more fingers oppose the others. Although these modular bases do not cover all possible gripper configurations, they do describe the most significant grasp models that mimic human hand taxonomy [8].

### B. Modular Grasping Design Algorithm

The algorithm for our iterative design process is independent of the kind of modular robot adopted as the fundamental building block. For further details, the reader is referred to [2]. The following variables are used: $m(i)$, the modules used for the modular gripper at the $i$-th iteration (excluding base modules); $M$, the maximum number of modules per finger of the modular device; $M_{min}$, $M_{max}$, the lower-bound and upper-bound values of $M$; $R$, the radius of the minimum volume sphere that envelops the object to grasp; $L$, the length of one module; $f$, the total number of fingers; $f_{min}(i)$, the minimum number of fingers needed in the device at the $i$-th iteration; $x_j \in \mathbb{N}$, the number of modules of the $j$-th finger; $w$, the weight of a finger; $\tau_{max}$, the torque needed to overcome the moment due to the weight of the completely outstretched finger. $M$ is computed at the beginning of the algorithm based on the features of the module and the object to grasp. The lower-bound, $M_{min}$, is calculated as follows:

$$M_{min} = \left\lceil \frac{R}{L} \right\rceil . \tag{1}$$

$M_{min}$ takes the size of the object into consideration. In this preliminary study, this information is considered known in advance. The upper-bound, $M_{max}$, is calculated based on the worst-case scenario, a completely outstretched finger. In this situation, the maximum torque, $\tau_{max}$, of the module closest to the finger base, must overcome the moment due to the weight of the whole finger:

$$\tau_{max} > \frac{LMw}{2} \quad \Longrightarrow \quad M_{max} = \left\lfloor \frac{2\tau_{max}}{Lw} \right\rfloor . \tag{2}$$

$M$ is chosen as a trade-off between $M_{min}$ and $M_{max}$ in the initialisation phase of the algorithm and the finger configuration can be denoted as $\{x_1, x_2, ..., x_f\}$.

The goal of this algorithm is to obtain the most efficient configuration for a desired performance in terms of grasp quality with the minimum number of modules. Therefore, the resulting generated models do not necessarily show a human-like configuration. For simple objects or tasks, relatively simple manipulators may be sufficient to achieve the desired grasp. An evaluation of the grasp quality is therefore required. Grasp quality indices are known in related literature [9]. The quality criteria introduced by Ferrari and Canny [10] were used in our preliminary study and is also used in this work. However, other solutions can be implemented while maintaining the algorithm's structure. As a quality index, Ferrari and Canny considered the radius of the largest inscribed sphere centred at the origin and bound in the so-called *Grasp Wrench Space* (GWS). The GWS is the set of all wrenches capable of being resisted by a grasp when unit contact forces are applied at the contact points. It is given by the convex hull of the elementary wrenches:

$$GWS = ConvexHull\left(\cup_{i=0}^{n}\{w_{i,1}, \ldots, w_{i,k}\}\right), \tag{3}$$

where $n$ is the number of contact points and $k$ is the number of faces of the friction cone. The measure of the radius of the largest inscribed sphere centered at the origin that is contained in the *GWS* can be also seen as the magnitude of the largest worst-case disturbance wrench that can be resisted by a grasp with a unit strength grip. It will hereafter be denoted as $Q$, while the desired grasp quality will be denoted as $Q_{desired}$.

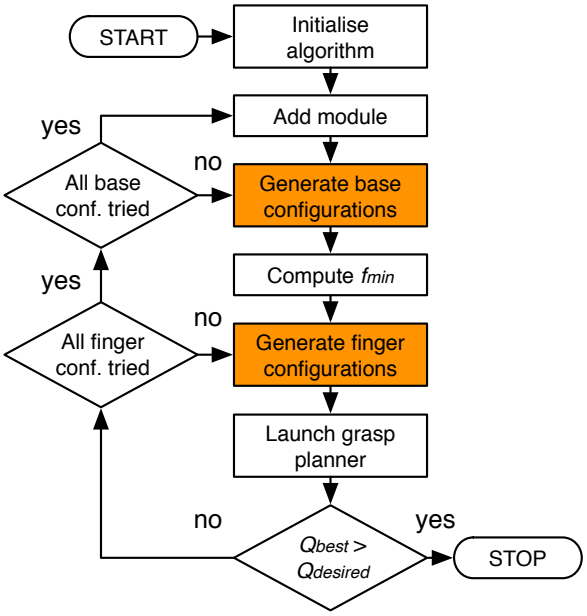



Fig. 2: The flowchart of the proposed efficient design method for modular grasping hands.

The flowchart of the proposed algorithm is shown in Fig. 2. The main iterative loop starts with the simplest modular configuration which consists of one finger with one module and one base plate. With each iteration, an additional module is added to increase the possible DOF. The number of modules for each finger is then set by selecting one among all the possible gripper configurations that can be obtained considering $m(i)$ modules. Consequently, a configuration for the modular base of the device is selected, depending on the number of fingers, among the set of all the predefined base configurations. Once a configuration is generated, a grasp planner is used to find the best grasp achievable. If the corresponding grasp quality is less than $Q_{desired}$ and all the possible finger configurations and base configurations achievable with $m(i)$ modules have been tested, a new iteration begins and one more module is added. In the following, the key steps of the algorithm are described.

*Initialise algorithm:* the shape and size of the target object are set. The values of $M$ and $Q_{desired}$ are assigned, $m(0)$ and $f_{min}(0)$ are initialised as $m(0) = f_{min}(0) = 1$.

*Generate base configurations:* the set of all possible base configurations (*linear base*, *circular base*, *opposable-fingers base*) is defined. Other base configurations could also be considered by simply adding those in the predefined set.

*Compute $f_{min}$:* at each iteration a module is added, so $m(i) = m(i-1)+1$. The value of $f_{min}$ has to be updated to avoid the case of more than $M$ modules per finger, so it can be defined as follows:

$$f_{min}(i) = \left\lceil \frac{m(i)}{M} \right\rceil . \quad (4)$$

*Generate finger configurations:* a new gripper configuration is generated. The algorithm does not generate all the configurations at the same time. Each configuration is tested and a new one is generated only if $Q_{best} < Q_{desired}$. Otherwise the algorithm returns the current version. This approach avoids testing unnecessary configurations.

*Launch grasp planner:* a grasp planner is used to determine the grasp quality achievable with each configuration for the given object. A grasp is simulated by setting an initial base position (pose) and initial joint angles (pre-shape) to the manipulator device. For each gripper configuration, fifty poses and pre-shapes are tested. This number is a parameter that can be manually set by the designer. A larger number of predefined poses and pre-shapes will result in more computation during the grasp planning phase. For each pose, the approach phase is realised by moving the device along the normal to the palm plane until it hits the target object. The fingers of the gripper then close around the object until they can not close any more. The contacts between the device and the object are extracted and the grasp quality index is calculated. By the end of this step, the best grasp that can be obtained with the current configuration is returned together with the corresponding initial base position.

*Stop condition:* The algorithm stops when the desired grasp quality is reached. The current modular configuration is efficient in the sense that it allows for reaching the desired grasp quality.

### C. Overview of OpenMRH

Even though the effectiveness of the summarised design algorithm was proved in [2], the highlighted steps shown in Fig. 2 were manually performed by the designer. The manual process of generating different modular configurations required a significant effort for the designer in terms of time. To overcome this challenge, *OpenMRH*, a plugin for *OpenRAVE* is presented that allows for a fast and automated generation of different modular hand models according to the previously proposed iterative design algorithm.

*OpenRAVE* uses the Extensible Markup Language (XML) [11] to store all robot and scene descriptions. The main modelling features and syntax guidelines are summarised in the following. For further details, the reader is referred to [3]. The XML format allows for file linking via inclusion, thus allowing the use of previously created objects or robots in the environment. A robot manipulator can be defined as a kinematic chain of its joint hierarchy when using the

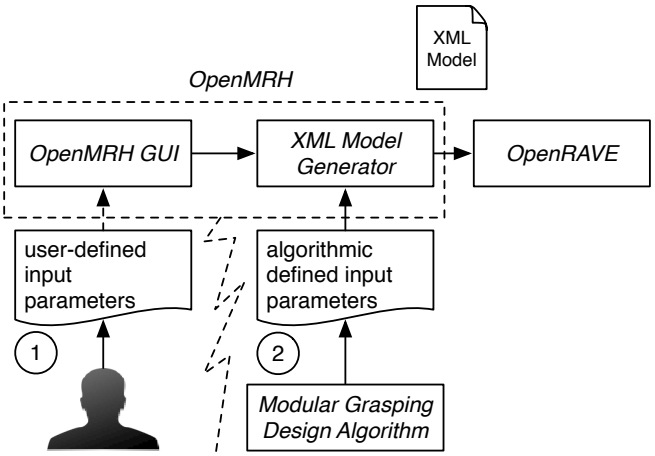



Fig. 3: The system architecture of *OpenMRH*.

XML format. The optional gripper joint values can also be included, even though they are not used in inverse kinematics calculations and are only needed for grasping purposes. Chains for heads and legs, for example, do not need joint values. A new frame of reference with respect to the end-effector link is defined by the manipulator to perform all inverse kinematics calculations. Moreover, it is possible to include a "direction" tag to specify the axis for the line-of-sight or for approaching objects. By using different XML tags, several interface types can be defined as follows:

- the *Environment* tag can be used to specify multiple robots and objects. Some GUI properties such as the camera's start location and background colour can also be defined with this tag. The *Environment* tag permits the creation of any *OpenRAVE* interface. Each interface has a type attribute to be used in specifying the interface type and defining custom XML readers;
- the *KinBody* tag can be used to define the basic object from which all other objects are derived. A collection of rigid bodies and connective joints make up a kinematic body;
- the *Robot* tag is a basic robot interface derived from the *KinBody* tag. Usually a *Robot* tag has at least one *KinBody* declaration inside it. A list of *Manipulator* and *AttachedSensor* objects can also be included within a *Robot* tag, to describe the robot's manipulation and sensing capabilities.

Even though this XML-based interface is quite flexible and intuitive, *OpenRAVE* does not provide any tools for an automated generation of robotic models. To overcome this issue and to improve our previously proposed design algorithm, we propose *OpenMRH*. The system architecture of *OpenMRH* is shown in Fig. 3. The proposed plugin essentially consists of two main components: the *OpenMRH GUI* and the *XML Model Generator*, which is the core of the system. It should be noted that the *XML Model Generator* can be launched not only through the proposed *OpenMRH GUI* with a user-defined input but also programmatically. This second possibility makes the integration with our pre-

viously proposed design algorithm possible. Referring to Fig. 3, the user-defined input and the algorithmic defined input approaches are labelled with the number 1 and 2, respectively.

*OpenMRH* is written in Java, thus making cross-platform support possible. In the following, the two key components are presented.

*1) OpenMRH GUI:* a configuration GUI is provided with *OpenMRH*, which allows user-defined input parameters. This interface guides the user through all the steps that are necessary to configure the modular robotic hand model to be simulated. The user can specify the following input parameters:

- the fundamental building module to be adopted. Specifically, the standard Open Inventor file format [12] is adopted to load the 3D models;
- the number of DOF;
- the base configuration. Three possible base configurations (*linear base*, *circular base*, *opposable-fingers base*) are considered according to the previously proposed generalised model;
- the distance between fingers. A discrete set of predefined lengths is considered;
- the finger configuration.

The configuration GUI is designed for users who are not necessarily familiar with the *OpenRAVE* simulation software. *OpenMRH* only requires very little knowledge about robotic grasping principle. This makes the model generation process easy from the user point of view. Once all the input parameters are set, the *XML Model Generator* starts the model generation process.

*2) XML Model Generator:* The core of *OpenMRH* consists of an *XML Model Generator*. This component makes it possible to automatically generate all the necessary XML files. The XML model generation process is shown in Fig. 4. Once the model properties are defined with either a user-defined input or programmatically, the *XML Model Generator* initialises the process of creating the corresponding XML document. The creation of the XML document is implemented in Java by using the XML Document Object Model (DOM) parser [13]. The XML DOM defines a standard for accessing and manipulating XML documents. The underlying idea is very simple. A DOM object with the desired tree structure is created, then the DOM object is written into a stream, in our case an XML file. The obtained XML file is is consistent with the standard syntax expected by *OpenRAVE* for the simulated models. In this way, once the model is generated, an instance of *OpenRAVE* can be launched and the model can be visualised within the selected simulation environment. The generated models can be used for testing, developing and deploying grasp or motion planning algorithms.

## IV. Case Studies

Two case studies are presented in this section to validate the efficiency of the proposed model generator. A modular hand model with 9 DOF is first generated with *OpenMRH*

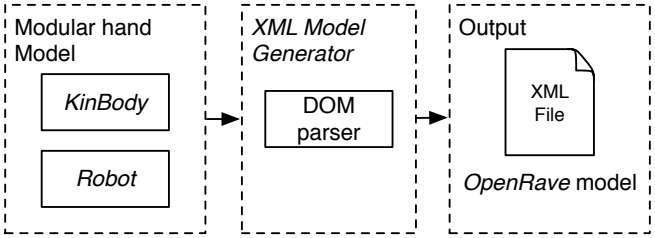



Fig. 4: The XML model generation process.

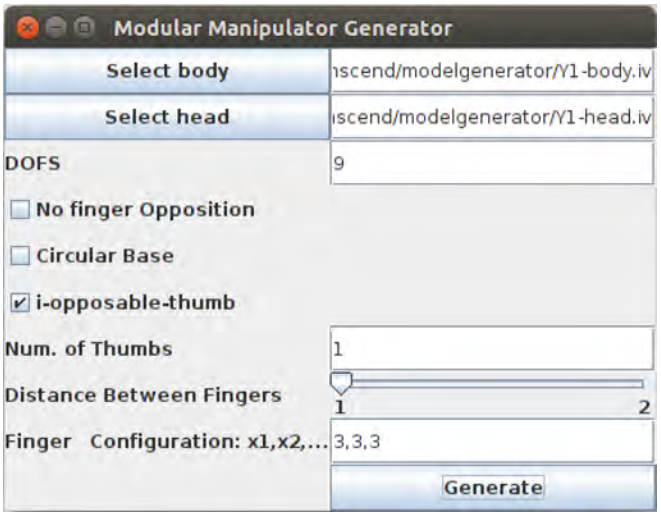


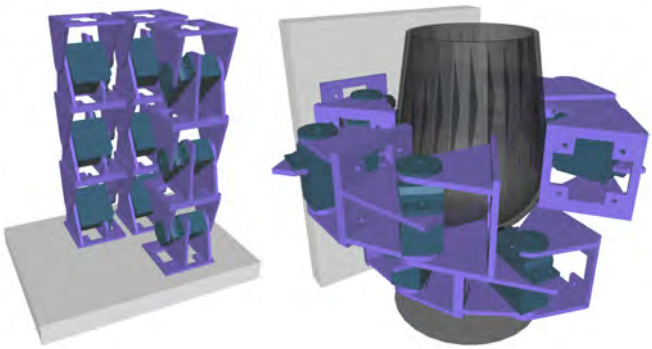

Fig. 5: The *OpenMRH GUI* and the user-defined input parameters adopted in the first case study.

by using a user-defined input. The hand consists of a 3 DOF thumb, which opposes the other two fingers, each having 3 DOF. The distance between fingers is equal to one discrete unit and the finger configuration can be described as $\{3,3,3\}$. The adopted input parameters for the *OpenMRH GUI* are shown in Fig. 5. The resulting generated model is shown in Fig. 6-a, while Fig. 6-b shows the same model grasping a glass.

Successively, a modular robotic hand is generated with *OpenMRH* by using algorithmic defined input parameters. Specifically, our design algorithm is used to find an efficient modular configuration to grasp a ketchup bottle. The maximum number of modules per finger $M$ is set to 3. According to the experimental results presented in [14], the quality threshold $Q_{desired}$ is set to 0.1 since this or a greater measure of quality corresponds to grasps that a human would consider stable. In Fig. 7, the resulting algorithm iterations and the corresponding required time of this experiment are shown. The first modular configuration able to reach the desired grasp quality is shown in Fig. 7-g.

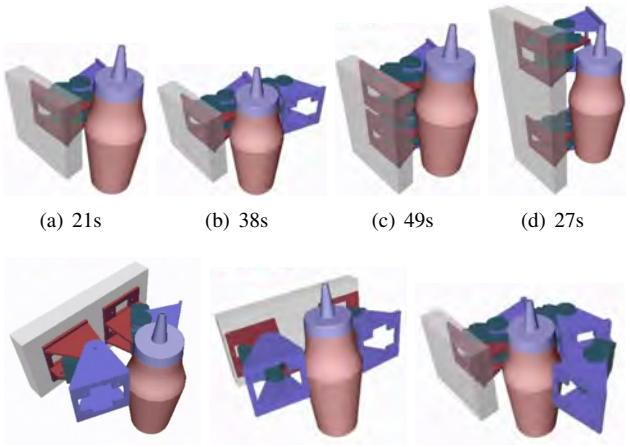
## V. Conclusion and Future Work

This paper, *OpenMRH*, an open-source plugin for *OpenRAVE* was presented that allows for a fast and automated generation of different modular hand models. The models can be generated either from a user-defined input by using the *OpenMRH GUI* or programmatically by adopting our previously presented design algorithm. The generality of the



(a) (b)

Fig. 6: (a) a model generated with *OpenMRH* by using user-defined input parameters, (b) the same model grasping a glass.



(a) 21s (b) 38s (c) 49s (d) 27s

(e) 34s (f) 37s (g) 47s

Fig. 7: From (a) to (g), the necessary algorithm iterations to find an efficient modular configuration to grasp a ketchup bottle. The first modular configuration able to reach the desired grasp quality is shown in (g).

proposed plugin makes it possible to use *OpenMRH* for many other educational and research purposes. In this sense, *OpenMRH* represents a useful extension for the *OpenRAVE* simulation environment. This extension opens up to a variety of possible application scenarios, making it feasible to develop alternative design approaches and control methods for modular robotic hands. The virtual-prototyping approach can easily be combined with the modular concept.

In the future, a possible improvement to *OpenMRH* could consist of adding the ability to also generate the manipulator environment so that multiple objects and robots can be added according to current needs. Another possible future work could consider the integration of the presented system with *ModGrasp* [15]–[17], an open-source virtual and physical rapid-prototyping framework developed by our research group. This integration would allow for testing different control algorithms. The possibility of exploring precision grasping approaches may be also considered by developing different quality indices and planners. The challenge of adapting the proposed method to a multi-objective optimisation problem may be studied. Finally, some effort should be put into the standardisation of *OpenMRH* to make it even more reliable for both the industrial and the academic practice. It is the opinion of the authors that the key to maximising the long-term, macroeconomic benefits for the robotics industry and for academic robotics research relies on the closely integrated development of open content, open standards and open source.

## REFERENCES


[1] K. Gilpin and D. Rus, "Modular robot systems," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 38–55, 2010.

[2] F. Sanfilippo, G. Salvietti, H. Zhang, H. P. Hildre, and D. Prattichizzo, "Efficient modular grasping: an iterative approach," in *Proc. of the 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), Rome, Italy*, 2012, pp. 1281–1286.

[3] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA), workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[5] D. Krupke, G. Li, J. Zhang, H. Zhang, and H. P. Hildre, "Flexible modular robotic simulation environment for research and education." in *Proc. of the 26th European Conference on Modelling and Simulation (ECMS), Koblenz, Germany*, 2012, pp. 243–249.

[6] J. Gonzalez-Gomez, A. Ranganath, and D. Estevez. (2010, January) OpenMR: Modular Robots plug-in for OpenRAVE. Last access on Sept. 2015. [Online]. Available: https://github.com/Obijuan/openmr.

[7] J. Gonzalez-Gomez, H. Zhang, E. Boemo, and J. Zhang, "Locomotion capabilities of a modular robot with eight pitch-yaw-connecting modules," in *Proceeding of CLAWAR*. Citeseer, 2006, pp. 12–14.

[8] M. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 269–279, 1989.

[9] D. Prattichizzo and J. Trinkle, "Grasping," in *Handbook on Robotics*, S. B. and K. O., Eds. Springer, 2008, pp. 671–700.

[10] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 1992, pp. 2290–2295.

[11] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml)," *World Wide Web Consortium Recommendation REC-xml-19980210. http://www.w3.org/TR/1998/REC-xml-19980210*, p. 16, 1998.

[12] P. S. Strauss and R. Carey, "An object-oriented 3D graphics toolkit," in *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, 1992, pp. 341–349.

[13] W3Schools. (2015, January) XML DOM Tutorial. Last access on Sept. 2015. [Online]. Available: http://www.w3schools.com/dom/.

[14] C. Goldfeder, P. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. Citeseer, 2007, pp. 10–14.

[15] F. Sanfilippo, H. Zhang, K. Y. Pettersen, G. Salvietti, and D. Prattichizzo, "ModGrasp: an open-source rapid-prototyping framework for designing low-cost sensorised modular hands," in *Proc. of the 5th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), São Paulo, Brazil*, 2014, pp. 951–957.

[16] F. Sanfilippo, H. Zhang, and K. Y. Pettersen, "The new architecture of ModGrasp for mind-controlled low-cost sensorised modular hands," in *Proc. of the 2015 IEEE International Conference on Industrial Technology (ICIT2015), Seville, Spain*, 2015, pp. 524–529.

[17] F. Sanfilippo, "Alternative and flexible control approaches for robotic manipulators: on the challenge of developing a flexible control architecture that allows for controlling different manipulators," Ph.D. dissertation, Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, Department of Engineering Cybernetics, Trondheim, June 2015.