Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Automated Synthesis of Modular Manipulators' Structure and Control for Continuous Tasks around Obstacles

Thais Campos
Cornell University
Email: tcd58@cornell.edu

Samhita Marri
Cornell University
Email: sm2733@cornell.edu

Hadas Kress-Gazit
Cornell University
Email: hadaskg@cornell.edu

***Abstract*—In this work, we describe an end-to-end system for automatically synthesizing correct-by-construction structure and controls for modular manipulators from high-level task specifications. We define specifications that include both continuous trajectories the end-effector must follow and constraints on the physical space (obstacles and possible locations of the base of the manipulator). In our formulation, trajectories are composed of basic shape primitives (lines, arcs, and circles) and we avoid discretizing the desired trajectory, as other approaches in the literature do. We encode the task as a set of constraints on the manipulator's kinematics and return the manipulator's structure and associated control to the user, if a solution is found. By solving for the continuous trajectory, as opposed to a discretized one, we ensure that the original task is satisfied. We demonstrate our approach on three different specifications, and present the resulting physical robots tracing complex trajectories in the presence of obstacles.**

I. Introduction

For a given off-the-shelf robot, many of its capabilities, such as its reachable space and maximum payload, are defined a priori by its structure and actuators. Modular robots [1], on the other hand, allow flexibility in the robot's structure as their components can be rearranged or replaced, thus composing a new machine with new capabilities that can perform a new set of tasks. Modular robots can be specifically designed for a task, thereby guaranteeing its feasibility. However, deciding on a modular robot's structure and controls can be challenging. Such a design process requires knowledge of kinematics, dynamics and controls, and designing these robots manually is a tedious and error prone process.

Serial chain robots are used in domains ranging from maneuvering payloads in space [2] to assisting physicians in surgeries [3]. With such different functions, their structure can vary greatly; however, it follows a general template: a sequence of rigid links connected by joints beginning at a base and ending with an end-effector. The manipulator components, their sizes, capacities and quantities, as well as its location and the surrounding environment will govern its performance. Thus, choosing the correct pieces and their arrangement is crucial for the feasibility of a specific task, but this choice can be difficult to reason about.

We propose an end-to-end system for automatically synthesizing correct-by-construction modular manipulator structure and controls from a high-level task specification. Here, tasks are continuous trajectories in a 2D plane, constructed from parameterized curves. These curves, which we refer to as shape primitives, can be of three types: line segments, arcs and circles. Furthermore, the workspace may include obstacles the robot must avoid and there may be constraints on the position of the manipulator's base. In our approach, we define a set of constraints representing the manipulator's kinematics, which guarantees that, if a design is found, the task is feasible. Users specify a high-level task, such as the "rose" trajectory between obstacles in Fig. 1, and our tool provides instructions for how to construct a robot (link lengths, number of joints, etc.) as well as the control inputs to perform the task.

The automated design of mechanisms based on a task description has been studied in the past [5–25]. In [6–15], the authors use optimization techniques to create a manipulator's structure able to reach a set of points in 3D. For works that considered obstacles in the environment [6–9], a collision-free path that drives the robot through all the task points was computed. Other works [16–19] focus on designing a planar linkage to trace a path that includes selected target points. Other types of high-level tasks were also considered; in [25], the optimization goal is to find a serial chain structure that satisfies industrial requirements, such as level of accuracy.

For common robotic manipulator tasks, such as pick-and-place, if the robot can visit the required points while avoiding obstacles in the environment, then the trajectory performed while executing the task is typically of lesser importance. However, some tasks (e.g. painting, welding, or opening a door) require the end-effector to follow a specified continuous trajectory. If the robot has a fixed structure, motion planners can be used to search for a path such that the end-effector traces the required trajectory while avoiding obstacles. If no path is found, fixed robots cannot accomplish the task. Here, we leverage the configurability of modular robots to jointly synthesize the structure of the robot and the required control thereby enabling the system to achieve a variety of tasks.

Other works consider continuous trajectories as tasks; in [20–22], given an input motion, a time varying rotation and/or translation with respect to a reference point, the authors present techniques for synthesizing mechanisms that display desired output motion. These approaches accept a description of the motion, for example, a rotation, and create a mechanism

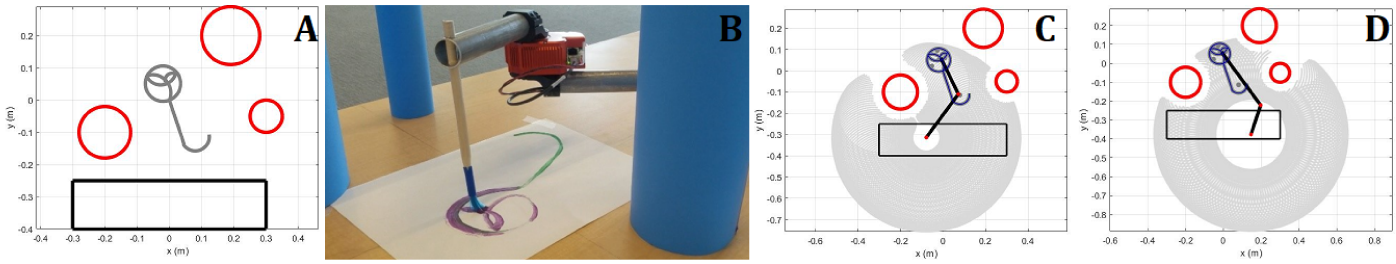

Fig. 1: "Paint a rose". (A) Task specified in the GUI. The red circles are obstacles, the gray line is the desired trajectory and the black rectangle is a constraint on the location of the base (B) The synthesized manipulator executing the task. Reachable space (light gray) of the solution obtained using (C) SV^{app} and (D) SV^{over} as approximations of the swept volume.

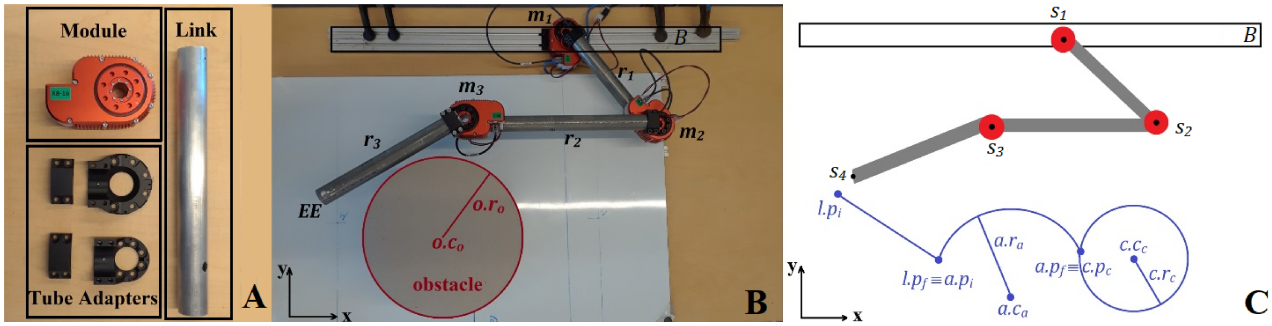

Fig. 2: Manipulator and task definitions. (A) Hebi Robotics X8-16 Module [4], link and tube adapters. (B) Visualization of obstacle and manipulator parameters. (C) Scheme representing a manipulator and a task.

that maps it into another type of motion, such as linear. In these works, a finite set of components were used to construct more complex mechanisms able to perform the required motion. While in [20], the authors used Simulated Annealing to find the best design, in [21] they devised a synthesis methodology based on a matrix representation of the motions, and in [22] they synthesize a structure that implements a given motion specification using constraint programming. In [23, 24], the method takes an input curve and discretizes it to design a device able to reach the points and follow the curve as closely as possible. In [23], the goal was to create mechanical characters capable of performing complex movements. For that, the researchers used a library of parameterized mechanisms to find a structure and its optimized parameters able to reach the discretized points. Similarly, in [24], a library of modular components was used to compose robotic devices to follow a sequence of points selected from a given trajectory. As the set of components in [24] is finite, the authors used A* to perform the search in the design space.

A tradeoff that needs to be considered when discretizing a trajectory is the number of points used; there should be enough to make sure that the curve is still recognizable, but not too many to ensure the problem is still computationally efficient to solve. Furthermore, even if the selected design can reach all the extracted points, there are no guarantees that it can execute the full trajectory, consequently, a discrepancy between the original and the resultant curve might occur (as we demonstrate in Section IV). In our approach, by defining the set of allowable trajectories to be those composed of the shape primitives, we solve for the continuous trajectory, ensuring that the original trajectory is followed.

The main contributions of this paper are: (i) a novel manipulator task formulation that allows a user to specify a desired continuous trajectory in a workspace with obstacles and constraints on the base, (ii) a framework for automatically synthesizing correct-by-construction structure and controls for modular manipulators from high-level tasks, (iii) a continuous formulation of the problem that does not require discretization of the desired trajectory, thereby guaranteeing the solution satisfies the continuous task, and (iv) physical demonstration of our approach on tasks that include complex trajectories (drawing a rose), coverage tasks (wiping off a table) and following a path around obstacles (cleaning a table).

II. Definitions

We use the . operator to refer to the parameters of a variable in our formulation. For example, $l.p_i$ is the first end point of a line segment l. We use $[a, b]$ to denote continuous intervals and $\{a, \ldots, b\}$ for discrete sets. All variables and parameters are depicted in Fig. 2. We represent a line segment between points u, v as $\overline{uv}$, the vector as $\overrightarrow{uv}$, and the vector's length as $||\overrightarrow{uv}||$. The distance between two curves h_1 and h_2 or a curve h_1 and a point p is represented by $||h_1, h_2||$ and $||h_1, p||$, respectively. $||h_1, h_2||$ is the minimum distance between two points on h_1 and h_2, and $||h_1, p||$ between a point on h_1 and p.

A. Task

A shape primitive (SP) is a planar geometric shape characterized by its features. We consider three types of shapes

(Fig. 2(C)): line segment (l), characterized by initial ($l.p_i$) and final ($l.p_f$) points; circle (c), characterized by its center ($c.c_c$), radius ($c.r_c$), and initial and final point ($c.p_c$); and arc (a), characterized by initial ($a.p_i$) and final ($a.p_f$) points, direction ($a.dir$, clockwise or counterclockwise) and its center ($a.c_a$). All points are in a global reference frame. A task includes q shape primitives, such that $q = q_l + q_c + q_a$, where q_l is the number of lines, q_c circles, and q_a arcs.

The workspace of the manipulator may include a set of n_{obs} obstacles, O. An obstacle $o \in O$ is modeled as a circle described by its center's position in the global reference frame ($o.c_o$) and its radius ($o.r_o$). Other obstacle shapes can be over approximated by a circle. We assume that no obstacles are inside the circles or circles defined by the arcs that compose the desired trajectory. Constraints B on the location of the base of the manipulator, if any, are defined as a set of rectangles whose sides are parallel to the global reference frame axes.

A task $T = (SP_q, O, B)$ includes an ordered list of q shape primitives, SP_q, constituting a continuous trajectory to be executed by the manipulator's end-effector, the set of obstacles O the manipulator must not collide with and the constraints B on the location of the base. The shape primitives are connected: the final point of the current shape primitive coincides with the initial point of the next one. We refer to the connection points as *transition points*.

B. Robot structure and control

A module m_k, where $k \in \{1, \ldots, n_{DOF}\}$ and n_{DOF} is the number of modules, is a unit of the manipulator that provides one rotational degree-of-freedom (DOF) (actuator). We define θ_k according to the Denavit-Hartenberg convention [26]. The angle of the first module, θ_1 is defined with respect to the global z-axis.

A link is a rigid body with a fixed length r and diameter Δ that connects two consecutive modules or a module and the manipulator's end-effector (EE).

A point $s_k \in \mathbb{R}^2$ denotes the position of the center of rotation of module m_k in the task plane. We define the state of the manipulator as $S = \{s_1, \ldots, s_{n_{DOF}}, s_{n_{DOF}+1}\}$, where $s_{n_{DOF}+1}$ denotes the end-effector position. For example, in Fig. 2(C), $S = \{s_1, s_2, s_3, s_4\}$. A manipulator state uniquely defines the set of joint angles Θ, where $\Theta = \{\theta_1, \ldots, \theta_{n_{DOF}}\}$. Module m_1 is fixed on the plane at point s_1, the origin (base) of the manipulator. The farthest point of a shape primitive from $s_{n_{DOF}-1}$ is referred to as $p_{farthest}$; and the closest, $p_{closest}$. For $n_{DOF} = 2$, $s_{n_{DOF}-1}$ is the origin of the manipulator. The state of the manipulator that places the EE at a certain point p_m of SP_j is referred to as $S_{j,m}$.

For a 2 DOF planar arm, the shoulder joint is located at the origin of the manipulator (s_1), while the elbow joint, at the end of the first link (s_2). We refer to the angle between the two links as β or the elbow angle, where $\beta \in [0, \pi]$. For points in the reachable space that are not on its boundaries, there are two solutions that place the EE at the point: *upwards* and *downwards* elbow configurations. In the upwards configuration, the joint is located to the left of the line connecting $s_{n_{DOF}-1}$ and the EE (see Fig. 4(A-i)); in the downwards, to the right (see Fig. 4(A-ii)). For points on the boundary, there is only one solution where β is either 0 or π.

A structure D is a list of n_{DOF} modules m_k connected by links with length r_k such that m_1 is fixed on the plane at point s_1. Here, we consider $n_{DOF} \in \{2, 3\}$. The reachable space (RS) is the set of points that the EE can reach.

III. Approach

We formulate the problem of finding both the structure D and controls θ_t (joint angles over time) as an optimization problem where the constraints are the manipulator's kinematics, and the cost is the sum of the link lengths, thus minimizing the size of the manipulator. We are solving for feasibility, i.e. all constraints are met, but the solution may not be globally optimal. We prioritize solutions with less DOF so that the design is less complex. Since the possible number of actuators is small, $n_{DOF} \in \{2, 3\}$, we explicitly enumerate over the options: we first attempt solutions with $n_{DOF} = 2$, and then with $n_{DOF} = 3$. In our optimization formulation, the decision variables are the manipulator's states $S_{j,u}$ of specific points on each shape primitive, and the link lengths r_k. Given $S_{j,u}$, we uniquely calculate θ_t necessary to achieve the desired manipulator state and create the control. The optimization problem we solve is:

$$\min \sum_{k=1}^{n_{DOF}} r_k, \text{ Subject to}$$
$$\forall j \in \{1, ..., q\}, \forall u \in \{i, f, closest, farthest\}, \text{ eqs. (2) to (11)} \quad (1)$$

The solution to the optimization problem is a candidate structure and the configurations of the manipulator for the specified points on the shape primitives. We add verification steps, when necessary, to ensure the existence of collision-free motion as discussed in Sections III-E and III-F.

In the following sections, we define the equations we use to encode the synthesis problem and the methods we use to verify the solution. We first address the main challenges of our problem: how we ensure that the trajectory is collision-free and continuous, which, in our context, means that the end-effector does not deviate from the trajectory. Then, we introduce self-collision avoidance constraints. Finally, we define the constraints due to consistency of the structure.

A. Collision avoidance constraints

We address collision avoidance in two parts: during the execution of shape primitives and during the transition between them. For the first part, we use a method commonly employed in continuous collision detection, which consists of calculating the swept volume (SV) of the manipulator, the space the manipulator moves through as it is executing a motion, and checking for intersections with any obstacles [27–30]. As the SV can be a complex shape, over approximations are used to simplify its calculation. Bounding boxes [29] and convex hulls and their variations [27, 28, 30, 31] are commonly used.

For a 2 DOF planar manipulator, for all points on a shape primitive, there are at most two inverse kinematics (IK)

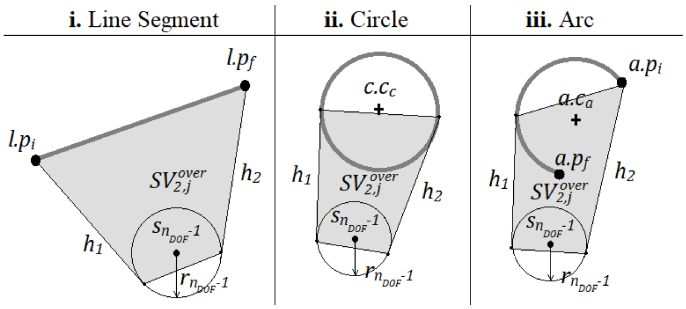

Fig. 3: Over approximation of the swept volume construction for a line segment (i), a circle (ii), and an arc (iii).

solutions that place the EE at the point. However, as more links are added to the manipulator, it becomes redundant, which may result in an uncountable number of solutions for the IK. Although this redundancy is beneficial when the manipulator needs to avoid an obstacle, it increases the complexity of calculating the SV. We simplify the SV calculation for a 3 DOF manipulator by only allowing the last two links (distal links) to move during the execution of a shape primitive, while the remaining one (proximal link) is static. To clarify the nomenclature, a 2 DOF has only distal links.

We reduced the problem of finding a 3-link manipulator able to perform the complete collision-free trajectory into finding a 2-link manipulator with a "changing origin", which is the end of the proximal link. Since only the distal links move during the execution of a SP, the SV calculation can be performed by predicting the boundary positions of each distal link for each SP and bounding the motion by convex shapes. In our system, we first attempt to find a solution by over approximating the SV, SV^{over}. If a solution is found, then it is correct and collision-free. However, since SV^{over} can be conservative, if we cannot find a solution, we proceed to attempt to find a solution by approximating the swept area, SV^{app}, such that it contains most of the SV. In this case, if we find a feasible solution, the system verifies that it is indeed collision-free.

B. Calculation of SV^{over}

The over approximation of the SV for tracing SP_j, SV^{over}_j, is composed of $SV^{over}_{1,j}$ for link $n_{DOF}-1$ and $SV^{over}_{2,j}$ for link n_{DOF}. Link $n_{DOF}-1$, for any SP, will always trace a section of a circle, since it has only 1 DOF. This area is over approximated by a circle $SV^{over}_{1,j}$ with origin at $s_{n_{DOF}-1}$ and radius $r_{n_{DOF}-1}$. The area swept by link n_{DOF} will depend on the arc traced by the elbow and the trajectory traced by the EE, which is SP_j itself. However, we can bound its motion by calculating the tangents h_1 and h_2 from $SV^{over}_{1,j}$ to the SP_j. Then, $SV^{over}_{2,j}$ is the convex hull of the set of points composed of the points where h_1 and h_2 meets SP_j, as well as SP_j end points. Fig 3 shows $SV^{over}_{2,j}$ for each SP.

C. Critical points and reachability

We denote the points p_i, p_f, $p_{closest}$, and $p_{farthest}$ of a shape primitive as the critical points of lines and arcs, and $p_{closest}$, and $p_{farthest}$ as the critical points of a circle. If the end-effector can reach these points, the whole shape primitive is reachable if there are no obstacles. Intuitively, for a line, $p_{farthest}$ is one of the end points of the line segment and $p_{closest}$ is either the intersection point, if it exists, between the line segment and the perpendicular line going through $s_{n_{DOF}-1}$ or one of the end points. For a circle, the critical points are the intersection points between the circle and the line defined by $s_{n_{DOF}-1}$ and $c.c_c$, in which $p_{closest}$ is on the line segment $\overline{s_{n_{DOF}-1}c.c_c}$ and $p_{farthest}$ is outside of it. Similarly, for an arc, the critical points might be the end points, or the intersection points between the arc and the line defined by $s_{n_{DOF}-1}$ and $a.c_c$. We present the equations of $p_{farthest}$ and $p_{closest}$ in Table I.

The RS of a two-link planar manipulator is an annulus, whose larger radius is $r_{n_{DOF}-1}+r_{n_{DOF}}$ and smaller, $|r_{n_{DOF}-1}-r_{n_{DOF}}|$. Thus, for all points $v \in$ RS, Eq. 2 is satisfied, where $s_{n_{DOF}-1}$ is the origin of the 2 DOF distal-links manipulator.

$$|r_{n_{DOF}-1}-r_{n_{DOF}}| \leq ||\overrightarrow{vs_{n_{DOF}-1}}|| \leq r_{n_{DOF}-1}+r_{n_{DOF}} \quad (2)$$

If $p_{farthest}$ and $p_{closest}$ of a shape primitive SP satisfy Eq. 2, then all points on SP are reachable.

D. Calculation of SV^{app}

We define e_j as an approximation of the arc traced by the elbow during the execution of the shape primitive SP_j. This arc is used in the calculation of SV^{app} and is characterized by its end points ($e_j.p_1$ and $e_j.p_2$), direction ($e_j.dir$, clockwise or counterclockwise) and its center ($s_{n_{DOF}-1}$). Given the critical points, $r_{n_{DOF}-1}$, $r_{n_{DOF}}$, and $s_{n_{DOF}-1}$, we apply the IK equations to calculate the elbow position at the critical points, which are then used to define e_j. Fig. 4(A) shows the elbow positions at the critical points for each SP. Both $e_j.p_1$ and $e_j.p_2$ are approximations of the end points of the arc traced by the elbow, used to bound the SV, but the elbow may reach points outside e_j while tracing SP_j.

We calculate SV^{app}_j, the approximation of the volume swept by the manipulator to trace SP_j, by separating it into two convex polygons $SV^{app}_{1,j}$ and $SV^{app}_{2,j}$. $SV^{app}_{1,j}$ is a pentagon over approximation of the section defined by $s_{n_{DOF}-1}$ and arc e_j, as shown in Fig. 4(B).

To construct $SV^{app}_{2,j}$, we first compute the lines g_1 and g_2 such that the over approximation of e_j and SP_j are to the right of g_1 and to the left of g_2. Intuitively, g_1 connects the left-most points on SP_j and the over approximation of e_j; and g_2, the right-most points. For a line segment l, g_1 connects the left-most end point of l and the left-most end point of the over approximation of e_j. Similarly, we construct g_2 by replacing the left-most points with the right-most. Fig. 4(B-i) shows g_1 and g_2 for a line segment l. For a circle c, as it has no end points, g_1 and g_2 are lines that are tangents to both c and the over approximation of e_j (Fig. 4(B-ii)). Similarly, for an arc a, g_1 and g_2 are either the external tangents of SP_j and the over approximation of e_j, if the tangency points lie on a; and/or

SP	Line segment	Circle	Arc
Equation	$\forall l.p \subset l, \forall t \in [0,1]$, $l.p = l.p_i + (l.p_f - l.p_i)t$	$\forall c.p \subset c, \forall t \in [0, 2\pi]$, $c.p = c.c_c + c.r_c[cos(t), sin(t)]$	$\forall a.p \subset a, \forall t \in [t_i, t_f]$ where t_i and t_f are the angles to reach $a.p_i$ and $a.p_f$, respectively, $a.p = a.c_c + a.r_c[cos(t), sin(t)]$
$p_{farthest}$	$l.p_i$ or $l.p_f$	$c.c_c - \frac{c.r_c(s_{n_{DOF}-1} - c.c_c)}{\|\|\overrightarrow{s_{n_{DOF}-1}c.c_c}\|\|}$	$a.c_c - \frac{a.r_c(s_{n_{DOF}-1} - a.c_c)}{\|\|\overrightarrow{s_{n_{DOF}-1}a.c_c}\|\|}$ or $a.p_i$ or $a.p_f$
$p_{closest}$	$l.p_i + (l.p_f - l.p_i)\frac{\overrightarrow{s_{n_{DOF}-1}l.p_i} \cdot \overrightarrow{l.p_f l.p_i}}{\|\|\overrightarrow{l.p_f l.p_i}\|\|}$ or $l.p_i$ or $l.p_f$	$c.c_c + \frac{c.r_c(s_{n_{DOF}-1} - c.c_c)}{\|\|\overrightarrow{s_{n_{DOF}-1}c.c_c}\|\|}$	$a.c_c + \frac{a.r_c(s_{n_{DOF}-1} - a.c_c)}{\|\|\overrightarrow{s_{n_{DOF}-1}a.c_c}\|\|}$ or $a.p_i$ or $a.p_f$

TABLE I: Shape primitive equations, $p_{farthest}$ and $p_{closest}$ for each type.

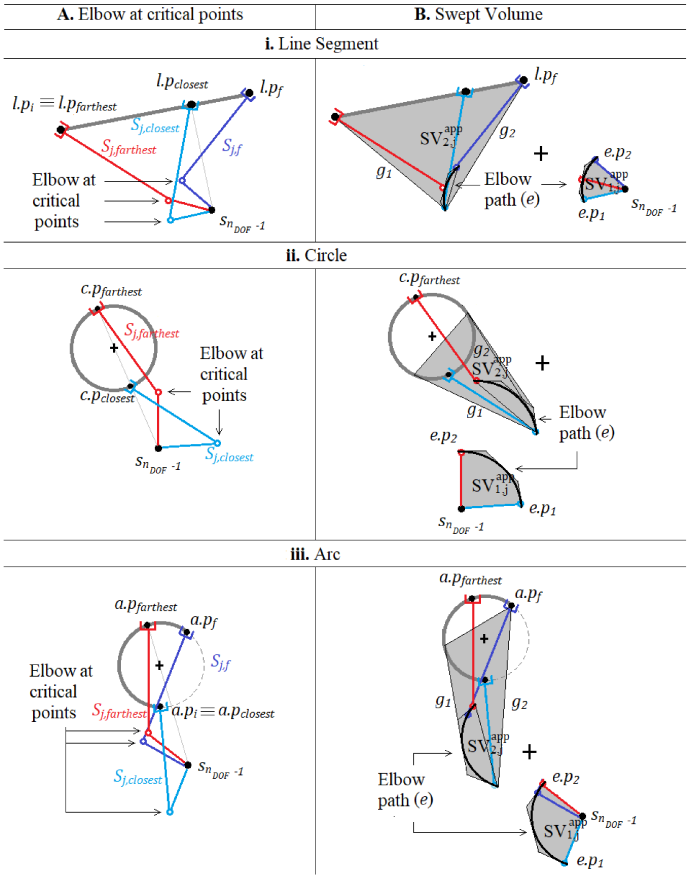

Fig. 4: SV^{app} construction. We do not allow obstacles inside circles or circles defined by arcs. (A) Elbow position at critical points and (B) swept volume for a line segment (i), a circle (ii) and an arc (iii). Critical states are in dark blue ($S_{j,f}$), light blue ($S_{j,closest}$) and red ($S_{j,farthest}$). The SV is decomposed into the SV of the n_{DOF}–1 link, a pentagon over approximation of a sector, and the n_{DOF} link, the convex hull of the elbow and EE boundary positions.

the line connecting the left and/or right-most end point of a and the over approximation of e_j (Fig. 4(B-iii)).

Finally, $SV^{app}_{2,j}$ is the convex hull of the set of points composed of the points where g_1 and g_2 meets SP_j, as well as SP_j end points, and the points that form the quadrilateral that contains e_j, as seen in Fig. 4(B).

E. Collision avoidance constraint encoding

We enforce that $SV_j \cap O = \emptyset$, where SV_j can be either SV^{app}_j or SV^{over}_j. This requirement is encoded in two parts. First, we guarantee that $\forall o \in O$, their centers are outside SV_j. Thus, there cannot be any obstacles fully inside SV_j.

$$\forall j \in \{1, ..., q\}, \forall o \in O, o.c_o \notin SV_j \tag{3}$$

However, even if Eq. 3 holds, part of the obstacles can still be inside SV_j, which occurs when they intersect any edge E of SV_j. Thus, the second part guarantees that the distance between any edge of SV_j and any obstacle center is larger than the obstacle's radius, as encoded in Eq. 4.

$$\forall j \in \{1, ..., q\}, \forall E \in SV_j, \forall o \in O, ||E, o.c_o|| > o.r_o \tag{4}$$

When SV_j is SV^{app}_j, this constraint is not sufficient to guarantee collision-free motion. Thus, after obtaining a candidate solution, we verify that the motion while executing SP_j is in fact collision-free. This verification step is performed as follows. First, we use the IK equations to calculate the joint angles of the candidate structure so the EE traces SP_j. Second, using the joint angles, we calculate the actual SV of the candidate manipulator. Then, we verify that it does not intersect any obstacle. If it does, we restart the optimization with a new initial guess.

F. Continuity constraints

We separate the continuity constraints into two parts: during the execution of SP and during the transition between them.

1) During execution of a SP: The elbow positions at the critical points must be such that the manipulator maintains continuous motion of the EE in tracing the trajectory. As previously discussed, the IK equations for a given point inside the RS of the manipulator can return up to two solutions, in which the elbow is up or down, or, if the point lies on the RS boundary, the elbow is fully extended or fully retracted. Thus, if, for example, the elbow up configuration was returned as a solution for reaching a critical point, then the elbow must remain upwards for the entire motion unless it fully extends while still tracing the path. In our constraint formulation, the elbow is not allowed to change its direction during the execution of the shape primitive. If the elbow position ($s_{n_{DOF}}$) of the distal arm is to the left of the vector connecting $s_{n_{DOF}-1}$ and $s_{n_{DOF}+1}$, then the cross product between the vectors

$\overrightarrow{s_{n_{DOF}-1}s_{n_{DOF}+1}}$ and $\overrightarrow{s_{n_{DOF}-1}s_{n_{DOF}}}$ is positive; if it is to the right, negative. Thus, for any two critical points of a shape primitive, the product between these cross products should be positive so that the elbow maintains its relative position. This constraint is encoded in Eq. 5.

$$\forall j \in \{1,...,q\}, \forall u_1, u_2 \in \{i, f, closest, farthest\}, \overrightarrow{S_{j,u_1}.s_{n_{DOF}}S_{j,u_1}.s_{n_{DOF}-1}} \times \overrightarrow{S_{j,u_1}.s_{n_{DOF}+1}S_{j,u_1}.s_{n_{DOF}-1}}. \overrightarrow{S_{j,u_2}.s_{n_{DOF}}S_{j,u_2}.s_{n_{DOF}-1}} \times \overrightarrow{S_{j,u_2}.s_{n_{DOF}+1}S_{j,u_2}.s_{n_{DOF}-1}} \geq 0 \quad (5)$$

2) During transition between SP: When at a transition point, the manipulator must move from the final state of SP_j (the state for reaching p_f), $S_{j,f}$ into the initial state of SP_{j+1} (the state for reaching p_i), $S_{j+1,i}$. Since the EE remains fixed at the transition point, the manipulator forms a closed kinematic chain with $n_{DOF}+1$ links: the manipulator links in addition to a "ground" link which connects s_1 to the EE. This constraint reduces the total number of DOF of the system. For example, for a 3 DOF manipulator, we need to specify its three joint angles for its state to be fully known. However, for a 4-link closed loop chain, the state is determined by one joint angle. To determine how many DOF a closed loop linkage has, we use the Mobility Formula [32], which determines the number of independent parameters (here, joint angles) that need to be specified for the state to be fully known. The mobility formula for a single loop planar linkage [32] is $F = n - 3$, where F is the number of DOF of the single loop planar linkage and n the number of links. The transition motion must ensure that the loop stays closed. We present constraints for this motion separately for each n_{DOF}.

a) 2 links: Since $n = 3$ (2 links from the manipulator plus the ground link), the mobility F is zero. This means that this linkage has no DOF, thus $S_{j,f}$ and $S_{j+1,i}$ must coincide, as no motion can be created.

b) 3 links: Since $F = 1$, given an input joint angle, the remaining ones can be determined, if they exist. For the transition motion to exist, we calculate the limits of the range of motion for θ_1 and θ_3, the first and third joint angles, respectively, as explained in [32], and replicated in Eqs. 6 and 7. We calculate the angles at the beginning of the transition from SP_j to SP_{j+1}, referred to as $\theta_{1,tr_j,i}$ and $\theta_{3,tr_j,i}$, and at the end of the transition, $\theta_{1,tr_j,f}$ and $\theta_{3,tr_j,f}$. The range of motion depends, exclusively, on the dimensions of the linkage, in which $r_{g,j}$ is the ground link length or the distance between s_1 and the respective transition point.

$$\forall j \in \{1,...,q-1\}, \forall u \in \{i, f\}, \frac{r_{g,j}^2 + r_1^2 - (r_2 - r_3)^2}{2r_1 r_{g,j}} \leq \cos\theta_{1,tr_j,u} \leq \frac{r_{g,j}^2 + r_1^2 - (r_2 + r_3)^2}{2r_1 r_{g,j}} \quad (6)$$

$$\frac{(r_{g,j} + r_3)^2 - (r_2 - r_1)^2}{2r_3 r_{g,j}} \leq \cos\theta_{3,tr_j,u} \leq \frac{(r_{g,j} + r_3)^2 - (r_2 + r_1)^2}{2r_3 r_{g,j}} \quad (7)$$

This constraint is necessary but not sufficient for motion to exist, as during the transition motion, θ_1 may reach a value that yields no solution for θ_3. Thus, when synthesizing a 3 DOF solution to the task, we first obtain a candidate solution and then verify that the transition motions exist and are collision-free. The verification step is performed in two steps. First, we generate a linearly spaced set of θ_1, in which the limits are $\theta_{1,tr_j,i}$ and $\theta_{1,tr_j,f}$. For each θ_1 in the set, we attempt to calculate the resultant joint angles by using the IK equations. If the joint angles exist such that the closeness of the linkage is satisfied, we generate a set of transition states for the manipulator. Second, for each state, we certify that there are no collisions with obstacles by calculating the distance of each link to each obstacle and checking that it is larger than the obstacle's radius plus the link radius. If the verification step fails, we restart the optimization with a new initial guess.

G. Self-collision Constraint

Due to the geometry of the components and the constraints we are imposing on the structure, all links are parallel to each other along the z-axis and are stacked, such that the links closer to the origin are closer to the plane of the task. Therefore, the only possible self-collision is if the EE, which is connected to the end of the last link and extends to the task plane, collides with one of the other links. If the links had no thickness, the EE would collide with the first link of the distal arm only if $\beta = 0$ and $r_{n_{DOF}-1} > r_{n_{DOF}}$. Since $\beta \in [0, \pi]$, if β is equal to zero for a point on the shape primitive, then this point is $p_{closest}$. Thus, the minimum distance between the EE and the link should be larger than zero when the EE reaches $p_{closest}$. However, since the links have thickness, the minimum distance should be larger than the diameter of the links (Δ), if $r_{n_{DOF}-1} > r_{n_{DOF}}$. Eq. 8 guarantees that this minimum distance is satisfied.

$$\forall j \in \{1,...,q\}, u = closest, ||\overline{S_{j,u}.s_{n_{DOF}-1}S_{j,u}.s_{n_{DOF}}}, S_{j,u}.s_{n_{DOF}+1}|| > \Delta \quad (8)$$

Now, we need to consider the possible collision of the EE and the first link for a 3 DOF arm. Since during the execution of the SP, the first link is static, the collision will happen only if the distance between it and the shape primitive itself is smaller than Δ. This constraint is represented in Eq. 9.

$$\forall j \in \{1,...,q\}, ||SP_j, \overline{S_j.s_1 S_j.s_2}|| > \Delta \quad (9)$$

H. Constraints on the structure

We require the structure, i.e. the link lengths and origin, to be identical for all states of the manipulator.

1) Origin: The origin of the manipulator, s_1, must be inside B. If B is not specified, s_1 can be anywhere on the plane.

$$\forall j_1, j_2 \in \{1,...,q\}, \forall u_1, u_2 \in \{i, f, closest, farthest\}, S_{j_1,u_1}.s_1 \equiv S_{j_2,u_2}.s_1, s_1 \in B \quad (10)$$

2) Consistent link lengths: The distance between two consecutive points in S is equal to the corresponding link length.

$$\forall j \in \{1,...,q\}, \forall u \in \{i, f, closest, farthest\}, \forall k \in \{1,...,n_{DOF}\}, ||S_{j,u}.s_k, S_{j,u}.s_{k+1}|| = r_k \quad (11)$$

I. Implementation

We solve the constrained optimization problem for the design and control of a manipulator using Sequential Quadratic Programming (SQP) where we solve for a feasible solution. We use the *fmincon* function in MATLAB R2019a. In this method, an initial guess is required, and the final result can change drastically depending on it. To identify an initial guess that will more likely yield a solution, we first solve a simpler version of the problem and use that as our initial guess.

Initial guess: To initialize the optimization, we choose points along the trajectory and solve for a manipulator that can reach these points while avoiding obstacles, similar to [6, 7]. For computational efficiency, we do not solve for a collision-free trajectory connecting these points, and the number of points selected is small (2 to 5 for each shape primitive). To increase the likelihood of finding a solution to the full optimization, we require for all consecutive EE positions, if an elbow up solution is chosen for a certain point, then it is also selected for the next one (or a solution where the arm is fully extended), and the same for elbow down.

IV. Demonstrations

The results provided by our framework are general enough to be implemented by any kind of module that provides one rotational DOF. In our demonstrations, we used the HEBI Robotics X8-Series Actuators with 110 mm x 73 mm x 45 mm dimensions [4]. They are equipped with sensors that allow position, velocity and torque control. Each module provides one DOF and can rotate continuously. We controlled each separately, via Ethernet, using a 64 bit desktop computer running Ubuntu 14.04 with 8 GB RAM and 3.6 GHz processor.

We used aluminum tubes with 31.75 mm diameter as links and cut them to the required length based on the synthesis results. To connect a link and an actuator, we used tube adapters manufactured by HEBI Robotics to provide zero degrees link twist and result in a planar manipulator. Fig. 2(A) shows each component. We drilled the tube connected to the end-effector to accommodate a 9.5 mm diameter rod that ends with the EE, responsible for interacting with the task plane. The type of the end-effector used depended on the task: sponges for wiping the table and a brush to paint a rose.

Our approach outputs a design and the associated joint angles required for the task. We used an open-loop proportional controller based on the actuator position. The actuators used have position sensors, so no external sensors are needed.

A. Physical Demonstrations

In this section we demonstrate our approach to manipulator synthesis from high-level specifications. We implemented a Graphical User Interface (GUI) to facilitate the specification of high-level tasks. We show the task as defined by the user in our GUI, the resulting synthesized solution and the physical implementation of the task using Hebi robotics modules and a variety of end effectors.

Table 2 presents the task name, the shape primitive types used to create the task, the resultant manipulator and the total computational time to find a solution. Fig. 1 and Fig. 5 show the task definitions in the GUI, the solutions obtained and the physical implementations. All the solutions we fabricated were obtained using the SV^{app} approximation of the SV. Using SV^{over}, we only found a solution for the rose task (Fig. 1 (D)). For the tasks in Fig. 5, the manipulator is required to go around obstacles and therefore it is not surprising that an over approximation of the swept area is too restrictive.

A video depicting the simulations and the physical demonstrations is available at https://youtu.be/9Uvyu2FJtVM.

B. Comparison to discretized solutions

In previous works (e.g. [23, 24]) the desired trajectory is discretized by choosing a set of points that are then used to find a robot design. Similarly, in [6, 9], the manipulator's structure is synthesized based on a task composed of a set of points and a trajectory connecting them is computed using sampling-based techniques. To illustrate the benefit of solving for a continuous trajectory, as opposed to a discretized one, we revisit our "paint a rose" task, and compare the synthesized solutions and their resulting trajectories.

We discretize the required curve using 25 equidistant points and calculate a solution in MATLAB using a modification of the formulation presented in [6]. Specifically, since the manipulator is planar, we enforce α to be zero. Furthermore, it is important to find the configurations to reach each selected point that are more likely to yield a continuous and smooth trajectory as similar as possible to the original. For that, we added the elbow consistency constraint, used also for the initial guess for the continuous solution, and we implemented a cost function to minimize the distance between the states of two consecutive poses. We used RRT [33] with bias towards the next point as the motion planner.

The computation time for our approach is 0.5min, using SV^{over}, and for the discretized approach, it is 6.8min. Both approaches result in a 2 DOF solution. Fig. 6(A) presents the RS of the synthesized manipulator using the discretized approach. Although all discretized points are reachable, the required curve is not. Fig. 6(B) shows a region of the task curve located outside the RS (black curve identifying the boundary). Fig. 6(C) shows the collision-free trajectory performed by both synthesized manipulators. Although the path obtained with the sampling-based motion planner is close to the required curve, they are not identical. Thus, an additional step needs to be included to recalculate the path after the selection of the structure, so it is closer to the original, or more points need to be added. Such steps are not necessary in our approach.

V. Conclusions

Summary. In this work, we defined a novel, high-level, manipulator task formulation, and created a framework to automatically synthesize a manipulator's structure and controls from the task, presenting an end-to-end system from user task definition to hardware implementation. We demonstrated the versatility of our approach by synthesizing and implementing manipulators for three different tasks and we compared our

Task name	Types and number of SP of the task	Number of Obstacles	Resulting manipulator	Computational Time (min)	Figure reference
Paint a "rose"	7 - All types	3	2 DOF	2.7 (SV^{app}) and 0.5 (SV^{over})	Fig. 1
Wipe off a table	3 - Line segments	2	3 DOF	9.6	Fig. 5(A)
Clean a table	5 - Line segments and arcs	2	3 DOF	7.5	Fig. 5(B)

TABLE II: Tasks and results. The types of SP used to construct the task are specified as well as the number of obstacles, the resulting manipulator, and the computational time to obtain it.

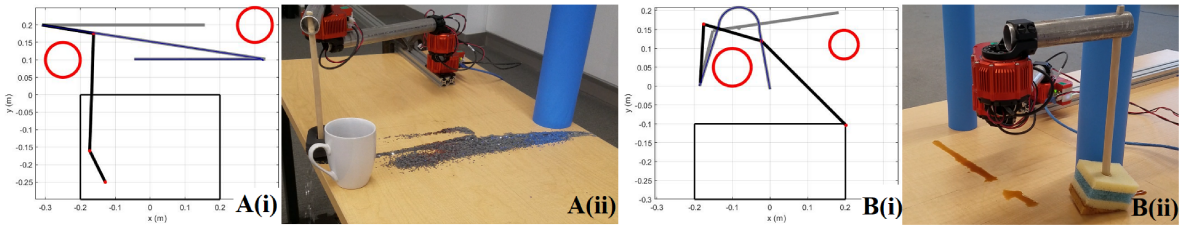

Fig. 5: Task definition, solution and physical implementation of two different tasks. (A) Wipe off a table; (B) Clean a table. (i) Task definition (grey trajectory and red obstacles) and solution (black) and (ii) physical implementation.

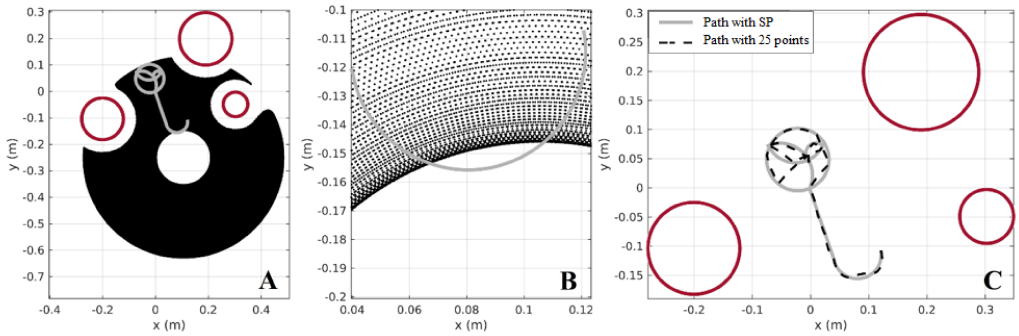

Fig. 6: Reachable space and path of solution obtained by discretizing the original curve. (A) Obstacles in red, trajectory in gray and obstacle-free reachable space in black of the design for a task containing 25 target points obtained by discretizing the original curve. (B) Zoom in showing the curve is outside the reachable space of the solution obtained with discretization. (C) The task performed with the solution synthesized based on the approach in this paper (solid grey) and with the approach of [6] (dashed black) with RRT [33].

approach with previous method that discretize the trajectory and showed the advantage of specifying and synthesizing for continuous tasks. The tasks demonstrated include complex trajectories in constrained environments, coverage of area while avoiding obstacles and circumvention of obstacles. By creating a high-level task specification formalism and automatically synthesizing a correct-by-construction solution, we allow non-expert users to develop and manufacture their own robots based on their needs. The solutions returned by our approach are designed specifically for the task and are therefore less complex than off the shelf manipulators that are typically 6 or 7 DOF. Moreover, by implementing the results with modular robots, it is easy to replace if a piece is broken and to reassemble if the task changes. **Future Work.** There are several directions for future work: First, in this work, we consider trajectories that lie on a 2D plane. We will extend this approach to include trajectories in 3D, possibly with more DOF, by modifying the SV representation from convex polygons to convex polyhedrons and adding new constraints in the robot's configurations (similar to the static proximal link constraint). Second, while our approach produces correct solutions, manipulators that can achieve the desired task, it is not complete. There might exist a solution that is not found. We will develop both automated ways to increase the likelihood of finding a solution, for example by iteratively increasing the number of iterations of the optimizer and trying out different initial guesses, and ways to provide feedback to the users regarding infeasible tasks, similar to [6]. Such feedback could be in the form of suggested changes to the task, relaxation of constraints, or parts of the task that are particularly difficult to achieve. Third, we will explore physical limitations on the resulting behavior. For example, in the tasks presented in this paper, the manipulator was required to press the end-effector on the table, generating friction, which might deform the original trajectory.

ACKNOWLEDGMENTS

This work was funded by NSF CNS-1837506.

References

[1] Mark Yim, Ying Zhang, and David Duff. Modular robots. *IEEE Spectrum*, 39(2):30–34, 2002.

[2] Dan King. Space servicing: past, present and future. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS*, pages 18–22, 2001.

[3] Simon DiMaio, Mike Hanuschik, and Usha Kreaden. The da vinci surgical system. In *Surgical Robotics*, pages 199–217. Springer, 2011.

[4] HebiRobotics. URL https://www.hebirobotics.com/x-series-smart-actuators.

[5] Chris Leger and John Bares. Automated task-based synthesis and optimization of field robots. 1999.

[6] Thais Campos, Jeevana Priya Inala, Armando Solar-Lezama, and Hadas Kress-Gazit. Task-based design of ad-hoc modular manipulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6058–6064. IEEE, 2019.

[7] Julian Whitman and Howie Choset. Task-specific manipulator design and trajectory synthesis. *IEEE Robotics and Automation Letters*, 4(2):301–308, 2018.

[8] EJ Van Henten, DA Van't Slot, CWJ Hol, and LG Van Willigenburg. Optimal manipulator design for a cucumber harvesting robot. *Computers and electronics in agriculture*, 65(2):247–257, 2009.

[9] Cenk Baykal and Ron Alterovitz. Asymptotically optimal design of piecewise cylindrical robots using motion planning. In *Robotics: Science and Systems*, 2017.

[10] Sarosh Patel and Tarek Sobh. Task based synthesis of serial manipulators. *Journal of advanced research*, 6(3): 479–492, 2015.

[11] Wan Kyun Chung, Jeongheon Han, Youngil Youm, and SH Kim. Task based design of modular robot manipulator using efficient genetic algorithm. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 507–512. IEEE, 1997.

[12] J-O Kim and Pradeep K Khosla. A formulation for task based design of robot manipulators. In *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, volume 3, pages 2310–2317. IEEE, 1993.

[13] I-Ming Chen and Joel W Burdick. Determining task optimal modular robot assembly configurations. In *proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 132–137. IEEE, 1995.

[14] Saleh Tabandeh, William Melek, Mohammad Biglarbegian, Seong-hoon Peter Won, and Chris Clark. A memetic algorithm approach for solving the task-based configuration optimization problem in serial modular and reconfigurable robots. *Robotica*, 34(9):1979–2008, 2016.

[15] M Althoff, A Giusti, SB Liu, and A Pereira. Effortless creation of safe robots from modules through self-programming and self-verification. *Science Robotics*, 4 (31):eaaw1924, 2019.

[16] JA Cabrera, A Ortiz, F Nadal, and JJ Castillo. An evolutionary algorithm for path synthesis of mechanisms. *Mechanism and Machine Theory*, 46(2):127–141, 2011.

[17] JA Cabrera, A Simon, and M Prado. Optimal synthesis of mechanisms with genetic algorithms. *Mechanism and machine theory*, 37(10):1165–1177, 2002.

[18] Suwin Sleesongsom and Sujin Bureerat. Four-bar linkage path generation through self-adaptive population size teaching-learning based optimization. *Knowledge-Based Systems*, 135:180–191, 2017.

[19] M Khorshidi, M Soheilypour, M Peyro, A Atai, and M Shariat Panahi. Optimal design of four-bar mechanisms using a hybrid multi-objective ga with adaptive local search. *Mechanism and Machine Theory*, 46(10): 1453–1465, 2011.

[20] Shean-Juinn Chiou and Kota Sridhar. Automated conceptual design of mechanisms. *Mechanism and machine theory*, 34(3):467–495, 1999.

[21] Lifeng Zhu, Weiwei Xu, John Snyder, Yang Liu, Guoping Wang, and Baining Guo. Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127–1, 2012.

[22] Devika Subramanian et al. Kinematic synthesis with configuration spaces. *Research in Engineering Design*, 7 (3):193–213, 1995.

[23] Stelian Coros, Bernhard Thomaszewski, Gioacchino Noris, Shinjiro Sueda, Moira Forberg, Robert W Sumner, Wojciech Matusik, and Bernd Bickel. Computational design of mechanical characters. *ACM Transactions on Graphics (TOG)*, 32(4):83, 2013.

[24] Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. Computational design of robotic devices from high-level motion specifications. *IEEE Transactions on Robotics*, 34(5):1240–1251, 2018.

[25] Anna Valente. Reconfigurable industrial robots: A stochastic programming approach for designing and assembling robotic arms. *Robotics and Computer-Integrated Manufacturing*, 41:115–126, 2016.

[26] Peter I Corke. A simple and systematic approach to assigning denavit–hartenberg parameters. *IEEE transactions on robotics*, 23(3):590–594, 2007.

[27] Holger Täubig, Berthold Bäuml, and Udo Frese. Real-time continuous collision detection for mobile manipulators-a general approach. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 461–468. IEEE, 2012.

[28] Andre Gaschler, Ronald Petrick, Torsten Kröger, Oussama Khatib, and Alois Knoll. Robot task and motion planning with sets of convex polyhedra. In *Robotics: Science and Systems (RSS) Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.

[29] Stephane Redon, Ming C Lin, Dinesh Manocha, and Young J Kim. Fast continuous collision detection for articulated models. 2005.

[30] Jing Xia, Zainan Jiang, Hong Liu, Hegao Cai, and Guangxin Wu. A manipulator's safety control strategy based on fast continuous collision detection. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2380–2385. IEEE, 2013.

[31] Holger Täubig, Berthold Bäuml, and Udo Frese. Real-time swept volume and distance computation for self collision detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1585–1592. IEEE, 2011.

[32] J Michael McCarthy and Gim Song Soh. *Geometric design of linkages*, volume 11. Springer Science & Business Media, 2010.

[33] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.