International Journal of Robotic Computing
Vol. 3, No. 1 (2021) 121-145
© KS Press, Institute for Semantic Computing Foundation
DOI: 10.35708/RC1870-126268

A Quadratic Programming Approach to Manipulation in Real-Time Using Modular Robots

Chao Liu and Mark Yim

University of Pennsylvania at 3401 Grays Ferry Avenue, Philadelphia, US
{chaoliu,yim}@seas.upenn.edu

Received (03/17/2021)
Accepted (04/14/2021)

Abstract. Motion planning in high-dimensional space is a challenging task. In order to perform dexterous manipulation in an unstructured environment, a robot with many degrees of freedom is usually necessary, which also complicates its motion planning problem. Real-time control brings about more difficulties in which robots have to maintain the stability while moving towards the target. Redundant systems are common in modular robots that consist of multiple modules and are able to transform into different configurations with respect to different needs. Different from robots with fixed geometry or configurations, the kinematics model of a modular robotic system can alter as the robot reconfigures itself, and developing a generic control and motion planning approach for such systems is difficult, especially when multiple motion goals are coupled. A new manipulation planning framework is developed in this paper. The problem is formulated as a sequential linearly constrained quadratic program (QP) that can be solved efficiently. Some constraints can be incorporated into this QP, including a novel way to approximate environment obstacles. This solution can be used directly for real-time applications or as an off-line planning tool, and it is validated and demonstrated on the CKBot and the SMORES-EP modular robot platforms.

Keywords: Manipulation; Quadratic Programming; Modular Robots.

1 Introduction

Manipulation tasks are common in robotics applications. In unstructured, cluttered environments, these tasks are usually executed by redundant robots to reach larger workspaces while avoiding obstacles and other constraints. This results in motion planning in high-dimensional space.

The motion planning problem is usually solved by some well developed framework (e.g. MoveIt! [6]) containing three components: a *path planner*, a *trajectory* *generator*, and a *tracking controller*. The path planner is responsible for generating collision-free paths. The trajectory generator smooths the computed paths and generates trajectories that can be parameterized by time while satisfying motion constraints, such as maximum velocities and accelerations. The tracking controller guarantees the motion of the robot when executing the generated trajectories. This type of framework has shown successful applications in many scenarios but rarely achieves real-time performance for all three components in high dimensions. Some approaches combined path planning with trajectory optimization that can directly construct trajectories resulting from optimization over a variety of criteria. These approaches are related to optimal control of robotic systems.

Modular self-reconfigurable robotic systems are usually composed of a small set of building blocks with uniform docking interfaces that allow the transfer of mechanical forces and moments, electrical power, and communication throughout the robot [38]. These platforms are designed to be versatile and adaptable with respect to different tasks, environments, functions or activities. A single module in a modular robotic system usually has one or more degrees of freedom (DoFs). Combining many modules to form versatile systems results in robots requiring representations with high dimensions. This dimensionality makes control and motion planning difficult. That the system is not a single structure but can take a very large number of configurations (typically exponential in the number of modules) requires an approach that can be applied to arbitrary configurations. For example, a modular robot configuration built with PolyBot [37] modules is shown in Fig. 1 which has multiple serial kinematic chains. This is different from common multi-limb systems with a single base. These systems can be modeled such that chains are decoupled.

In a modular robotic system, modules usually are approximated by simple shapes such as a cube or a sphere. This is often useful for reconfiguration but can make manipulation more complex. Rather than two long fingers in a parallel jaw gripper, to obtain similar geometry, a modular robot may require many modules to form those long fingers. This results in grasping type of motions in which two or more chains can behave as a multi-arm system to clamp an object [33].

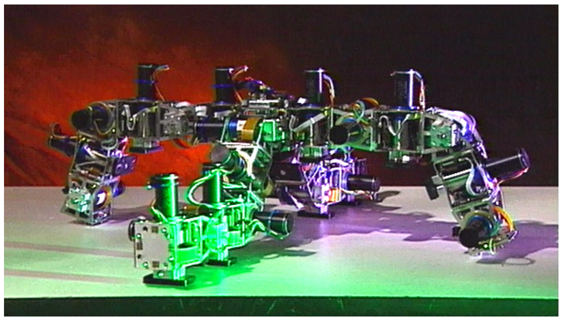


Fig. 1: A modular robot configuration built by PolyBot modules is composed of multiple chains [37].

Hence, in order to grasp some object, motion planning for multiple (potentially high DoF) chains are necessary. In addition their motions are strongly coupled. This fact leads to a more complicated control and planning problem.

In this paper, we present a new approach for real-time manipulation planning and control as well as a novel way to approximate environment obstacles, and apply it to two modular robotic systems. In order to solve the problem in general, an universal kinematics model is required for arbitrary configurations. This approach concomitantly can be easily extended, such as a dual-arm system or a modular robot configuration that has multiple chains. We propose a quadratic programming approach that can be solved efficiently for real-time applications. This requires that system control stability, hardware motion constraints (joint limits and actuation limits), and collision avoidance can be incorporated into this quadratic program (QP) as linear constraints. One advantage of this approach is that the large variety of configurations and kinematic structures found from modular robotic systems can be represented easily as linear constraints, including situations where multiple portions of the robot may have different simultaneous goals. Our approach can also be used as an off-line trajectory planner by simple Euler integration. The framework is tested and evaluated on CKBot [39] and SMORES-EP [20] in the end.

The paper is organized as follows. Sec. 2 reviews relevant and previous works. Sec. 3 introduces the details to derive the kinematics model required to describe the motion of any modular robotic configuration. Sec. 4 discusses the approach to control and motion planning for given tasks. Some experiments are validated in Sec. 5 with some analysis. Sec. 6 includes the conclusions and future work.

2 Related Work

High-dimensional motion planning and control have been studied over several decades. In this section, we review several types of approaches from previous work and some special approaches for modular robots.

2.1 Motion Planning for Manipulation

Artificial potential field manipulation planning methods can avoid searching in high-dimensional configuration space, planning in operational space directly [16]. Robots can avoid collision in real time, but may get stuck at local minima. Analytical navigation functions that have a unique minimum at the goal configuration avoiding local minima are shown in [30]. However, it is usually computationally expensive to build such a navigation function in general. A Monte Carlo technique was applied to escape local minima of the potential by executing Brownian motions [3].

Sampling-based approaches have been used widely for high-dimensional motion planning problems. The probabilistic roadmap (PRM) has been demonstrated on planar articulated robots with many DoFs [2,15]. Expansive configuration space was proposed to resolve the narrow passage issue which is a common problem for sampling-based planners [12]. Rapidly-exploring Random Trees (RRT) approach was later presented in [17] to deal with nonholonomic constraints. An optimal sampling-based planner (RRT*) is introduced in [14] with less efficiency. These approaches require post-processing to generate smooth trajectories in order to be executable for real tasks.

Search-based planners rely on the discretization of the space. However these approaches are generally not suitable for high-dimensional problems. For example, naïve A* can rarely scale to large complicated planning problems. In order to increase the efficiency of these approaches, a number of suboptimal heuristic searches have been proposed [8,18,19]. These methods are promising but are currently computationally inefficient when solving motion planning problems in high-dimensional space.

Once a feasible path is found, a trajectory generator is needed to smooth and shorten the computed path with time parameterization. Trajectories are modeled as elastic bands that need to maintain equilibrium states under internal contraction forces and external repulsion forces [28,5]. Obstacles in the workspace are considered directly which is also beneficial for real-time trajectory modification while a precomputed path is necessary.

Another class of planners are related to optimal control. Rather than separate the planning process into two phases (path planning and trajectory planning), trajectories are constructed directly by these frameworks which optimize over a variety of criteria. A global time-optimal trajectory generator is introduced in [35]. It combines a grid search with a local optimization to obtain the global optimal solution. This approach requires the representations of obstacle regions in configuration space which is difficult to derive for high-dimensional problems. CHOMP [29,42] formulates the cost to be the combination of trajectory smoothness and obstacle avoidance, and gradients for these two terms are needed. This approach uses pre-computed signed distance fields for collision checking. A similar idea is used in ITOMP that can also deal with dynamic environments [27]. In contrast, STOMP [13] can also handle more general objective functions for which gradients are not available by using trajectory samples, but can be difficult to determine the number of samples. A sequential convex optimization approach is presented in [32] which adds new constraints and costs during the motion so as to tackle a larger range of motion. Collision is detected by checking the intersection of the swept-out volume of the robot in an interval and obstacles, and a collision-avoidance penalty gradient can be incorporated into the optimization problem to ensure safety. These works mainly focus on single-high-DoF-arm manipulation tasks. Given the trajectories of end-effectors, an optimal control framework is formulated to solve whole-body manipulation tasks [34]. A repulsive velocity can be applied to any rigid body whenever it collides with any obstacle based on a physical simulator. Reachable sets are used for safe and real-time trajectory design in [11], but the reachability analysis has to be offline. Some optimal controllers handle the obstacles by mixed integer programming which is known to be an NP-hard problem [31].

Our approach is also related to optimal control and differs from these previous works in two ways: (a) the way in which the motion planning problem is formulated and (b) the simple model that approximates the environment obstacles. We incorporate multiple motion goals into the objective function in the form of feedback controllers to guarantee the trajectory tracking performance or efficient search for navigation, and the output can be applied on the system directly to achieve real-time performance. During the motion, both the objective function and constraints may be updated according to the current scenario which allows our approach to tackle a wider range of tasks. For collision avoidance, we present a new way to simplify obstacles dynamically during the motion, and the collision avoidance constraint can be modeled as linear constraints in order to solve the optimization problem efficiently. A collision-avoidance penalty is added to the objective function when any rigid body is near any obstacle in the form of the projected motion from the rigid body to this obstacle. Our approach is well suited for real-time applications since its output can be applied on robotic systems directly in real-time, or can be used as an off-line trajectory planner by integrating the output over time.

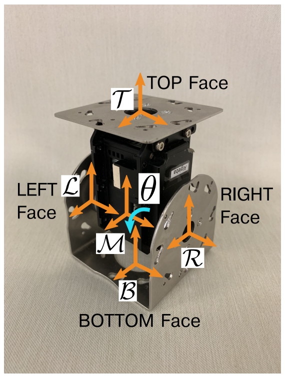
2.2 Modular Robot Control and Planning

Modular robots are inherently systems with many DoFs. They are usually composed of a large number of modules with each module has one or more DoFs. This paper addresses the manipulation tasks of modular robots that form configurations in tree topologies. That is they are constructed from multiple serial chain configurations without forming loops.Work related to manipulation of modular robot systems includes inverse kinematics for highly redundant chains using PolyBot [37,1], and constrained optimization techniques with nonlinear constraints [7]. Due to complicated constraints in these approaches, real-time applications for large systems cannot be guaranteed and numerical issues have to be addressed when solving the optimization problem in the presence of obstacles. Another set of related work includes controller design for modular robots, such as an adaptive control approach using a neural network architecture [25], a virtual decomposition control approach [41], a distributed control method with torque sensing [24], and a centralized controller [9]. These approaches consider the control problem in a free environment and require extra motion planning to fully control the system in a complex environment.

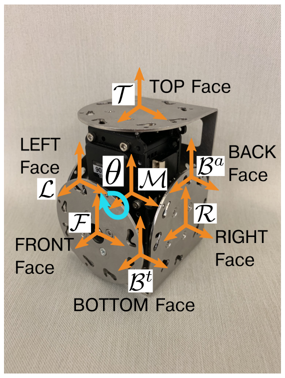
This paper is built upon the work from our conference paper [21]. In the conference paper, we introduced a quadratic programming approach to address real-time control and planning, as well as a general solution to build kinematics models of modular robotic systems. In this work, we present a sequential optimization approach in which the objective function and constraints are updated according to the current situation to ensure safety. In order to handle more complex environments, a novel model is introduced to approximate obstacles by their significance to the current robot motion.

3 Kinematics For Modular Robots

In this section, we derive a general kinematics model for modular robots. For other manipulators, a similar technique can be applied to derive necessary models in order to utilize our planning framework.



(a)



(b)

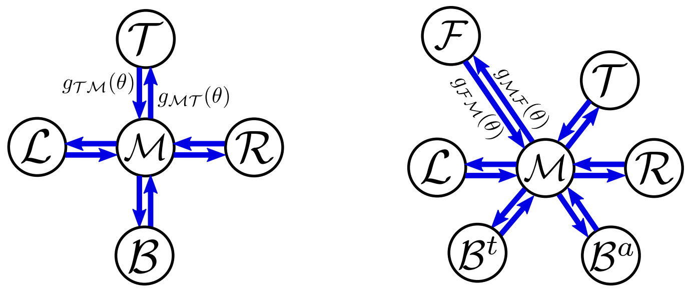
Fig. 2: (a) A CKBot UBar module has one DoF and four connectors. (b) A CKBot CR module has one DoF and six connectors.

3.1 Kinematics Graph

The representation of a modular robot configuration is discussed in [23] which is an undirected graph $G = (V, E)$. Each vertex $v \in V$ represents a module and each edge $e \in E$ represents the connection between two modules.

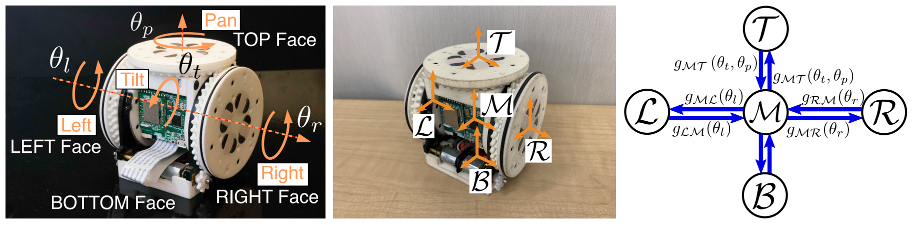
We use a *module graph* to model a module's topology which includes all connectors and joints. A **module graph** is a directed graph $G_m = (V_m, E_m)$: each vertex is a rigid body in the module which is either a connector or the module body, and each edge represents how two adjacent rigid bodies are connected. The transformations among all rigid bodies are determined by the joint set and the geometry. For example, a CKBot UBar module in Fig. 2a is a single-DoF module as well as four connectors (TOP Face or $\mathcal{T}$, BOTTOM Face or $\mathcal{B}$, LEFT Face or $\mathcal{L}$, and RIGHT Face or $\mathcal{R}$). For simplicity, when the module joints are in its zero position, all rigid bodies are in the same orientation and the translation offsets among them are determined by the module geometry. Let $\mathcal{B}$ be fixed in $\mathcal{M}$, then the homogeneous transformations among $\mathcal{M}$, $\mathcal{L}$, and $\mathcal{R}$ are invariant of the joint parameter θ because they are rigidly connected. Only the homogeneous transformation between $\mathcal{M}$ and $\mathcal{T}$ is not invariant to θ. This relationship can be fully represented in a directed graph shown in Fig. 3a. The edge direction denotes the direction of the corresponding forward kinematics. $\mathbf{G}_m$ is the set of unique module graphs G_m for a modular robotic system since some systems have more than one type of modules (e.g., CKBot in Fig. 2).

In general, given a module m with connector set C and joint set Θ, a frame $\mathcal{C}$ is attached to each connector $c \in C$ and frame $\mathcal{M}$ is attached to the module body. Let mapping $g_{\mathcal{F}_1\mathcal{F}_2} : Q \to SE(3)$ describe the forward kinematics from $\mathcal{F}_1$ to $\mathcal{F}_2$ in joint space Q, then $\forall c \in C$, $g_{\mathcal{M}\mathcal{C}}$ and $g_{\mathcal{C}\mathcal{M}}$ can be defined with respect to Θ. The results for CKBot CR modules and SMORES-EP modules are shown in Fig. 3b and Fig. 4. With a module graph model, we can easily obtain the **kinematics graph** $G_K = (V_K, E_K)$ for a modular robot configuration which is constructed by composing the modules by connecting connectors. A directed



(a) (b)

Fig. 3: (a) The module graph of a CKBot UBar module in which $g_{\mathcal{MB}}$, $g_{\mathcal{BM}}$, $g_{\mathcal{ML}}$, $g_{\mathcal{LM}}$, $g_{\mathcal{MR}}$, and $g_{\mathcal{RM}}$ are invariant of θ. (b) The module graph of a CKBot CR module in which $g_{\mathcal{MB}^a}$, $g_{\mathcal{B}^a\mathcal{M}}$, $g_{\mathcal{MB}^t}$, $g_{\mathcal{B}^t\mathcal{M}}$, $g_{\mathcal{MT}}$, $g_{\mathcal{TM}}$, $g_{\mathcal{ML}}$, $g_{\mathcal{LM}}$, $g_{\mathcal{MR}}$, and $g_{\mathcal{RM}}$ are invariant of θ.



θ_p
Pan
TOP Face
θ_l
θ_t
Tilt
Left
LEFT Face
θ_r
Right
BOTTOM Face
RIGHT Face
$\mathcal{T}$
$\mathcal{M}$
$\mathcal{L}$
$\mathcal{R}$
$\mathcal{B}$
$g_{\mathcal{MT}}(\theta_t, \theta_p)$
$g_{\mathcal{MT}}(\theta_t, \theta_p)$
$g_{\mathcal{ML}}(\theta_l)$
$g_{\mathcal{RM}}(\theta_r)$
$g_{\mathcal{LM}}(\theta_l)$
$g_{\mathcal{MR}}(\theta_r)$

(a) (b) (c)

Fig. 4: (a) A SMORES-EP module has four DoFs and four connectors. (b) The frames of all rigid bodies are shown and $\mathcal{B}$ is fixed in $\mathcal{M}$. (c) The module graph of a SMORES-EP module in which $g_{\mathcal{MB}}$ and $g_{\mathcal{BM}}$ are invariant of $\Theta = (\theta_l, \theta_r, \theta_p, \theta_t)$.

edge is used to denote each connection and the transformation between the two mating connectors is fixed since they are rigidly connected. Using this kinematics graph, a *kinematic chain* from frame $\mathcal{F}_1$ to frame $\mathcal{F}_2$ can be derived by following the shortest path $G_K : \mathcal{F}_1 \rightsquigarrow \mathcal{F}_2$. This creates a graph with no loops. A simple configuration built by two CKBot UBar modules is shown in Fig. 5a. Frame $\mathcal{W}$ is the world frame and module m_1 is fixed to it via its BOTTOM Face. The kinematics graph for this configuration is shown in Fig. 5b and the kinematic chain from $\mathcal{W}$ to $\mathcal{T}_2$ shown in Fig. 5c. All the edges have fixed homogeneous transformations except for edge $(\mathcal{M}_1, \mathcal{T}_1)$ and edge $(\mathcal{M}_2, \mathcal{T}_2)$, and we can conclude that the forward kinematics mapping is $g_{\mathcal{W}\mathcal{T}_2} : \mathbb{T}^2 \to SE(3)$ where $\mathbb{T}^p$ represents the p-torus. However, we can also see that all the edges in the shortest path from $\mathcal{W}$ to $\mathcal{L}_1$ have fixed homogeneous transformations, so $\mathcal{L}_1$ is fixed in $\mathcal{W}$.

Similar to the configuration discovery algorithm in [23], the kinematics graph can be built by visiting modules in breadth-first-search order. The given configuration is traversed from the module fixed to the world frame $\mathcal{W}$. When visiting a new module m, denoting its parent via its connector c as $\tilde{m}$ and the mating

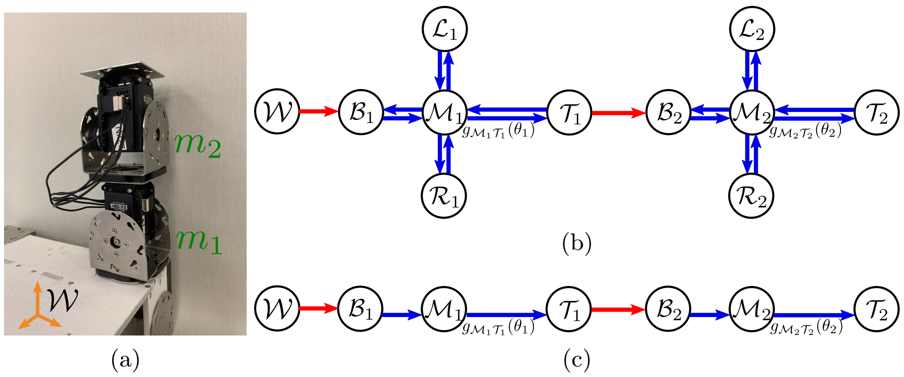


Fig. 5: (a) A configuration by two CKBot UBar modules. (b) The kinematics graph model of the configurations. (c) The kinematic chain from $\mathcal{W}$ to $\mathcal{T}_2$.

connector of $\tilde{m}$ as $\tilde{c}$, record the fixed homogeneous transformation $g_{\mathcal{C}\tilde{\mathcal{C}}}$ in which frame $\mathcal{C}$ and frame $\widetilde{\mathcal{C}}$ are attached to c and $\tilde{c}$ respectively. Not until all modules are visited, is the $G_K = (V_K, E_K)$ of the given configuration constructed. With this structure, there is no need for case-by-case derivation of the kinematics as long as the kinematics for each type of module and connection are defined.

3.2 Kinematics for Modules

Recall that given a module m with connector set C and joint set Θ, a frame $\mathcal{C}$ is attached to each connector $c \in C$ and frame $\mathcal{M}$ is attached to the module body. For a joint $\theta \in \Theta$, a twist $\hat{\xi}_\theta \in se(3)$ can be defined with respect to $\mathcal{M}$ in which $\xi_\theta = (v_\theta, \omega_\theta) \in \mathbb{R}^6$ is the twist coordinates for $\hat{\xi}_\theta$[1], and $\boldsymbol{\xi}$ is the set of the twist associated with each joint. For homogeneous transformation $g_{\mathcal{MC}}$, it is straightforward to have

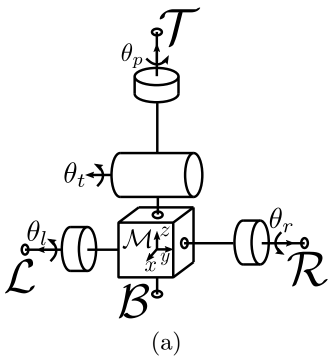
$$g_{\mathcal{MC}} = g_{\mathcal{MC}}(\Theta^{\mathcal{C}}) = \prod_i \exp(\hat{\xi}_{\Theta_i^{\mathcal{C}}} \Theta_i^{\mathcal{C}})\; g_{\mathcal{MC}}(0) \tag{1}$$

in which $\Theta^{\mathcal{C}}$ denotes the parameter vector in the joint space of the kinematic chain from $\mathcal{M}$ to $\mathcal{C}$. If no joints are involved in the kinematic chain from $\mathcal{M}$ to $\mathcal{C}$, then $\mathcal{C}$ is fixed in $\mathcal{M}$ and $g_{\mathcal{MC}}$ is a constant determined by the geometry of the module. $g_{\mathcal{CM}}$ is just the inverse of $g_{\mathcal{MC}}$.

3.3 Kinematics for Chains

A kinematic chain from frame $\mathcal{S}$ to $\mathcal{F}$ can be obtained as $G_K : \mathcal{S} \rightsquigarrow \mathcal{F}$ where $\mathcal{S}$ and $\mathcal{F}$ are two vertices of G_K. In this kinematic chain, all homogeneous transformations between connectors (e.g., $g_{\mathcal{T}_1\mathcal{B}_2}$ in Fig. 5c) are fixed and can be easily computed. The relative orientation between connectors is determined by examining the connector design. For example, there are four cases for connecting SMORES-EP modules shown in Fig. 6b — Fig. 6e due to the arrangement

[1] Refer to Chapter 2 in [26] for background.



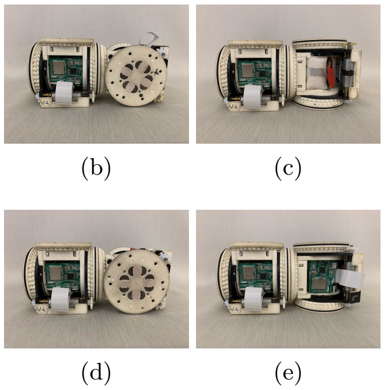


Fig. 6: (a) Kinematics for SMORES modules. (b) — (e) Four cases to connect $\mathcal{R}$ and $\mathcal{T}$.

of the magnets on the connector. The homogeneous transformation $g_{\mathcal{SF}}$ can be computed by multiplying the homogeneous transformation of each edge of path $G_K : \mathcal{S} \rightsquigarrow \mathcal{F}$ in order. In particular, let $\mathcal{S}$ be world frame $\mathcal{W}$, if module $m_1, m_2, \cdots, m_N$ are involved in this chain, then the position of the origin of $\mathcal{F}$ in $\mathcal{W}$ is given by

$$p^{\mathcal{W}}_{\mathcal{F}} = g_{\mathcal{WF}} \begin{bmatrix} 0 \; 0 \; 0 \; 1 \end{bmatrix}^{\intercal} \tag{2}$$

and the instantaneous spatial velocity of $\mathcal{F}$ is given by the twist

$$\widehat{V}^s_{\mathcal{WF}} = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \left(\frac{\partial g_{\mathcal{WF}}}{\partial \theta_{ij}} g^{-1}_{\mathcal{WF}} \right) \dot{\theta}_{ij} \tag{3}$$

in which θ_{ij} is the jth joint parameter of module m_i involved in this chain and the number of joints of module m_i involved in this chain is N_i. Rewrite Eq. (3) in twist coordinates as

$$V^s_{\mathcal{WF}} = J^s_{\mathcal{WF}} \dot{\Theta}^{\mathcal{WF}} \tag{4}$$

in which

$$\Theta^{\mathcal{WF}} = [\theta_{11} \cdots \theta_{1N_1} \; \theta_{21} \cdots \theta_{2N_1} \; \cdots \; \theta_{N1} \cdots \theta_{NN_n}]^{\intercal} \tag{5}$$

$$J^s_{\mathcal{WF}} = \begin{bmatrix} J_1 \; J_2 \cdots J_N \end{bmatrix} \tag{6}$$

$$J_i = \left[\left(\tfrac{\partial g_{\mathcal{WF}}}{\partial \theta_{i1}} g^{-1}_{\mathcal{WF}} \right)^{\vee} \left(\tfrac{\partial g_{\mathcal{WF}}}{\partial \theta_{i2}} g^{-1}_{\mathcal{WF}} \right)^{\vee} \cdots \left(\tfrac{\partial g_{\mathcal{WF}}}{\partial \theta_{iN_i}} g^{-1}_{\mathcal{WF}} \right)^{\vee} \right] \tag{7}$$

and $J^s_{\mathcal{WF}}$ is the *spatial chain Jacobian.*

Define the twist of the jth joint of module m_i with respect to $\mathcal{W}$ as ξ'_{ij} that is

$$\xi'_{ij} = \left(\frac{\partial g_{\mathcal{WF}}}{\partial \theta_{ij}} g^{-1}_{\mathcal{WF}} \right)^{\vee} = \mathrm{Ad}_{g_{\mathcal{WM}_i}} \xi_{ij}$$

in which $\mathrm{Ad}_{g_{\mathcal{W}\mathcal{M}_i}}$ is the adjoint transformation[2] and ξ_{ij} is defined in Sec. 3.2 for each joint in a module with respect to its module body frame. Then J_i becomes

$$J_i = \left[\xi'_{i1}\ \xi'_{i2} \cdots \xi'_{iN_i}\right] \tag{8}$$

With this spatial chain Jacobian, the velocity of the origin of frame $\mathcal{F}$ is

$$v^s_{\mathcal{F}} = \widehat{V}^s_{\mathcal{W}\mathcal{F}} p^{\mathcal{W}}_{\mathcal{F}} = \left(J^s_{\mathcal{W}\mathcal{F}}\dot{\Theta}^{\mathcal{W}\mathcal{F}}\right)^{\wedge} p^{\mathcal{W}}_{\mathcal{F}} \tag{9}$$

For a module m_i in the kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}$ ($\mathcal{M}_i$ is a vertex in the corresponding path), a sub-kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{M}_i$ can be defined with joint parameter vector $\Theta^{\mathcal{W}\mathcal{M}_i} = \left[\theta_{11}, \theta_{12}, \cdots, \theta_{\bar{i}\bar{j}_i}\right]^{\mathsf{T}}$ where $\theta_{\bar{i}\bar{j}_i}$ is the parameter of the $\bar{j}_i$th joint of module $m_{\bar{i}}$. For example, take the sub-kinematic chain from $\mathcal{W}$ to $\mathcal{M}_2$ in Fig. 5c, then $i = 2$, $\bar{i} = 1$, $\bar{j}_i = 1$, since there is only one joint between $\mathcal{W}$ and $\mathcal{M}_2$ which is the 1st joint of module m_1. Then the *spatial module Jacobian* $J^s_{\mathcal{W}\mathcal{M}_i}$ or $J^s_{\mathcal{M}_i}$ for simplicity can be defined as

$$J^s_{\mathcal{M}_i} = \left[\xi'_{11}\ \xi'_{12} \cdots \xi'_{\bar{i}\bar{j}_i}\right] \tag{10}$$

and the velocity of the origin of $\mathcal{M}_i$ is

$$v^s_{\mathcal{M}_i} = \left(J^s_{\mathcal{M}_i}\dot{\Theta}^{\mathcal{W}\mathcal{M}_i}\right)^{\wedge} p^{\mathcal{W}}_{\mathcal{M}_i} \tag{11}$$

By replacing all twists associated with joints after the $\bar{j}_i$th joint of module $m_{\bar{i}}$ in the spatial chain Jacobian of chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}$ with 6×1 zero vectors, the spatial module Jacobian can also be written as

$$J^s_{\mathcal{M}_i} = \left[\xi'_{11}\ \xi'_{12} \cdots \xi'_{\bar{i}\bar{j}_i}\ 0_{6\times1} \cdots 0_{6\times1}\right] \tag{12}$$

then the velocity of the origin of $\mathcal{M}_i$ is represented as

$$v^s_{\mathcal{M}_i} = \left(J^s_{\mathcal{M}_i}\dot{\Theta}^{\mathcal{W}\mathcal{F}}\right)^{\wedge} p^{\mathcal{W}}_{\mathcal{M}_i} \tag{13}$$

4 Control and Motion Planning

4.1 Control

Given the kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}$, the goal of the control task is to move $p^{\mathcal{W}}_{\mathcal{F}}$ (or $p_{\mathcal{F}}$ for simplicity) — the position of $\mathcal{F}$ — to follow a desired trajectory.

Let $\tilde{p}_{\mathcal{F}} = \tilde{p}_{\mathcal{F}}(t)$ be the desired trajectory for the robot to track and $\tilde{v}^s_{\mathcal{F}}$ (or $\tilde{v}_{\mathcal{F}}$ for simplicity) is the derivative of $\tilde{p}_{\mathcal{F}}$, and the error and its derivative are defined as

$$e = \tilde{p}_{\mathcal{F}} - p_{\mathcal{F}}$$
$$\dot{e} = \dot{\tilde{p}}_{\mathcal{F}} - \dot{p}_{\mathcal{F}} = \tilde{v}_{\mathcal{F}} - v_{\mathcal{F}}$$

[2] Refer to Chapter 2 in [26] for adjoint transformation definition.

The error e can converge exponentially to zero as long as it satisfies

$$\dot{e} + Ke = 0 \tag{14}$$

in which K is positive definite. Substitute e and $\dot{e}$

$$\tilde{v}^s_{\mathcal{F}} - v^s_{\mathcal{F}} + K(\tilde{p}_{\mathcal{F}} - p_{\mathcal{F}}) = 0 \tag{15}$$

With Eq. (9), Eq. (15) can be rewritten as

$$\left(J^s_{\mathcal{W}\mathcal{F}} \dot{\Theta}^{\mathcal{W}\mathcal{F}}\right)^{\wedge} p_{\mathcal{F}} = \tilde{v}^s_{\mathcal{F}} + K(\tilde{p}_{\mathcal{F}} - p_{\mathcal{F}}) \tag{16}$$

Eq. (16) is the control law to control the position of frame $\mathcal{F}$, namely $\dot{\Theta}^{\mathcal{W}\mathcal{F}}$ (or $\dot{\Theta}^{\mathcal{F}}$ for simplicity) — the velocities of all involved joints that satisfy this equation — can move $p_{\mathcal{F}}$ to $\tilde{p}_{\mathcal{F}}$ in exponential time.

Suppose there are α motion goals $\tilde{p}_{\mathcal{F}_1}, \tilde{p}_{\mathcal{F}_2}, \cdots, \tilde{p}_{\mathcal{F}_\alpha}$, then the control law for all motion goals can be written as

$$\mathbf{J}\mathbf{P} = \widetilde{\mathbf{V}} + \mathbf{K}(\widetilde{\mathbf{P}} - \mathbf{P}) \tag{17}$$

which is the stack of Eq. (16) for each motion goal. This makes the control problem for multiple motion goals easier without considering the fact that some motion goals may be coupled. That is, some kinematic chains may share DoFs. We need only build an Eq. (16) for each individual motion goal and then stack them as linear constraints. Building a specific model for different combinations of motion goals is not necessary.

Recall that a modular robotic system is usually redundant so that there can be an infinite number of solutions to Eq. (17). This problem is formulated as a quadratic program

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\dot{\Theta}^{\mathsf{T}}\dot{\Theta} \\ \text{subject to} \quad & \mathbf{J}\mathbf{P} = \widetilde{\mathbf{V}} + \mathbf{K}(\widetilde{\mathbf{P}} - \mathbf{P}) \end{aligned} \tag{18}$$

where Θ is the set of joint parameters in kinematic chains $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_1$, $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_2, \cdots, G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_\alpha$. Then solving (18) yields the minimum norm solution of joint velocities at every moment.

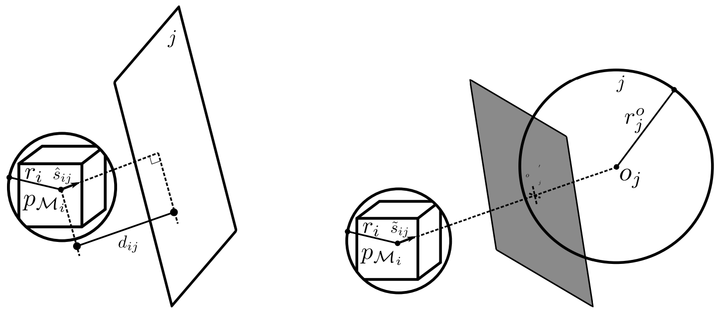
The joint position and velocity limits can be added to the quadratic program as inequality constraints

$$\frac{\Theta_{\min} - \Theta}{\Delta t} \leq \dot{\Theta} \leq \frac{\Theta_{\max} - \Theta}{\Delta t} \tag{19}$$

$$\dot{\Theta}_{\min} \leq \dot{\Theta} \leq \dot{\Theta}_{\max} \tag{20}$$

in which Δt is the time duration for the current step. Due to these two constraints, K cannot be too aggressive or solutions may not be obtained.

This optimization approach is helpful for many types of motion tasks. The controller can be used to move $p_{\mathcal{F}}$ to a desired position $\tilde{p}_{\mathcal{F}}$ by setting $\tilde{v}^s_{\mathcal{F}} = 0$, and it can also control $p_{\mathcal{F}}$ to move at a desired velocity by increasing $\tilde{p}_{\mathcal{F}}$ by $\tilde{v}_{\mathcal{F}}\Delta t$ for every time step.



(a) (b)

Fig. 7: (a) Environment boundary. (b) Sphere obstacle avoidance.

4.2 Motion Planning

The goal of the motion planning task is to enable a cluster of modules to navigate collision-freely in an environment with obstacles.

Frame Boundaries The cluster of modules can be kept in any polyhedral region in space which is defined by the boundaries of the environment. For a module m_i in the kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}$, let $\hat{s}_{ij}$ be the unit direction vector from $p^{\mathcal{W}}_{\mathcal{M}_i}$ (or $p_{\mathcal{M}_i}$ for simplicity) — the origin of $\mathcal{M}_i$ in world frame $\mathcal{W}$ — to the jth face of the environment polyhedron perpendicular with distance d_{ij}, then if we enforce the constraint

$$v^s_{\mathcal{M}_i} \bullet \hat{s}_{ij} = (J^s_{\mathcal{M}_i}\dot{\Theta})^\wedge p_{\mathcal{M}_i} \bullet \hat{s}_{ij} \leq d_{ij} \tag{21}$$

for every side of the environment polyhedron, $p_{\mathcal{M}_i}$ will never cross the boundary of the environment as long as this kinematic chain follows the velocity for much less than 1 second. Using a sphere with radius r_i to approximate the geometry size of module m_i, then the constraint

$$v^s_{\mathcal{M}_i} \bullet \hat{s}_{ij} = (J^s_{\mathcal{M}_i}\dot{\Theta})^\wedge p_{\mathcal{M}_i} \bullet \hat{s}_{ij} \leq d_{ij} - r_i \tag{22}$$

will ensure that the module body will always be inside the environment boundaries (Fig. 7a). Thus, applying constraint (22) to all modules in the kinematic chain will ensure the chain will stay inside the environment.

Obstacle Avoidance It is hard to represent the collision-free space analytically in joint space due to the high DoFs of modular robotic systems. Here we propose an alternative. The obstacles can be approximated by a set of spheres using a sphere-tree construction algorithm [4]. Similar ideas have been explored in [7,40]. There are two issues using this idea. This collision-avoidance constraint is modeled as the condition that the distance between every sphere approximating the robot and every sphere approximating the obstacles is greater than the sum of their radius. This leads to quadratic constraints which are not suitable for real-time applications of large systems due to numerical issues. In addition, in order to approximate obstacles with decent accuracy, many spheres have to be generated. For example, a block object shown in Fig. 8a is constructed by

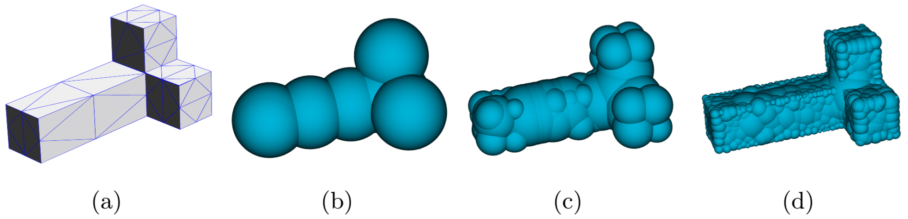


Fig. 8: A block obstacle (a) is approximated with 3 levels of spheres. (b) 8 spheres in level 1. (c) 64 spheres in level 2. (d) 470 spheres in level 3.

multiple spheres. A more accurate approximation of this object requires more spheres (Fig. 8b — Fig. 8d). An advantage of this approach is that obstacles automatically have some level of buffer that can further guarantee motion safety. However, using a small number of spheres to approximate an obstacle can lead to too conservative planning space not finding collision-free paths when they exist. On the other hand, if there are a large number of spheres, there will also be a large number of constraints which can prohibit the optimization problem being solved efficiently without numerical issues.

In this paper, the obstacle avoidance requirement is modeled as linear constraints which are efficient to solve stably. For a module m_i in the kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}$, let $\tilde{s}_{ij}$ be the unit direction vector from $p_{\mathcal{M}_i}$ to the center of the jth obstacle sphere o_j in world frame $\mathcal{W}$ with radius r^o_j. Imaging a plane P_{ij} with $\tilde{s}_{ij}$ as its normal vector and o'_j being the point of tangency to this sphere, then if we enforce the constraint

$$v^s_{\mathcal{M}_i} \bullet \tilde{s}_{ij} = (J^s_{\mathcal{M}_i}\dot{\Theta})^\wedge p_{\mathcal{M}_i} \bullet \tilde{s}_{ij} \leq \|o'_j - p_{\mathcal{M}_i}\| - r_i \tag{23}$$

in which $o'_j = o_j - r^o_j s_{ij}$ for every obstacle sphere, $p_{\mathcal{M}_i}$ will never touch an obstacle (Fig. 7b). In order to enable the system to safely navigate the environment, we need to apply this constraint for every module.

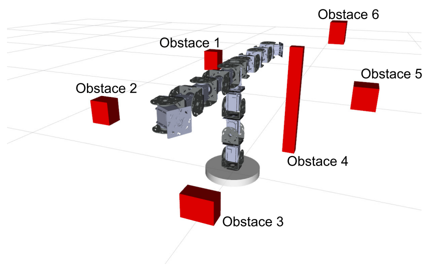
In order to resolve the difficulty that there can be a large number of obstacle spheres leading to a large number of constraints, we present a novel way to significantly simplify these constraints as a pre-processing step for optimization. As mentioned, for module m_i and the jth obstacle sphere, we can compute an obstacle plane P_{ij} to build a linear constraint. If another obstacle sphere and module m_i are perfectly separated by P_{ij}, then this obstacle sphere can be ignored for m_i at the current planning step, because as long as module m_i never intersects with obstacle plane P_{ij}, m_i cannot touch this obstacle sphere. By this simple rule, we can refine the set of obstacle spheres by iteratively applying an erase-remove idiom technique efficiently for real-time performance. For example, a robot configuration built by fourteen CKBot UBar modules is placed in a cluttered environment where there are six obstacles (Fig. 9a). After applying the sphere-tree construction algorithm, we derive 373 obstacle spheres in total shown in Fig. 9b. For the module circled in the figure, after refining the set of all obstacle spheres, we only need to consider five obstacle spheres. Their obstacle planes are shown in Fig. 9b. In the current step, Obstacle 1 and Obstacle 6 are behind other obstacles so they can be ignored. And for the rest of the obstacles, we only need five spheres to approximate these obstacle-avoidance constraints.

4.3 Integrated Control and Motion Planning

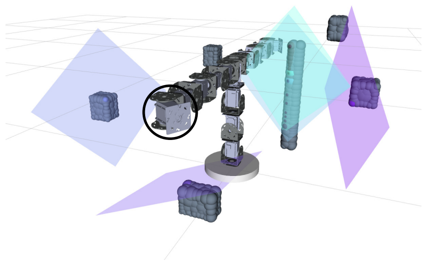
With the control law in Sec. 4.1 and the motion constraints in Sec. 4.2, we can formalize the control and motion planning problem for multiple kinematic chains $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_i, i = 1, 2, \cdots, \alpha$ as the following quadratic program with linear constraints

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\dot{\Theta}^{\intercal}\dot{\Theta} \\
\text{subject to} \quad & \mathbf{JP} = \widetilde{\mathbf{V}} + \mathbf{K}(\widetilde{\mathbf{P}} - \mathbf{P}) \\
& \frac{\Theta_{\min} - \Theta}{\Delta t} \leq \dot{\Theta} \leq \frac{\Theta_{\max} - \Theta}{\Delta t} \\
& \dot{\Theta}_{\min} \leq \dot{\Theta} \leq \dot{\Theta}_{\max} \\
& (J^s_{\mathcal{M}_i}\dot{\Theta})^{\wedge} p_{\mathcal{M}_i} \bullet \hat{s}_{ij} \leq d_{ij} - r_i \\
& \qquad \forall(\mathcal{M}_i, f_j) \in V_K \times F \\
& (J^s_{\mathcal{M}_i}\dot{\Theta})^{\wedge} p_{\mathcal{M}_i} \bullet \tilde{s}_{ik} \leq \|o'_k - p_{\mathcal{M}_i}\| - r_i \\
& \qquad \forall(\mathcal{M}_i, S_k) \in V_K \times \mathbf{S}_i
\end{aligned} \tag{24}$$

in which F is the set of all faces of the environment polyhedron and f_j is the jth face, $\mathbf{S}$ is the set of all spheres approximating the environment obstacles, $\mathbf{S}_i \subseteq \mathbf{S}$ is the current set of obstacle spheres under consideration for module m_i, and S_k is the kth sphere in $\mathbf{S}_i$. By solving this quadratic program, the minimum norm solution that satisfies the hardware limits, control requirement, and motion constraints can be obtained for the current time step given the current state of every kinematic chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_i$ where $i = 1, 2, \cdots, \alpha$, the desired velocity, and the position of the origin of each frame $\mathcal{F}_i$.



(a)



(b)

Fig. 9: (a) A configuration built by 14 CKBot UBar modules is placed in a cluttered environment with 6 obstacles. (b) Apply the sphere-tree construction algorithm on all obstacles and the total number of obstacle spheres is 373. For the module inside the circle at its current state, only 5 obstacle spheres are necessary for collision avoidance and their obstacle planes are shown.

This formulation can be used for motion tasks with simple constraints (e.g., when obstacles are far from robots and motion goals). The equality constraint enables the motion goal to be achieved very fast with suitable gains. However, this can also cause difficulties for optimization. For complicated scenarios, we propose a sequential convex optimization formulation in which the objective function and constraints are updated when encountering obstacles. Initially the objective function is in the following form

$$f(\dot{\Theta}) = \|\dot{\Theta}\|^2 + \lambda\|\mathbf{J}\mathbf{P} - (\widetilde{\mathbf{V}} + \mathbf{K}(\widetilde{\mathbf{P}} - \mathbf{P})\|^2 \tag{25}$$

in which λ is a weight to address the significance of the feedback controller. In order to avoid entering space that is hard to maintain safety, we penalize aggressive motions towards obstacles by adding $\|v^s_{\mathcal{M}_i} \bullet s_{ij}\|^2$ to the objective function when the distance between module body frame $\mathcal{M}_i$ of the robot and the jth obstacle is less than $d_{\min}$, and the objective function becomes

$$f(\dot{\Theta}) = \|\dot{\Theta}\|^2 + \lambda\|\mathbf{J}\mathbf{P} - (\widetilde{\mathbf{V}} + \mathbf{K}(\widetilde{\mathbf{P}} - \mathbf{P})\|^2 + \mu_{ij}\|v^s_{\mathcal{M}_i} \bullet s_{ij}\|^2 \tag{26}$$

in which μ_{ij} is also a weight. The new penalizing term means that we want this module m_i to minimize its motion towards the jth obstacle sphere. We have to check every module to update the objective function and this can be computed easily by sphere-to-sphere distance.

If module $\mathcal{M}_i$ makes contact with the jth obstacle sphere, this module will be forced to move away by defining a repulsive velocity as the normal to the obstacle plane. This can be done by adding a hard inequality constraint

$$v^s_{\mathcal{M}_i} \bullet s_{ij} = \left(J^s_{\mathcal{M}_i}\dot{\Theta}\right)^{\wedge} p_{\mathcal{M}_i} \bullet s_{ij} \leq \gamma_{ij} \tag{27}$$

in which γ_{ij} is bounded between $-\|v^s_{\mathcal{M}_i}\|$ and 0. This constraint can force module body frame $\mathcal{M}_i$ to move away from the jth obstacle sphere. Note that the previously added penalizing term for this module $\mu_{ij}\|v^s_{\mathcal{M}_i} \bullet s_{ij}\|^2$ is removed from the objective function. The larger γ_{ij} is, the faster the module $\mathcal{M}_i$ moves away from the jth obstacle.

4.4 Iterative Algorithm for Manipulation Planning

The set of module graph $\mathbf{G}_m$ described in Sec. 3.1 and the twist set $\boldsymbol{\xi}$ described in Sec. 3.2 associated with all the joints in different type of modules are computed and stored. For a modular robot configuration G, assuming the base module $\bar{m}$ and how it is attached to the world frame $\mathcal{W}$ as well as the motion goals $\tilde{p}_{\mathcal{F}_1}$, $\tilde{p}_{\mathcal{F}_2}$, $\cdots$, $\tilde{p}_{\mathcal{F}_\alpha}$ for frame $\mathcal{F}_1$, $\mathcal{F}_2$, $\cdots$, $\mathcal{F}_\alpha$ respectively are known, the set of all faces of the environment polyhedron is F and the set of all spheres approximating environmental obstacles is $\mathbf{S}$, the full algorithm framework is shown in Algorithm 1 with following functions:

- `BFS`$(G, \mathbf{G}_m, \bar{m})$: Traverse a modular robotic configuration G in breadth-first-search order starting from $\bar{m}$ to construct the kinematics graph $G_K = (V_K, E_K)$;

```
Algorithm 1: Control and Motion Planning
   Input: $\boldsymbol{\xi}$, $\mathbf{G}_m$, $\bar{m}$, $\mathcal{F}_1$, $\mathcal{F}_2$, $\cdots$, $\mathcal{F}_\alpha$, $\{\tilde{p}_{\mathcal{F}_i}(t) | 0 \le t \le T, i = 1, 2, \cdots, \alpha\}$, $F$, $\mathbf{S}$
   Output: result
 1 $G_K \leftarrow$ BFS($G, \mathbf{G}_m, \bar{m}$);
 2 $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_i \leftarrow$ GetChain($G_K, \mathcal{F}_i$), $i \in [1, \alpha]$;
 3 Initialize $\Theta$;
 4 Initialize $\mathbf{K}$ and $\Delta t$;
 5 $t \leftarrow 0$;
 6 while $\sum_{i=1}^{\alpha} \|p_{\mathcal{F}_i} - \tilde{p}_{\mathcal{F}_i}(T)\| \ge \epsilon$ do
 7     Compute $\hat{s}_{ij}$ $\forall (\mathcal{M}_i, f_j) \in V_K \times F$;
 8     Compute $\mathbf{S}_i \subseteq \mathbf{S}$ $\forall \mathcal{M}_i \in V_K$ and $\tilde{s}_{ik}$ $\forall S_k \in \mathbf{S}_i$;
 9     $\dot{\Theta} \leftarrow$ SolveQP($G_K, \mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_\alpha, \widetilde{\mathbf{P}}(t), \widetilde{\mathbf{V}}(t), \mathbf{P}, \mathbf{K}, \Delta t$);
10     if $\dot{\Theta} =$ Null then
11         return result $\leftarrow$ False;
12     end
13     Publish $\dot{\Theta}$ to the system;
14     $t \leftarrow t + \Delta t$;
15 end
16 return result $\leftarrow$ True;
```

- `GetChain`$(G_K, \mathcal{F})$: Return the kinematic chain from $\mathcal{W}$ to $\mathcal{F}$ in G_K;
- `SolveQP`$(G_K, \mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_\alpha, \widetilde{\mathbf{P}}(t), \widetilde{\mathbf{V}}(t), \mathbf{P}, \mathbf{K}, \Delta t)$: Construct and try to solve the quadratic program described in Sec. 4.3. If failed to solve this program, then return `Null` as an invalid solution.

After initializing all the parameters, compute the unit direction vector $\hat{s}_{ij}$ between every $\mathcal{M}_i \in V_K$ and every face of the environment $f_j \in F$, compute $\mathbf{S}_i \subseteq \mathbf{S}$ for every $\mathcal{M}_i \in V_K$ and the corresponding unit direction vector $\tilde{s}_{ik}$ between $\mathcal{M}_i$ and every obstacle sphere $S_k \in \mathbf{S}_i$. If there is no valid solution, the program should stop, or the program will continue until every $p_{\mathcal{F}_i}$ is close enough to the destination $\tilde{p}_{\mathcal{F}_i}(T)$. If the trajectory $\tilde{p}_{\mathcal{F}_i}(t)$ is not specified and only $\tilde{p}_{\mathcal{F}_i}(T)$ where $T \to \infty$ is given, then this algorithm can automatically find a trajectory for modules to navigate the environment. The output from the planner can be applied directly online (e.g., running on robot modules) to achieve real-time performance, or can be integrated over time (one-step Euler integration) to generate the trajectory for each module or joint.

5 Experiments

We performed several experiments on two hardware platforms to verify the approach. Here, we show that the framework is able to execute a motion task with guaranteed control performance on real hardware while satisfying all hardware constraints, frame boundary constraint, and obstacle avoidance. Finally the framework is tested in a complex scenario showing its ability for online trajectory optimization for navigation tasks.

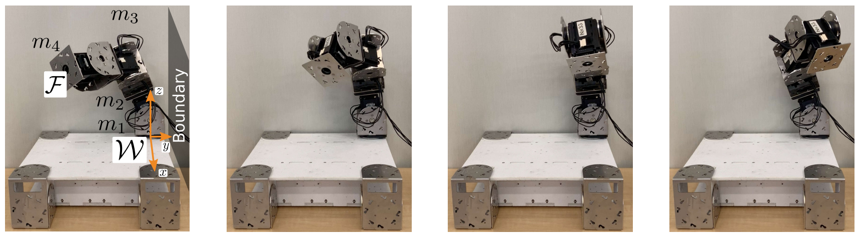


Fig. 10: Control $p_{\mathcal{F}}$ to follow a given trajectory along $+y$-axis of $\mathcal{W}$ by 15 cm from the initial pose (a) to the final pose (d). All the modules have to be on the left side of the boundary. m_1, m_2, and m_3 have to approach the boundary first (b) and then move away from the boundary (c) to finish the task.

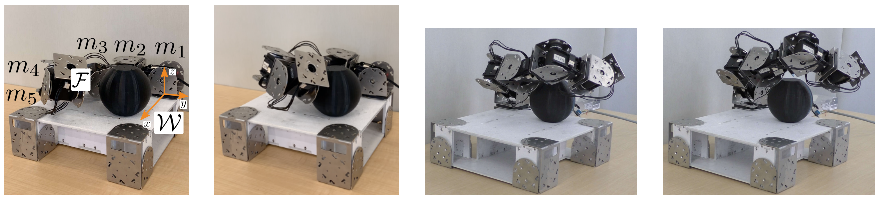
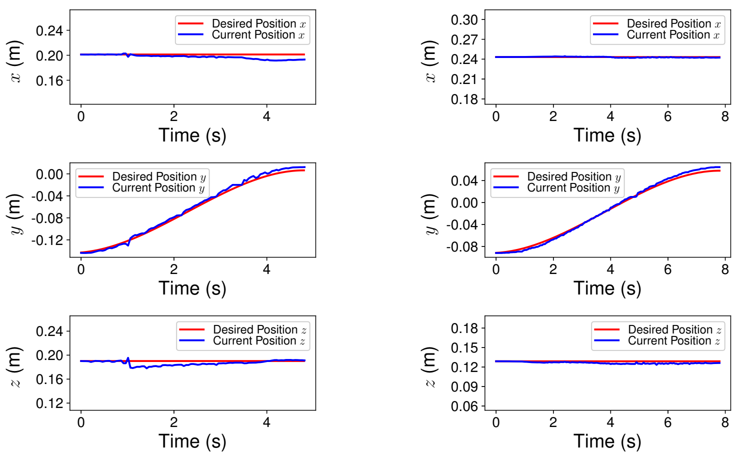


Fig. 11: Control $p_{\mathcal{F}}$ from its initial pose (a) to its final pose (d) by both following a given trajectory along $+y$-axis of $\mathcal{W}$ by 15 cm and navigating to the destination directly. The modules have to move around the sphere obstacle while executing these two tasks.

5.1 Real-Time Control

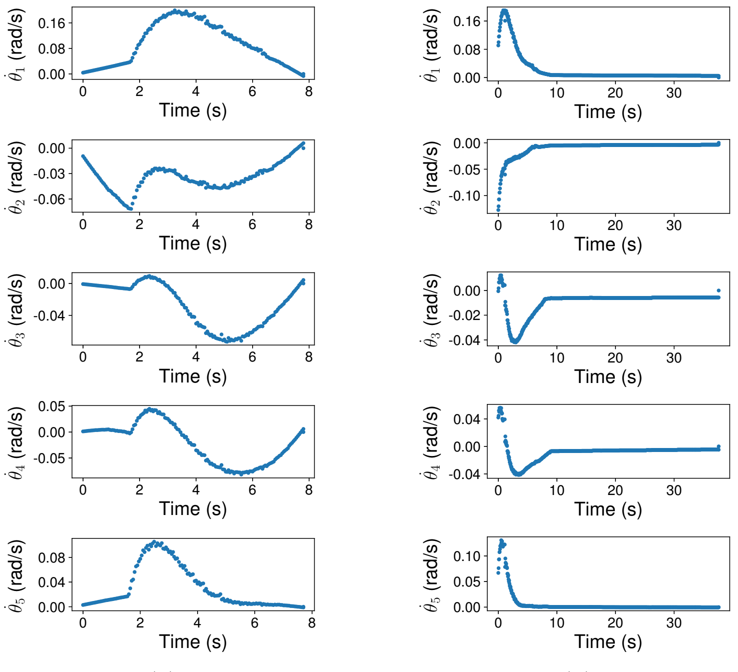
CKBot Chain A configuration with four CKBot UBar modules is shown in Fig. 10a. The base module $\bar{m} = m_1$ is attached to the world frame $\mathcal{W}$ and frame $\mathcal{F}$ is attached to connector $\mathcal{T}$ of module m_4. A virtual frame boundary is next to the right side of the base. The task is to control $p_{\mathcal{F}}$ to follow a given trajectory to the position shown in Fig. 10d. Another experiment setup with five CKBot UBar modules is shown in Fig. 11a. The black sphere is an obstacle, the base module $\bar{m} = m_1$ and frame $\mathcal{F}$ is attached to connector $\mathcal{T}$ of module m_5. Two tasks are executed: control $p_{\mathcal{F}}$ to follow a given trajectory and control $p_{\mathcal{F}}$ to approach a specified destination with the final position of $p_{\mathcal{F}}$ as shown in Fig. 11d. The control loop runs at 20 Hz with gain $K = \mathrm{diag}(1, 1, 1)$. Fig. 12 and Fig. 14a shows $p_{\mathcal{F}}(t)$ and $\tilde{p}_{\mathcal{F}}$ of these three tests demonstrating tracking and navigation performance. The velocity commands for all modules in these two five-module demonstrations are shown in Fig. 13 and all commands are within the constraints of each module. Modules move more aggressively at the beginning when executing the destination navigation task in order to quickly approach the destination.

SMORES-EP Chain The experiment setup with four SMORES-EP modules is shown in Fig. 15a. The base module $\bar{m} = m_1$ is fixed to the world frame



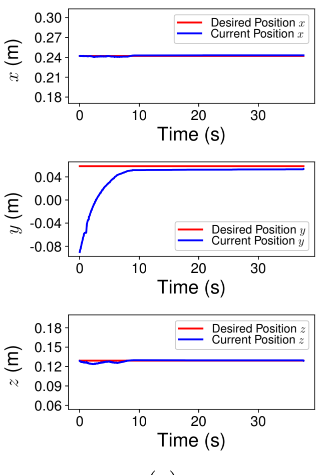
(a) (b)

Fig. 12: The motion of $p_{\mathcal{F}}$: (a) the four-module task; (b) the five-module trajectory following task.

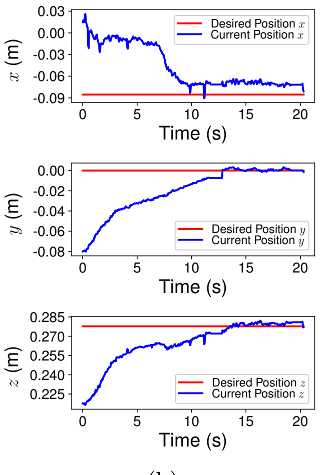


(a) (b)

Fig. 13: The control input $\dot{\Theta}$ for the five-module chain experiment: (a) the trajectory following task; (b) the destination navigation task.

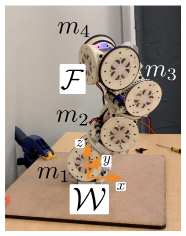


(a)

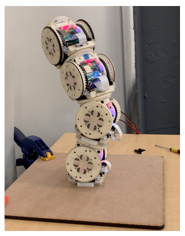


(b)

Fig. 14: The motion of $p_{\mathcal{F}}$: (a) the CKBot five-module destination navigation task; (b) the SMORES-EP four-module chain destination navigation task.



(a)



(b)

Fig. 15: Control a chain of SMORES-EP modules to navigate from its initial position (a) to a goal position (b). This chain is constructed by four modules with 16 DoFs.

$\mathcal{W}$ and frame $\mathcal{F}$ is attached to connector $\mathcal{T}$ of module m_4. This system has 16 DoFs and the task is to control $p_{\mathcal{F}}$ to navigate to a specified destination shown in Fig. 15b. The control loop runs at 20 Hz and the gain $K = \text{diag}(0.5, 0.5, 0.5)$. The experiment result $p_{\mathcal{F}}(t)$ is shown in Fig. 14b. The position sensors installed in SMORES-EP modules are customizable potentiometers using paints [36,22]. These low-cost sensors with a modified Kalman filter for nonlinear systems are used to provide position information of each DoF. Due to the limitations of the hardware, some noise is evident.

CKBot Branch A configuration with nine CKBot UBar modules is shown in Fig. 16a. The base module $\bar{m} = m_1$ is fixed to the world frame $\mathcal{W}$. Frame $\mathcal{F}_1$ is attached to connector $\mathcal{T}$ of module m_6 and frame $\mathcal{F}_2$ is attached to connector $\mathcal{T}$ of module m_9. Chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_1$ and $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_2$ have common parts composed by module m_1, m_2, and m_3. The task is to control $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$

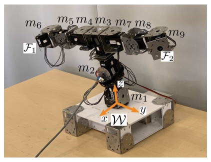
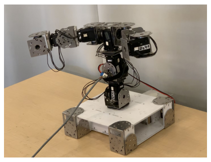
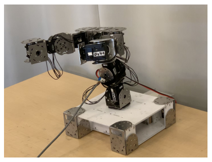
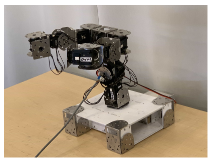

(a)

(b)

(c)

(d)

Fig. 16: Control $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ to follow two given trajectories respectively from their initial poses (a) to their final poses (d). Module m_1, m_2, and m_3 initially have to move backward (b) and then move forward (c) in order to control $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ to follow their trajectories.

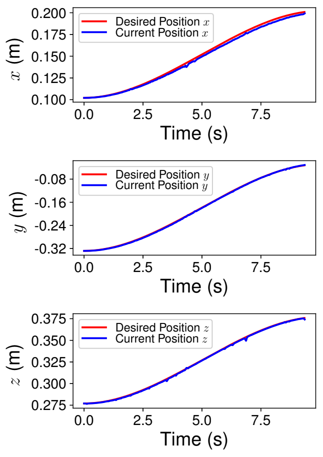
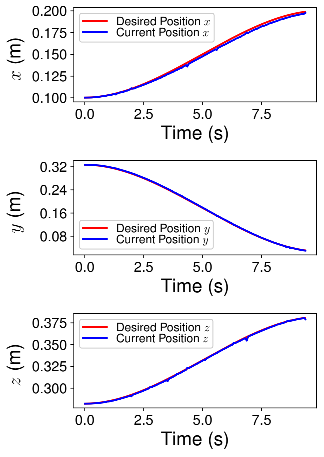
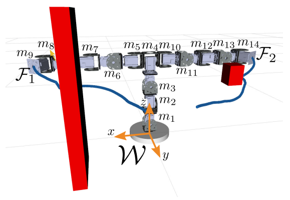

(a)

(b)

Fig. 17: (a) The tracking result for $p_{\mathcal{F}_1}$. (b) The tracking result for $p_{\mathcal{F}_2}$.

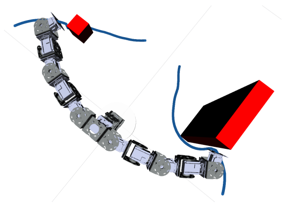
to follow trajectories respectively to the pose shown in Fig. 16d. The control loop runs at 20 Hz and the gain is diag(0.1, 0.1, 0.1) for both motion goals. The tracking performance is shown in Fig. 17a and Fig. 17b.

5.2 Whole-Body Manipulation

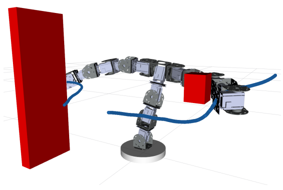
A configuration with fourteen CKBot UBar modules is constructed in a simulation environment shown in Fig. 18a. The base module $\bar{m} = m_1$ is fixed to the world frame $\mathcal{W}$. There are two obstacles in the workspace which are close to the robot. The sphere-tree construction outputs 126 obstacle spheres in total to approximate these two obstacles. Frame $\mathcal{F}_1$ and $\mathcal{F}_2$ are attached to connector $\mathcal{T}$ of module m_9 and module m_{14} respectively. Similarly, Chain $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_1$ and $G_K : \mathcal{W} \rightsquigarrow \mathcal{F}_2$ share four modules. The task is to control $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ to new locations between these two obstacles. The control loop runs at 20 Hz and the gain is diag(0.1, 0.1, 0.1) for both motion goals. In this complex scenario, the quadratic program can be constructed and solved by Gurobi [10] in 6.3 ms in



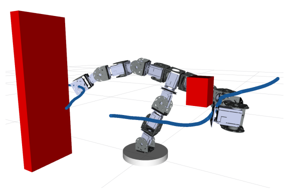
(a)



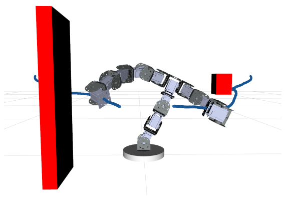
(b)



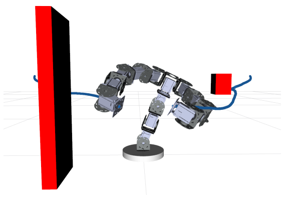
(c)



(d)

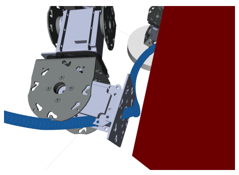


(e)

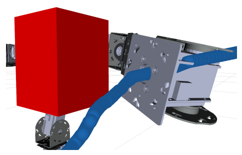


(f)

Fig. 18: Control $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ from the initial pose (a) to new locations between the obstacles (f). The body composed by module m_1, m_2, and m_3 first moves backward a little bit (b) and then moves to one side in order to help $\mathcal{F}_1$ and $\mathcal{F}_2$ to go around obstacles (c) — (e). After going around obstacles, both frames can navigate quickly to their destinations. The planned trajectories are shown as blue lines.



(a)

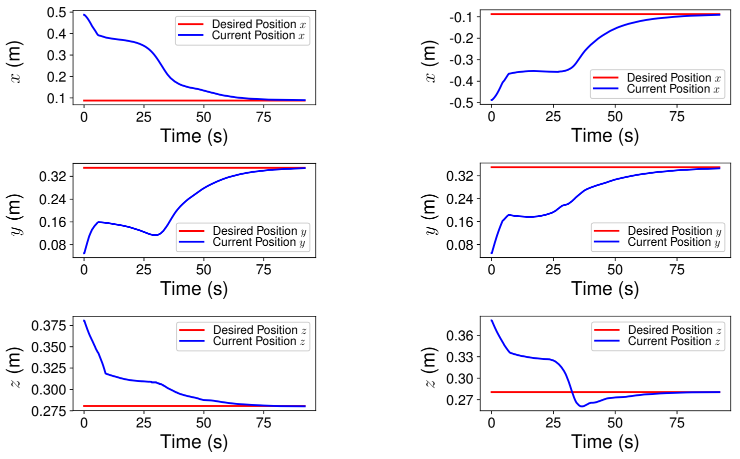


(b)

Fig. 19: (a) Module m_9 approaches an obstacle. (b) Module m_{14} approaches an obstacle.

average with standard deviation of 2.3 ms and a maximum time of 15.5 ms on a laptop computer (Intel Core i7-8750H CPU, 16GB RAM).

Initially the motion of $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ are symmetric because modules are all not very close to obstacles. At this stage, the main body composed by module m_1, m_2, and m_3 moves backward slightly. Frame $\mathcal{F}_1$ approaches one of the obstacle first (Fig. 18b and Fig. 19a), and the objective function is updated to penalize the motion of m_9 which has to move along the obstacle. Then module m_{14} approaches the other obstacle (Fig. 18c and Fig. 19b), and a penalty term for this module is also added to the objective function. Both frames move slowly during this phase (Fig. 18c and Fig. 18d). We can see from Fig. 20 that $p_{\mathcal{F}_1}$ and $p_{\mathcal{F}_2}$ change slowly. Repulsive motion constraints are added to the optimization function when some module nearly contact obstacles. The main body leans to one



(a) (b)

Fig. 20: (a) The motion of $p_{\mathcal{F}_1}$. (b) The motion of $p_{\mathcal{F}_2}$.

side to help both frames to go around obstacles. After moving around obstacles slowly, both frames can quickly navigate to their destinations (Fig. 18e and Fig. 18f). The final planned trajectory takes 92.15 s.

6 Conclusion

We present a new approach to online manipulation motion planning well suited for reconfigurable modular robot systems. This approach formulates the motion planning problem as a sequential quadratic program. We propose a novel way to approximate obstacles in the environment considering both accuracy and simplicity so that the obstacle-avoidance requirement can be modeled as a small number of linear constraints. The objective function and constraints are updated according to the current scenario in order to handle a larger range of tasks. All motion constraints are linear that allows it to be applied to real-time control. Multiple strongly coupled motion tasks can be handled easily which is particularly useful for modular robots.

References

1. Agrawal, S.K., Kissner, L., Yim, M.: Joint solutions of many degrees-of-freedom systems using dextrous workspaces. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164). vol. 3, pp. 2480–2485 vol.3 (May 2001)
2. Amato, N.M., Wu, Y.: A randomized roadmap method for path and manipulation planning. In: Proceedings of IEEE International Conference on Robotics and Automation. vol. 1, pp. 113–120 vol.1 (April 1996)
3. Barraquand, J., Latombe, J.C.: Robot motion planning: A distributed representation approach. The International Journal of Robotics Research **10**(6), 628–649 (1991)

4. Bradshaw, G., O'Sullivan, C.: Adaptive medial-axis approximation for sphere-tree construction. ACM Trans. Graph. **23**(1), 1–26 (jan 2004)
5. Brock, O., Khatib, O.: Elastic strips: A framework for motion generation in human environments. The International Journal of Robotics Research **21**(12), 1031–1052 (2002)
6. Coleman, D.T., Sucan, I.A., Chitta, S., Correll, N.: Reducing the barrier to entry of complex robotic software: a MoveIt! case study. Journal of Software Engineering for Robotics **5**(4), 3–16 (May 2014)
7. Fromherz, M., Hogg, T., Shang, Y., Jackson, W.: Modular robot control and continuous constraint satisfaction. In: Proceedings of IJCAI Workshop on Modelling and Solving Problems with Contraints. pp. 47–56. Seatle, WA (2001)
8. Furcy, D.A.: Speeding Up the Convergence of Online Heuristic Search and Scaling Up Offline Heuristic Search. Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA (2004)
9. Giusti, A., Althoff, M.: Automatic centralized controller design for modular and reconfigurable robot manipulators. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3268–3275 (Sep 2015)
10. Gurobi Optimization: Gurobi Optimizer, `https://www.gurobi.com/products/gurobi-optimizer/`
11. Holmes, P., Kousik, S., Zhang, B., Raz, D., Barbalata, C., Roberson, M.J., Vasudevan, R.: Reachable sets for safe, real-time manipulator trajectory design. In: Proceedings of Robotics: Science and Systems. Corvalis, Oregon, USA (July 2020)
12. Hsu, D., Latombe, J.., Motwani, R.: Path planning in expansive configuration spaces. In: Proceedings of International Conference on Robotics and Automation. vol. 3, pp. 2719–2726 vol.3 (April 1997)
13. Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: Stochastic trajectory optimization for motion planning. In: 2011 IEEE International Conference on Robotics and Automation. pp. 4569–4574 (May 2011)
14. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research **30**(7), 846–894 (2011)
15. Kavraki, L.E., Svestka, P., Latombe, J.., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation **12**(4), 566–580 (Aug 1996)
16. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research **5**(1), 90–98 (1986)
17. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). vol. 1, pp. 473–479 vol.1 (May 1999)
18. Likhachev, M., Gordon, G.J., Thrun, S.: ARA* : Anytime A* with provable bounds on sub-optimality. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems. vol. 16. MIT Press (2004)
19. Likhachev, M., Stentz, A.: R* search. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1. p. 344–350. AAAI'08, AAAI Press (2008)
20. Liu, C., Whitzer, M., Yim, M.: A distributed reconfiguration planning algorithm for modular robots. IEEE Robotics and Automation Letters **4**(4), 4231–4238 (Oct 2019)
21. Liu, C., Yim, M.: A quadratic programming approach to modular robot control and motion planning. In: 2020 Fourth IEEE International Conference on Robotic Computing (IRC). pp. 1–8 (Nov 2020)

22. Liu, C., Tosun, T., Yim, M.: A low-cost, highly customizable solution for position estimation in modular robots. Journal of Mechanisms and Robotics (2021)
23. Liu, C., Yim, M.: Configuration recognition with distributed information for modular robots. In: IFRR International Symposium on Robotics Research (ISRR). Puerto Varas, Chile (Dec 2017)
24. Liu, G., Abdul, S., Goldenberg, A.A.: Distributed control of modular and reconfigurable robot with torque sensing. Robotica **26**(1), 75–84 (2008)
25. Melek, W.W., Goldenberg, A.A.: Neurofuzzy control of modular and reconfigurable robots. IEEE/ASME Transactions on Mechatronics **8**(3), 381–389 (Sep 2003)
26. Murry, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press (1994)
27. Park, C., Pan, J., Manocha, D.: ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In: Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling. pp. 207–215. ICAPS'12, AAAI Press (2012)
28. Quinlan, S., Khatib, O.: Elastic bands: connecting path planning and control. In: [1993] Proceedings IEEE International Conference on Robotics and Automation. pp. 802–807 vol.2 (May 1993)
29. Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning. In: 2009 IEEE International Conference on Robotics and Automation. pp. 489–494 (May 2009)
30. Rimon, E., Koditschek, D.E.: Exact robot navigation using cost functions: the case of distinct spherical boundaries in E^n. In: Proceedings. 1988 IEEE International Conference on Robotics and Automation. pp. 1791–1796 vol.3 (April 1988)
31. Schouwenaars, T., De Moor, B., Feron, E., How, J.: Mixed integer programming for multi-vehicle path planning. In: 2001 European Control Conference (ECC). pp. 2603–2608 (Sep 2001)
32. Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., Abbeel, P.: Motion planning with sequential convex optimization and convex collision checking. The International Journal of Robotics Research **33**(9), 1251–1270 (2014)
33. Seo, J., Yim, M., Kumar, V.: A theory on grasping objects using effectors with curved contact surfaces and its application to whole-arm grasping. The International Journal of Robotics Research **35**(9), 1080–1102 (2016)
34. Shankar, K., Burdick, J.W., Hudson, N.H.: A Quadratic Programming Approach to Quasi-Static Whole-Body Manipulation, pp. 553–570. Springer International Publishing, Cham (2015)
35. Shiller, Z., Dubowsky, S.: On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. IEEE Transactions on Robotics and Automation **7**(6), 785–797 (Dec 1991)
36. Tosun, T., Edgar, D., Liu, C., Tsabedze, T., Yim, M.: PaintPots: Low cost, accurate, highly customizable potentiometers for position sensing. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp. 1212–1218 (May 2017)
37. Yim, M., Duff, D.G., Roufas, K.D.: PolyBot: a modular reconfigurable robot. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). vol. 1, pp. 514–520 vol.1 (April 2000)
38. Yim, M., Shen, W., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robotics Automation Magazine **14**(1), 43–52 (March 2007)

39. Yim, M., White, P., Park, M., Sastra, J.: Modular self-reconfigurable robots. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 5618–5631. Springer New York, New York, NY (2009)
40. Zhao, Y., Lin, H., Tomizuka, M.: Efficient trajectory optimization for robot motion planning. In: 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). pp. 260–265 (Nov 2018)
41. Zhu, W., Lamarche, T., Dupuis, E., Jameux, D., Barnard, P., Liu, G.: Precision control of modular robot manipulators: The VDC approach with embedded FPGA. IEEE Transactions on Robotics **29**(5), 1162–1179 (Oct 2013)
42. Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: CHOMP: Covariant hamiltonian optimization for motion planning. The International Journal of Robotics Research **32**(9-10), 1164–1193 (2013)