A Task-Driven Algorithm for Configuration Synthesis of Modular Robots

Esra Icer, Andrea Giusti, and Matthias Althoff

Abstract— **This paper presents a time-efficient, task-based configuration synthesis algorithm for modular robot manipulators. One of the main challenges in modular manipulators is to find possible combinations of modules that are able to complete given tasks while avoiding obstacles in the environment. Most studies on modular robots focus on obtaining combinations of modules to achieve a given task without considering the required path planning in an environment with obstacles. In contrast to previous works, we present a configuration synthesis method for modular manipulators, considering collision detection and path planning in task space. Our simulations show that our approach finds possible combinations with reduced computational time compared to previous techniques.**

I. INTRODUCTION

Although industrial robots generally offer high robustness, accuracy, performance and cost efficiency, it is difficult to adapt them to different tasks or environments. Purchasing new robots for each task in a flexible manufacturing scenario makes the manufacturing processes expensive. As an affordable solution, modular robots enable us to configure various robotic structures from several pre-designed modules, typically consisting of link modules and joint modules, so that they can be used for different tasks. Modular robots have been studied for the last 30 years [1]; their unique properties such as high versatility, easy maintenance, flexibility and robustness make them a promising technology for future flexible manufacturing.

One of the main challenges for adopting modular robots is the problem of determining the optimal combination of modules for given tasks. Those tasks can be specified by carrying a pre-determined load from a given initial position to a given goal position, passing through intermediate locations without colliding with any obstacle in the environment. In this work, we propose a time-efficient algorithm to select proper modules from a set of possible modules, such that aforementioned tasks are fulfilled. Obviously, our approach is not only useful for reassembling a modular robot, but also for initially setting up a robot. To the best knowledge of the authors, this is the first study which determines the set of possible configurations of modular robots for a given path planning task in one consistent approach. Task-based configuration synthesis of modular robots is performed in this work by automatically finding possible configurations for given modules. Our synthesis method is simple and utilizes automatic kinematic model generation of each configuration, sequentially discarding the configurations which cannot fulfil the given task.

The authors are with the Department of Computer Science, Technische Universität München, 85748 Garching, Germany. Corresponding e-mail: {icer, giusti, althoff}@in.tum.de

Among the previous works on configuration synthesis, a modular robot assembly methodology is presented in [2] to obtain non-isomorphic, serial or tree-like configurations. In that work, identical cube or prism modules that have multiple connection ports are used and an enumeration algorithm is introduced to remove redundant configurations resulting from symmetries. A minimized degrees-of-freedom (DOFs) approach is introduced in [3] to find the task-based optimal configuration of modules using the enumeration algorithm in [2]. A configuration synthesis methodology for task-based fault tolerant modular manipulators considering kinematic task requirements is proposed in [4]. However, their methodology mainly depends on defining a penalty function for less-likely combinations. A hierarchical modular design approach is presented in [5] which is based on selecting functional modules without considering kinematics and joint limits. A genetic algorithm (GA)-based configuration synthesis algorithm to find the optimal configuration for given end effector positions is introduced in [6] and the objective function is defined only regarding initial position and goal position of the end effector without considering obstacles and payloads. A two-level GA method is proposed in [7] to obtain the kinematic design of serial modular manipulators considering task specifications and environmental constraints like obstacles. Nevertheless, they consider all configurations without any elimination of possible compositions in the optimization process and define penalty functions to cancel less-likely configurations.

Most studies on the kinematic model of modular robots focus either on the Product-of-Exponentials (POE) method [2], [3] or the Denavit and Hartenberg (D-H) convention [8], [9]. Our work is based on the D-H approach as in [10] to automatically generate kinematic models for evaluating possible configurations.

As the pre-defined task mentioned earlier requires path planning in an obstacle-laden environment, collision-free path planning algorithms are required. Studies on collision-free path planning are mainly based on two approaches: *i*) the configuration space (C-space) approaches [11]–[14], and *ii*) task space approaches [15]–[19]. C-space approaches require a transformation of the obstacles from task space to C-space which makes the process time-consuming. This approach is not suitable for our problem due to the combinatorial complexity of possible compositions from a given set of modules. Since our goal is to find collision-free paths for a vast amount (typically several thousands) of unique kinematic chains, an algorithm capable of checking collision-free paths in a short period of time is needed. As a consequence, the methods that require a mapping from task space onto C-space cannot be used, which is the motivation behind this work. We focus on task space solutions due to their simplicity and computational time-efficiency.

In this paper, we present an algorithm to efficiently find compositions that can reach goal positions without any collision, from a given set of modules. Compositions are varied by changing the combination of different modules considering task requirements. In contrast to previous works, we *i*) consider heterogeneous modules, *ii*) find an individual solution for each configuration considering task requirements, *iii*) implement a quick obstacle avoidance algorithm in the task space, and *iv*) present a computationally-efficient configuration synthesis algorithm, which eliminates failed configurations step by step.

The paper is organized as follows: Sec. II explains the combinatorial problem for modular robots. Sec. III describes our configuration synthesis method, followed by implementation details in Sec. IV. Finally, conclusions are drawn in Sec. V.

II. Problem Statement

We consider modular and serially connected manipulators, whose kinematics are uniquely determined by a vector $\mathbf{q} \in \mathbb{R}^n$ of joint positions (angles for revolute joints and translations for prismatic joints), where n is the number of DOFs. The task requirements are only constrained by the kinematic model of the manipulator and static obstacles in the environment. We consider four types of modules that are categorized into bases, joints, links and end effectors. We work under the following assumptions to obtain the possible compositions: *i*) all modules have one input and one output connection port, *ii*) all manipulators begin with a base module and end with an end effector module, and *iii*) there is only one base and only one end effector module in all compositions. These are fairly practical assumptions for industrial robots which hold for many realistic situations. Without loss of generality, we refer to each possible composition with a unique number $k \in \mathbb{N}^+$ and the maximum number of possible compositions without considering task-specific constraints is N.

The environment consisting of the robot and obstacles is denoted by $\mathcal{W} \subset \mathbb{R}^3$. The subset of the space occupied by the robot is indicated by $\mathcal{A} \subset \mathcal{W}$ and the occupancy of a robot for a specific joint position vector $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q}) \subset \mathcal{W}$. The space occupied by the k^{th} configuration is denoted by $\mathcal{A}(\mathbf{q}_k(t)) \subset \mathcal{W}$. The obstacles are represented by arbitrary geometric shapes in $\mathbb{R}^3$, the occupancy of the j^{th} obstacle in the workspace is denoted by $\mathcal{O}_j \subset \mathcal{W}$ and the union of all obstacles is presented by $\mathcal{O} = \bigcup_j \mathcal{O}_j$.

Variable t is defined as $t = t_{cur}/t_f$, where t_{cur} is the current time and t_f is the final time to reach the goal. The function $\mathbf{q}(t)$ maps normalized time t to the joint position vector, where the initial time is assumed to be zero. The forward kinematics from the joint position vector $\mathbf{q}(t)$ to the end effector pose is denoted by $f(\mathbf{q}(t))$. It is assumed that all configurations start from a given set of initial pose $\mathcal{P}_s \subset \mathbb{R}^6$. $\mathcal{P}_s$ is defined as $\mathcal{P}_s = \{f(\mathbf{q}(0)) \mid \mathbf{q}(0) \in \mathcal{Q}_0\}$, where $\mathcal{Q}_0$ is the set of initial joint values. The final joint pose vector is indicated by $\mathbf{q}(1)$ and the set of goal poses is defined as $\mathcal{P}_g \subset \mathbb{R}^6$. $\mathcal{P}_g$ is defined as $\mathcal{P}_g = \{f(\mathbf{q}(1)) \mid \mathbf{q}(1) \in \mathcal{Q}_1\}$, where $\mathcal{Q}_1$ the set of goal joint values.

Variable $\boldsymbol{\tau}(\mathbf{q}(t))$ represents the static torque-force vector of the composition at time t. The set of combinations that fulfill the given task can be described as:

$$\begin{aligned} \kappa = \{k \;\; | \;\; & 1 \leq k \leq N \wedge \forall t \in [0,1] \;\exists\; \mathbf{q}_k(\cdot) : \\ & \mathcal{A}(\mathbf{q}_k(t)) \cap \mathcal{O} = \emptyset \\ & \wedge \;\; \mathbf{q}_k(t) \in \;\; [\mathbf{q}_{k,min},\, \mathbf{q}_{k,max}] \\ & \wedge \;\; |\boldsymbol{\tau}_k(\mathbf{q}_k(t))| \leq \boldsymbol{\tau}_{k,max} \\ & \wedge f_k(\mathbf{q}_k(0)) \in \mathcal{P}_s \;\; \wedge \;\; f_k(\mathbf{q}_k(1)) \in \mathcal{P}_g\} \end{aligned} \tag{1}$$

where $\mathbf{q}_{k,min}$ and $\mathbf{q}_{k,max}$ are the minimum and maximum joint position limit vectors and $\boldsymbol{\tau}_{k,max}$ is the maximum torque-force vector of the k^{th} composition, respectively. We face the problem to find the set of compositions κ that fulfill the described task in an efficient way.

III. Proposed Method

A large number of possible compositions can be generated by varying modules and changing the assembly of the modules. This large design space makes modular robot synthesis generally a complex and time-consuming combinatorial problem. The main idea behind our proposed configuration synthesis method is to go from simple task requirements to more complicated ones and eliminate unfeasible compositions as early as possible. The required tests are performed starting from simpler ones to more complicated ones. As a result, the compositions that cannot achieve the pre-defined task can be discarded by simple tests, whereas tests requiring longer time are only applied to the remaining compositions.

In the modular robot structure that we consider, each joint module introduces one DOF, whereas each link module and the fixed base module introduce no DOF. To restrict the search space, we use the pre-determined structure:

Base - Joint - Link - Joint - · · · - Link - End Effector.

Starting from the base module, all possible robot compositions are generated until the desired number of DOFs is reached considering the assumption mentioned above. The number of the joint modules is equal to the number of the link modules in the structure. The number of all compositions up to n DOFs from a set of modules can be obtained from (2), where $\tilde{b}$, $\tilde{j}$, $\tilde{l}$, $\tilde{e}$ denote the number of the different base modules, joint modules, link modules and end effector modules, correspondingly. The variable x represents the total number of DOFs of the end effector module in the structure and N denotes the maximum number of possible compositions.

$$N = \sum_{i=1}^{n-x} \tilde{b}^1 \cdot \tilde{j}^i \cdot \tilde{l}^i \cdot \tilde{e}^1 \tag{2}$$

The schematic representation of our proposed approach is presented in Fig. 1. The method consists of three sequential steps: *A*) generation of possible configurations that are feasible in the initial and the goal positions without considering obstacles in the environment, *B*) path planning without consideration of obstacles for the configurations remaining from (*A*), and *C*) path planning including collision detection for the configurations remaining from (*B*).

A. Generation of the Compositions with Feasible Initial Positions and Goal Positions

To quickly discard impossible compositions, we apply the tests given in Fig. 1, where κ_A refers to the set of compositions that can pass test *A*. We first check if the combined length of the modules can reach the initial and the goal positions to obtain the remaining set of feasible compositions κ_{length}. Next, we check whether joint angles at the initial and the goal positions are within the joint limits and the remaining set of feasible compositions resulting in $\kappa_{\text{kinematics}}$. Finally, the static torque-force limit test is applied to the remaining compositions to further reduce the set of feasible compositions to κ_{static}. Each of these sub-problems is checked sequentially as shown in Fig. 1, such that steps 4 and 5 are performed only on the remaining compositions. The total number of the compositions is reduced in each step such that $N_{\text{static}} \leq N_{\text{kinematics}} \leq N_{\text{length}}$, where N_{A}=$|\kappa_A|$ and $|\kappa_A|$ is the cardinality of κ_A.

1) Reachable Compositions: The maximum length of the composition is obtained from (3), where l_B, $l_{J,i}$, $l_{L,i}$, and l_E are the lengths of the base module, the i^{th} joint module, the i^{th} link module and the end effector module, respectively.

$$l = l_B + l_E + \sum_{i=1}^{n-x} (l_{J,i} + l_{L,i}) \tag{3}$$

It should be noted that for this test we use the maximum length of the prismatic joints. The base and the goal positions for a robot are denoted by P_b and P_g, respectively. The minimum distance between these two points in task space is $d_{b2g} = \|P_g - P_b\|_2$, where $\|P_g - P_b\|_2$ refers to the Euclidean distance between P_g and P_b. In this test, configurations which cannot satisfy the condition $l \geq d_{b2g}$ are eliminated.

2) Kinematics: Considering a modular and reconfigurable structure, the kinematic model is generally different for various modules and configurations of the robot. The manual derivation of all the possible kinematic models is not practical, especially when the number of different modules increases. To address this challenge, automatic kinematic modeling methods have been developed (see [10], [20]–[23]). As previously introduced, our kinematic modelling approach is based on the D-H convention proposed in [10]. The relative transformation of frames of subsequent links T_i^{i-1} is obtained using homogeneous transformation matrices and the four D-H parameters a_i, α_i, d_i, θ_i, which are explained in [24]. Each module is first characterized with parameters that are used to synthesize the forward kinematics using an extension of the standard D-H convention. When analyzing the configuration, the parameters of the links between each pair of joints are automatically synthesized using the information of the modules.

In order to map a desired motion from the task space to the joint space, the inverse kinematics is required. If this problem has admissible solutions, they can be multiple or even infinite (redundant manipulators) [24]. To address these difficulties, iterative non-linear optimization techniques are applied [25]. To map a desired pose in the workspace for the end effector and the positions of the joints to the joint space, the unit-quaternions-based scheme in [26] is used to obtain inverse kinematics.

3) Static Forces: Every joint of a manipulator has a static torque-force capacity to carry. Forces applied to the end effector affect the whole structure and are distributed across all joints. To find the torque-force values applied to each joint resulted from the payload, the wrench vector, $\mathbf{F} \in \mathbb{R}^6$, as defined in (4) is used, where $\mathbf{f} \in \mathbb{R}^3$ and $\mathbf{n} \in \mathbb{R}^3$ refer to the force and the moment vectors, applied to the end effector, respectively.

$$\mathbf{F} = \begin{bmatrix} \mathbf{f} \\ \mathbf{n} \end{bmatrix} \tag{4}$$

Static torque-force values of each joint are obtained from (5), where $\boldsymbol{\tau}(\mathbf{q}(t))$ is the static torque-force vector and $\mathbf{J}(\mathbf{q}(t))^{\text{T}}$ is the transpose of the Jacobian matrix at time t.

$$\boldsymbol{\tau}(\mathbf{q}(t)) = \mathbf{J}(\mathbf{q}(t))^{\text{T}}\, \mathbf{F} \tag{5}$$

We apply $\mathbf{f}$ and $\mathbf{n}$ to the end effector and calculate all torque-force values applied to joints via (4) and (5). The results for each joint obtained from (5) must satisfy the

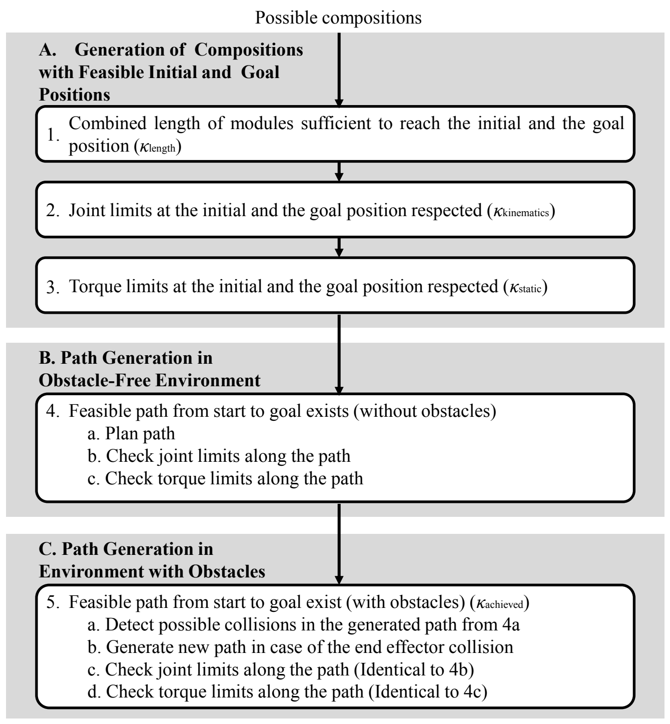

Possible compositions
A. Generation of Compositions with Feasible Initial and Goal Positions
1. Combined length of modules sufficient to reach the initial and the goal position (κ_{length})
2. Joint limits at the initial and the goal position respected ($\kappa_{\text{kinematics}}$)
3. Torque limits at the initial and the goal position respected (κ_{static})
B. Path Generation in Obstacle-Free Environment
4. Feasible path from start to goal exists (without obstacles)
a. Plan path
b. Check joint limits along the path
c. Check torque limits along the path
C. Path Generation in Environment with Obstacles
5. Feasible path from start to goal exist (with obstacles) (κ_{achieved})
a. Detect possible collisions in the generated path from 4a
b. Generate new path in case of the end effector collision
c. Check joint limits along the path (Identical to 4b)
d. Check torque limits along the path (Identical to 4c)

Fig. 1. Composition synthesis procedure.

condition $\boldsymbol{\tau}(\mathbf{q}(t)) \leq \boldsymbol{\tau}_{max}$ where $\boldsymbol{\tau}_{max}$ is the maximum torque-force vector of the manipulator.

B. Path Generation in Obstacle-Free Environment

Most of the path planning approaches for manipulators are based on joint space path planning due to its efficiency and singularity avoidance [24]. When assuming obstacle-free environments, we also plan the path in joint space, while the task space approaches are used for collision detection. In order to achieve joint space path planning, initial and goal points are mapped onto joint space using inverse kinematics. This step transforms the path planning problem from task space into C-space. A cubic polynomial function is considered to obtain a smooth path due to its simplicity [24]. Configurations are checked kinematically and statically at various points of normalized time t to see whether all points in time satisfies the minimum and the maximum limits.

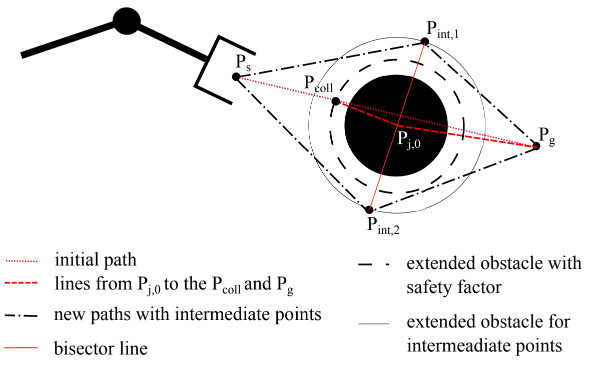
C. Path Generation in Environments with Obstacles

Path generation in an obstacle-laden environment requires collision detection and an obstacle avoidance algorithm. Since the projection of obstacles into C-space has an exponential complexity in the number of DOFs [19], a collision detection algorithm in task space is preferred in this study. Moreover, the projection of obstacles onto C-space is different for each composition and therefore it is difficult to implement C-space collision detection algorithms for our problem. Collision detection in task space depends mainly on two sub-problems: obstacle representation and collision detection of all components of the robot. For the sake of simplicity, many of the obstacle representation methods in task space use spherical representations [18]. When using a spherical representation, an obstacle is enclosed by the smallest sphere regardless of its shape; as a result, the representations of obstacles are larger than the real obstacles and the volume of obstacle-free spaces in the environment is smaller than the real obstacle-free spaces.

We perform collision checking of *1*) the end effector, *2*) the joints, *3*) the links, and *4*) the robot itself (self collision). All joints and links of the manipulator are assumed to be points and lines, respectively, and their thicknesses are added to the radius of the obstacles.

1) Collision of the end effector: The forward kinematics of the robot configuration explained in Sec. III-A.2 is employed to determine the end effector position of the n-DOF manipulator from the joint values. When we obtain the position of the end effector, distances between the position of the end effector and the center of the obstacles, d_{E2O}, must satisfy $d_{E2O_j} \geq r_{j,O} + r_E$ where $r_{j,O}$ is the radius of the corresponding obstacle and r_E is the radius of the end effector.

2) Collision of the joint: The coordinates of each joint in task space are found by subsequent multiplication of the homogeneous transformation matrices of the D-H frames and an additional translation for the i^{th} link defined as n_i in [10]. Checking the collisions between joints and obstacles, the distance between the mid-point of the joints and the center



P_s
P_coll
P_int,1
P_int,2
P_j,0
P_g
initial path
lines from P_j,0 to the P_coll and P_g
new paths with intermediate points
bisector line
extended obstacle with safety factor
extended obstacle for intermeadiate points

Fig. 2. Schematic diagram of the obstacle avoidance algorithm.

of obstacles, d_{J2O}, must satisfy the condition $d_{J_i 2O_j} \leq r_{j,O} + r_{J,i}$, where P_i is the mid-point of the i^{th} joint and $r_{J,i}$ is the radius of the i^{th} joint.

3) Collision of links: We consider each link as a cylinder and model them as a line to simplify the calculations. Variables P_{i,L_i} and P_{i,L_e} denote the initial and the final positions of the i^{th} link, respectively. The line segment between these two positions refers to the i^{th} link of the manipulator. As a result, d_{L2O}, the distance between the link and the obstacle is calculated from the point-line distance formula as in [27]. Each link should satisfy the condition $d_{L_i 2O_j} > r_{j,O} + r_{i,L}$ for collision-free path, where $r_{i,L}$ is the radius of the i^{th} link of the manipulator.

4) Self Collision of the Manipulator: Since all links are modelled as a line, variable $d_{L_i 2L_m}$, which refers to the distance between the i^{th} and the m^{th} link, is computed from the line segment to line segment distance formulation as in [27]. Each link should satisfy the condition $d_{L_i 2L_m} > r_{i,L} + r_{m,L}$ for all combinations, where $r_{i,L}$ and $r_{m,L}$ are the radius of the i^{th} and the m^{th} link, correspondingly.

5) Obstacle-Avoiding Path Planning: Our obstacle avoiding path planning approach is mainly based on vectors from the initial position to the goal position to generate intermediate points as in [28]. Based on the proposed approach, a path is generated from the initial point to the goal point using cubic polynomials in joint space and collision is checked in task space for each component as explained in this section. In case only the end effector collides with obstacles, a set of intermediate points, $\mathcal{P}_{int}$, are generated. To generate $\mathcal{P}_{int}$, the collision point P_{coll}, the goal point P_g and the center of the obstacles $P_{j,O}$ are used (see Fig. 2). For the collision of the end effector, two lines are generated in task space: one line is between P_{coll} and $P_{j,O}$; the other one is between P_g and $P_{j,O}$ where $P_{j,O}$ is the center of the j^{th} obstacle. Using these two lines, a new line passing through $P_{j,O}$ is created which is equiangular to these lines. The main differences between the proposed approach [28] and our work are that the collision check is done in task space, the intermediate points are generated using bisectors and the obstacles are considered as spheres. The obstacle is extended to prevent a collision and the set of intersection points between the generated line and extended obstacle are defined as $\mathcal{P}_{int}$. The set of distances $\mathcal{D}_{pnt}$, between the $\mathcal{P}_{int}$ and P_g is calculated as follows:

$$\mathcal{D}_{pnt} = \{\, \|P_g - P_p\|_2 \,|\, p \in \{1,2\}, P_p \in \mathcal{P}_{int}\} . \quad (6)$$

The manipulator initially follows the closest point of $P_{int,1}$ and $P_{int,2}$, and a new path is generated using this as an intermediate point. Using inverse kinematics the predetermined intermediate point is mapped onto C-space and a new path, as shown in Fig. 2, is generated to reach the goal position. According to the aforementioned assumptions, the obstacle avoidance algorithm is applied only in case of the end effector collision. For the other collision types, the module compositions are removed because implementing the collision avoidance algorithm to all collision types is time-consuming. This procedure is repeated until the end-effector reaches P_g.

IV. Simulations

To demonstrate our configuration synthesis algorithm, we implement it in MATLAB R2015b running on an Intel® Core™ i7-4712 2.30 GHz CPU equipped with 16 GB RAM. To compare the results, we use one type of base module, two types of 1-DOF joint modules (revolute (R) and prismatic (P)), three types of no DOF link modules ($\alpha_1 = 90°$, $\alpha_2 = 0$, $\alpha_3 = 90°$) and two types of one DOF end effector modules (revolute (R) and prismatic (P)) in the robot structure (see Fig. 3). We follow the composition structure defined in Sec. III without considering the orientations. The base is positioned in point $P_b = (0,\ 0,\ 0)$. We consider serially connected, modular manipulators with degrees of freedom varying from 2 to 6. The type of the modules, the number of the modules and the assembly of the modules are the three main design parameters of our combinatorial problem. Moreover, the types of the joints and the links, position of the joints, the position and the torque-force limits of the joints, and the length of joints and the links are design parameters for each module (see Tab. I).

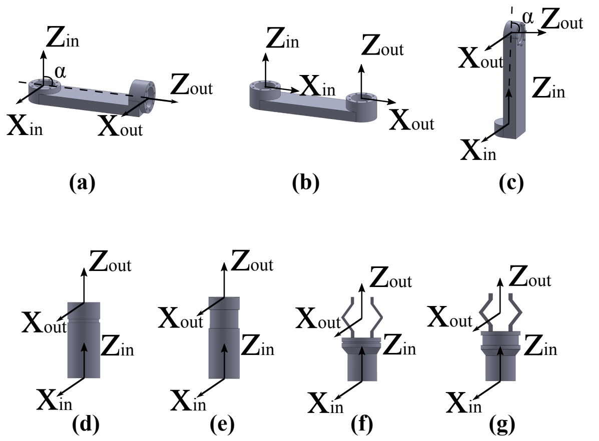


Fig. 3. Basic modules used in this study, (a) $\alpha = 90°$ link module, whose length is along the y direction of the previous coordinate system (L1), (b) $\alpha = 0°$ link module (L2), (c) $\alpha =\ 90°$ link module, whose length is along z direction of the previous coordinate system (L3), (d) rotational joint module (J1), (e) prismatic joint module (J2), (f) rotational end effector module (E1), (g) prismatic end effector module (E2)

All possible configurations from 2-DOF to 6-DOF are computed and 18660 different compositions are obtained. The initial and the goal positions are defined as $P_s = (1.5,\ 1.5,\ 2)$ and $P_g = (-2,\ -1.2,\ 0.6)$, respectively. As a first step, κ_{length} is generated and 18576 over 18660 combinations can reach the initial and the goal positions in terms of length. After that, $\kappa_{\text{kinematic}}$ is obtained by applying the kinematic check test. Considering a 50 N payload in -z direction, κ_{static} is obtained. Each test is applied to all remained compositions sequentially and the total number of feasible compositions is decreased in each step (see Tab. II).

After generating the feasible compositions of the initial and the goal positions, we apply the cubic polynomial path planning method as discussed in Sec. III-C to find a path between the initial position and the goal position. Moreover, the kinematic test and the static test are also applied to the intermediate points, to determine whether the joint values and the applied torques-forces are within the limits.

As the next step, stationary obstacles in the environment are taken into account in the configuration synthesis. Random and complex-shaped obstacles are generated. Spheres, which enclose each obstacle with the minimum radius, are generated to represent the obstacles as stated in Sec. III-C. The center of the spheres are located at the coordinates of $P_{1,O} = (0.5,\ 0.2,\ 1.3)$ and $P_{2,O} = (-1.2,\ 2.1,\ 0.8)$. Radii of the obstacles are defined as $r_1 = 0.3$ m and $r_2 = 0.35$ m (see Fig. 4).

The obstacle avoidance algorithm is applied only for the end-effector collision. To implement this algorithm, total time is divided into time-step values where $t \in [0, 1]$, the kinematic test and the static test are also applied to the generated intermediate points. The simulation time increases with the increase in the number of the time-step values. After $t = 0.004$, it is seen that the number of possible compositions is almost the same although the simulation time increased. The normalized time value t depends on the Euclidean distance between P_s and P_g, the radii of the obstacles in the environment and the extension value of the obstacles.

TABLE I

The module parameters involved in simulation (see Fig.3)

	Length [m]	Diameter [m]	Max τ [Nm]	Joint Limits [rad or m]
L1	0.75	0.2	-	-
L2	0.75	0.2	-	-
L3	0.75	0.2	-	-
J1	0.25	0.2	80	[-π,π]
J2	0.25	0.2	75	[0,0.2]
EE1	0.2	0.2	75	[-π,π]
EE2	0.2	0.2	70	[0,0.1]

TABLE II
REMAINING NUMBER OF COMPOSITIONS AFTER APPLYING EACH TEST

	Number of Feasible Comp.	Time [s]
Generation of Feasible Comp.		
κ_{length}	18576	3.2
$\kappa_{\text{kinematic}}$	247	736
κ_{static}	152	749
Path Generation Obs. Free Env.		
$\kappa_{\text{kinematic}}$	152	751
κ_{static}	60	809
Path Generation in Envi. with Obs.		
$\kappa_{\text{achieved (proposed method)}}$	44	983
$\kappa_{\text{achieved (TS-RRT)}}$	28	8584

We compare our method against rapidly-exploring random trees (RRT) approach, which is a popular path planning approach due to their fast convergence and high success probability [13]. We implement the task space RRT (TS-RRT) algorithm in [19], where random points are generated in a task space to find a collision-free path from the initial position to the goal position. In [19], collision detection is done in C-space using the pre-determined C-space configuration although the random points are generated in task space. Mapping all the obstacles onto C-space is not computationally efficient for our particular problem; therefore, collision detection is done in task space.

The set of random points, $\mathcal{P}_{rand}$, are generated directly in the task space. Using the inverse kinematics algorithm, the joint values of the robot are obtained and the configuration is checked for collisions as explained in Sec. III-C. The parameters of the TS-RRT algorithm are: the maximum displacement at each iteration (*step size*) and the maximum number of iterations (*max number of trees*). The *step size* is 0.05 as a reasonable compromise between efficient computation and collision detection and the *max number of trees* is 500. The comparison of the two algorithms is shown in Tab. II and in Fig. 4. Utilizing the selected parameters accurately, our method has more possible solutions compared to the TS-RRT algorithm. The possibility of finding an obstacle-free path in TS-RRT algorithm is higher when the *max number of trees* is increased and the *step size* is decreased, but at the same time, the computational time is increased. Since our goal is to obtain possible combinations in the quickest and the most accurate way, our proposed path planning method is preferable. In addition, the computational time of the TS-RRT algorithm is affected by the limits of the search space. In our problem, we cannot restrict the search space of the TS-RRT with the most effective limits because the workspace of the manipulator is varied in every different composition. The limits of the TS-RRT are assumed as the sum of the each module's length. Moreover, paths obtained from the proposed method are smoother than TS-RRT (see Fig. 4) because the new points are generated randomly in RRT algorithms.

V. CONCLUSION

In this paper, a time-efficient, task-based composition synthesis method for modular manipulators is proposed. The sequential composition synthesis algorithm is used to decrease the possible compositions considering the given task requirements step by step to reduce the computational effort. We assume that the given task has the following requirements: reachability, joint limits, static limits, collision detection and obstacle avoidance. Infeasible compositions

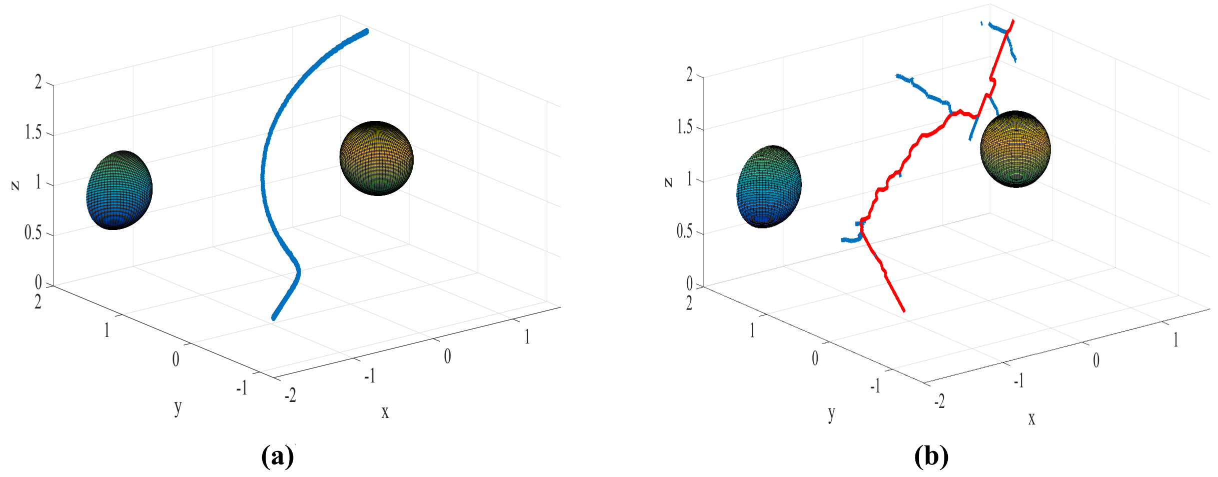


Fig. 4. Comparison of the paths generated from two methods: (a) the possible path obtained from the proposed method, (b) the possible path obtained from the TS-RRT algorithm where red line indicates the possible path and blue lines are the branches of the search tree.

are eliminated by task requirement tests starting from less time-consuming ones to most time-consuming ones. As a result, tests taking longer time periods are only applied to the remaining compositions, which reduces the computational time. Path planning is performed using cubic polynomials in joint space but the obstacle avoidance algorithm is performed in task space using forward and inverse kinematic models of the manipulators. To measure the efficiency of the path planning algorithm, compositions of robots with up to 6-DOFs are obtained and compared with the TS-RRT algorithm. It is observed that the proposed algorithm needs less computational time and provides more accurate results. The main advantages of the proposed composition synthesis algorithm are *i*) it is applicable to heterogeneous modules, *ii*) it can be applied to different types of joint or link modules, *iii*) it avoids repeated compositions, and *iv*) it gives all possible compositions for the pre-defined task in the computationally efficient way.

Acknowledgment

The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/2007- 2013/ under REA grant agreement number 608022.

References

[1] K. Wurst, "The conception and construction of a modular robot system," in *Proc. 16th Int. Sym. Industrial Robotics (ISIR)*, 1986, pp. 37–44.

[2] I.-M. Chen and J. W. Burdick, "Enumerating the non-isomorphic assembly configurations of modular robotic systems," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 702–719, 1998.

[3] G. Yang and I.-M. Chen, "Task-based optimization of modular robot configurations: minimized degree-of-freedom approach," *Mechanism and machine theory*, vol. 35, no. 4, pp. 517–540, 2000.

[4] C. J. Paredis and P. Khosla, "Synthesis methodology for task based reconfiguration of modular manipulator systems," in *Proc. of the 6th International Symposium on Robotics Research, ISRR*, 1993.

[5] S. Farritor, S. Dubowsky, N. Rutman, and J. Cole, "A systems-level modular design approach to field robotics," in *International Conference on Robotics and Automation*, 1996, pp. 2890–2895.

[6] W. Gao, H. Wang, Y. Jiang, and X. Pan, "Task-based configuration synthesis for modular robot," in *Proc. International Conference on Mechatronics and Automation (ICMA)*, 2012, pp. 789–794.

[7] O. Chocron and P. Bidaud, "Evolutionary algorithms in kinematic design of robotic systems," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 1997, pp. 1111–1117.

[8] C. J. Paredis and P. K. Khosla, "Kinematic design of serial link manipulators from task specifications," *The International Journal of Robotics Research*, vol. 12, no. 3, pp. 274–287, 1993.

[9] J. Denavit and R. Hartenberg, "A kinematic notation of lower-pair mechanisms based on matrices," *ASME Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[10] A. Giusti and M. Althoff, "Automatic centralized controller design for modular and reconfigurable robot manipulators," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3268–3275.

[11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 1, no. 5, pp. 90–98, 1986.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[13] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[14] A. Chakravarthy and D. G., "Obstacle avoidance in a dynamic environment: A collision cone approach," *Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 5, no. 28, pp. 562–574, 1998.

[15] G. Mesesan, E. Icer, and M. Althoff, "Hierarchical genetic path planner for highly redundant manipulators," in *Proc. of the Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.

[16] J. Yu and P. Müller, "An on-line cartesian space obstacle avoidance scheme for robot arms," *Mathematics and Computers in Simulation*, vol. 41, no. 5, pp. 627–637, 1996.

[17] W. Zhang and T. M. Sobh, "Obstacle avoidance for manipulators," *Systems Analysis Modelling Simulation*, vol. 43, no. 1, pp. 67–74, 2003.

[18] R. Menasri, A. Nakib, B. Daachi, H. Oulhadj, and P. Siarry, "A trajectory planning of redundant manipulators based on bilevel optimization," *Applied Mathematics and Computation*, vol. 250, pp. 934–947, 2015.

[19] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space Voronoi bias," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2061–2067.

[20] L. Kelmar and P. K. Khosla, "Automatic generation of kinematics for a reconfigurable modular manipulator system," in *Robotics and Automation, Proceedings., IEEE International Conference on*, 1988, pp. 663–668.

[21] B. Benhabib, G. Zak, and M. Lipton, "A generalized kinematic modeling method for modular robots," *Journal of Robotic Systems*, vol. 6, no. 5, pp. 545–571, 1989.

[22] I. Chen, S. Yeo, G. Chen, and G. Yang, "Kernel for modular robot applications: Automatic modeling techniques," *The International Journal of Robotic Research*, vol. 18, no. 2, pp. 225–242, 1999.

[23] Z. Bi, W. Zhang, I. Chen, and S. Lang, "Automated generation of the D-H parameters for configuration design of modular manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 23, pp. 553–562, 2007.

[24] J. J. Craig, *Introduction to robotics: mechanics and control*. 2nd ed. Reading, MA: Addison-Wesley, 1989.

[25] C. Welman, "Inverse kinematics and geometric constraints for articulated figure manipulation," Ph.D. dissertation, Simon Fraser University, 1993.

[26] S. Chiaverini and B. Siciliano, "The unit quaternion: A useful tool for inverse kinematics of robot manipulators," *Systems Analysis Modelling Simulation*, vol. 35, no. 1, pp. 45–60, 1999.

[27] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational geometry*. Springer, 2000.

[28] C. W. Warren, "A vector based approach to robot path planning," in *IEEE International Conference on Robotics and Automation*, 1991, pp. 1021–1026.