2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

Contact Mode Guided Motion Planning for Quasidynamic Dexterous Manipulation in 3D

Xianyi Cheng, Eric Huang, Yifan

***Abstract*— This paper presents Contact Mode Guided Manipulation Planning (CMGMP) for 3D quasistatic and quasidynamic rigid body motion planning in dexterous manipulation. The CMGMP algorithm generates hybrid motion plans including both continuous state transitions and discrete contact mode switches, without the need for pre-specified contact sequences or pre-designed motion primitives. The key idea is to use automatically enumerated contact modes of environment-object contacts to guide the tree expansions during the search. Contact modes automatically synthesize manipulation primitives, while the sampling-based planning framework sequences those primitives into a coherent plan. We test our algorithm on fourteen 3D manipulation tasks, and validate our models by executing some plans open-loop on a real robot-manipulator system[1].**

I. Introduction

Dexterous manipulation planning is challenging in many aspects. The first challenge is the exploitation of dexterity. The dexterity in manipulation can come from the dexterity of robot hands (intrinsic dexterity) and the exploitation of the environments (extrinsic dexterity) [1]. A general dexterous manipulation planner needs to use both intrinsic and extrinsic dexterity cleverly. The second challenge is the contact-rich nature of manipulation. Making and breaking contacts bring more complexity into the system by changing its kinematics and dynamics. This hybrid nature makes planning through contacts difficult. The CMGMP aims to move one step closer towards general dexterous manipulation planning by considering the dexterity from both the robot hand and the environment, and planning among these contacts.

The CMGMP algorithm is a Rapidly-Exploring Random Tree (RRT) [2] based planner with automatically enumerated contact modes to guide tree extensions. The key idea is to use contact modes. Contact modes are helpful both in generating continuous motions and capturing discrete changes. Contact modes serve like automatically generated "motion primitives", guiding the planning of continuous motions in submanifolds of the configuration space. Compared with manually designed motion primitives, contact modes automatically generate more varied motions while requiring less engineering effort. In the discrete space, contact modes help find paths across manifolds. The set of kinematically feasible contact modes capture all possible discrete contact changes in the current system configuration because it enumerates all possible transitions of the contact states. Combined with sampling-based planning, contact modes boost the exploration over large continuous and discrete spaces for contact-rich motions.

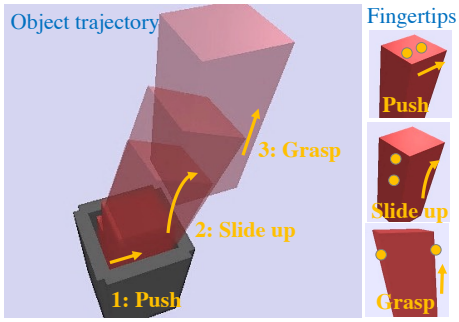


Fig. 1: The goal is to get the object out of the box while the gaps are too small for both fingers to get in. Our method plans first to push the object on the top to create a wider gap that lets the fingers slide up the object from the side.

This paper presents the complete version of the CMGMP algorithm that can solve 3D quasistatic and quasidynamic dexterous manipulation tasks with a rigid object in a rigid environment, assuming non-sliding finger contacts, compatible with user-provided 3D object mesh models and robot-manipulator kinematic models. An example for peg-out-of-hole planning is shown in Figure 1. The planner generates the strategy of first pushing the object on the top to create a wider gap that lets the fingers slide up the object from the side, and then form a grasp. Compared to our 2D quasistatic version [3], this work demonstrates that this type of algorithms can solve a much more general range of manipulation tasks. To the best of our knowledge, the CMGMP is the first method capable of solving diverse dexterous manipulation tasks of such levels of complexity without any pre-designed skill or pre-specified modes. We believe the flexibility, simplicity, and general applicability demonstrated by this work bring us one step closer towards general dexterous robotic manipulation.

II. Related Work

A. Manipulation Planning

Efficient search and optimization algorithms have been developed to solve motion sequencing/planning problems [4], [5]. Sampling-based planning methods like CBiRRT [6] and IMACS [7] explore the manifolds of known constraints. Most of these methods require predefined states or primitives. The solutions are also confined to be the combinatorics of

*This work was supported under NSF Grant IIS-1909021.

The authors are with Carnegie Mellon University, Pittsburgh, PA, 15213, USA. <xianyic, erichl, yifanh>@andrew.cmu.edu, mattmason@cmu.edu

[1] The video is available at https://youtu.be/JuLlliG3vGc

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9811872

predefined states/primitives. As existing taxonomies cannot even categorize all human grasping behaviors [8], it is therefore impractical to develop a skill library that fulfills the complexity and dexterity needed in general manipulation. In contrast, our work uses contact modes to automatically generate "motion primitives", allowing the planner to find a variety of lower-dimensional solutions.

Contact formations [9] have been explored in [10], [11], [12], [13] to plan motions between two rigid bodies [12] and within a robot hand [10], [14]. Contact information helps to decompose the search space into smaller chunks. Search and planning within contact formations are later combined into a complete solution. These methods all require to precompute all possible contact formations offline.

Our previous work is a simplified version of the CMGMP [3]. The addressed problems are limited to 2D domains. The quasistatic assumption prevents it from generating simple actions like dropping or toppling. This work presents a more complete version of the CMGMP by extending from 2D domains to 3D domains, from quasistatic to quasidynamic, from well-parameterized 2D object geometries to arbitray 3D object models, and from simple free-moving point manipulator model to more realistic user-defined robot kinematics and non-sliding finger contact models.

B. Contact-rich Motion Generation

Trajectory optimization is effective in generating contact-rich motion plans. Contact-Invariant Optimization methods [15], [16] produce complex whole-body and manipulation behaviors in simulation, assuming soft contacts which may violate physics laws. Dynamic manipulation planning for rigid bodies is explored in [17], [18], [19], [20], generating simple manipulation actions with small numbers of contact transitions, like pushing, pivoting, and grasping. Chen et al. [21] interleaves tree search and trajectory optimization to efficiently solve a 2D in-hand manipulation problem. However, contact implicit trajectory optimization (CITO) methods [17] could be intractable without good initialization. To our best knowledge, there has not been any trajectory optimization method that can solve the tasks of similar level of complexities shown here.

III. Problem Description

A. Inputs and Outputs

The inputs to our method are the start and goal poses of the object, the geometries, and the properties of the object and the environment:

1) **Object start pose**: $q_{\text{start}} \in SE(3)$.
2) **Object goal region**: $Q_{\text{goal}} \subset SE(3)$.
3) **Object properties**: a rigid body $\mathcal{O}$ with known geometry, mass distribution, and friction coefficients with environment μ_{env} and with the manipulator μ_{mnp}.
4) **Environment**: $\mathcal{E}$ with known geometries.
5) **Manipulator model**: a user-defined model for the robot-manipulator system. The manipulator can make at most N_{mnp} contacts with the object.

Our method outputs a trajectory π that is a sequence of object motions, contacts, and contact modes. At step t, the trajectory $\pi(t)$ gives:

1) **Object motion**: object configuration $q(t)$ at step t.
2) **Environment contacts**: the contact points of the object with the environment $N_{\text{env}}(t)$. The kth environment contact $c_k^{\text{env}}(t)$ is specified by its contact location $p_k^{\text{env}}(t)$ and contact normal $n_k^{\text{env}}(t)$.
3) **Manipulator configurations**: the manipulator configuration $q^{\text{mnp}}(t)$. The manipulator contacts with the object can be obtained from $q^{\text{mnp}}(t)$ using forward kinematics: $c^{\text{mnp}}(t) = [c_1^{\text{mnp}}(t), c_2^{\text{mnp}}(t), \ldots, c_{N_{\text{mnp}}}^{\text{mnp}}(t)]$. The kth manipulator contact is specified by its contact location $p_k^{\text{mnp}}(t)$ and contact normal $n_k^{\text{mnp}}(t)$.
4) **Contact mode**: the 3D contact mode $m(t)$ of the environment contacts and the manipulator contacts.

B. Assumptions

In addition to the standard assumptions made within rigid body simulators (non-penetration, point contacts, polyhedral friction cones [22], [23], etc.), we assume the following: (1) The users can choose to enforce either quasistatic or quasidynamic assumptions. For quasistatic manipulation, inertial forces are negligible, and the object needs to be in force balance all the time. For quasidynamic manipulation, the tasks involve occasional brief dynamic periods. The accelerations do not integrate into significant velocities. Momentum and restitution of impact are negligible [24]. In our experience, quasistatic [25] and quasidynamic assumptions hold even during fairly fast manipulator velocities. Moreover, manipulation motions synthesized in a quasistatic model are inherently safer. (2) Environment-object contacts can have any modes, but manipulator-object contacts are sticking only. We currently does not plan finger sliding motions.

IV. Contact Mode Based Manipulation Models

This section describes the force and motion models under contact modes with quasistatic and quasidynamic assumptions.

A. Contact Modes in 3D

A contact mode describes the relative contact velocities for all the contacts in a system [24]. A contact mode $m = [m_{\text{cs}}, m_{\text{ss}}]$ in 3D consists of two parts: the contacting/separating (CS) mode m_{cs} and the sticking/sliding (SS) mode m_{ss}. The CS mode m_{cs} is the sign of the contact normal velocities $v_{c,n}$. For a system with N contacts, we have $m_{\text{cs}} = [\textit{sign}(v_{c,n}^i)] \in \{0, +\}^N$, where $v_{c,n}^i$ is the normal velocity for the ith contact in its own contact frame. The SS mode m_{ss} of a CS mode identifies the directions of contact tangent velocities $v_{c,t} \in \mathbf{R}^2$. The contact tangent planes are first being divided by n_t equal angled hyperplanes $C_T = [C_T^1, \ldots, C_T^{n_t}]^T$, we have $m_{\text{ss}} = [\textit{sign}(C_T \cdot v_{c,t}^i)] \in \{\{-, 0, +\}^{n_t}\}^N$.

If we approximate unilateral contacts by linear complementarity constraints, each contact mode corresponds to a facet of the complementary cone [26]. Individually solving

for each contact mode divides the complementarity constraint into easier sub-problems. Our previous work [27] shows that the complexity of enumerating all 3D contact modes for one object is $\mathcal{O}(N^d)$, where N is the number of contacts and d is the effective degrees of freedom of the object. For example, d is 6 for a free cube, 3 for a cube between two tight walls, 1 for a cube inside a tight square pipe. The number of contact modes for one object is usually 50 to 400, which makes 3D contact mode enumeration practical in our algorithm.

B. 3D Contact Mode Constraints on Velocities and Forces

If we adopt the polyhedral approximation of friction cones [28], given a contact mode $m \in \mathbf{M}$, we obtain linear constraints about the object motion and contact forces.

Velocity Constraints Contact velocities for all the contacts in the contact frames can be written as:

$$v_c = G^T v^o - \begin{bmatrix} J\dot{q} \\ 0 \end{bmatrix} \tag{1}$$

where v^o is the object body velocity in the twist form; G is the contact grasp map [29]; $\dot{q}$ is the manipulator joint velocity and J is the manipulator's Jacobians; the 0 part is for environment contacts since they are fixed.

For the ith contact, its CS mode $m^i_{\text{cs}} \in \{+, 0\}$ constrains the contact normal velocity $v^i_{c,n}$:

$$\begin{cases} v^i_{c,n} > 0 & \text{if} \quad m^i_{\text{cs}} = + \\ v^i_{c,n} = 0 & \text{if} \quad m^i_{\text{cs}} = 0 \end{cases} \tag{2}$$

The SS mode $m^i_{\text{ss}} \in \{-, 0, +\}^{n_t}$ constrains the contact tangent velocity $v^i_{c,t}$:

$$\begin{cases} C^j_T \cdot v^i_{c,t} > 0 & \text{if} \quad m^{i,j}_{\text{ss}} = + \\ C^j_T \cdot v^i_{c,t} = 0 & \text{if} \quad m^{i,j}_{\text{ss}} = 0 \\ C^j_T \cdot v^i_{c,t} < 0 & \text{if} \quad m^{i,j}_{\text{ss}} = - \end{cases} \tag{3}$$

where C_T contains n_t vectors that partition the contact tangent plane. If $n_t = 2$ (for a 4-sided polyhedral friction cone), we have $C^1_T = [1, 0]$ and $C^2_T = [0, 1]$.

Force Constraints

Let the magnitudes of contact force of the ith contact be $\lambda^i = [\lambda^i_{t_1}, \lambda^i_{t_2}, \lambda^i_n]$, for two contact tangent directions and the contact normal direction respectively.

The CS mode decides whether there exists contact forces:

$$\begin{cases} \lambda^i = 0 & \text{if} \quad m^i_{\text{cs}} = + \\ \lambda^i_n > 0 & \text{if} \quad m^i_{\text{cs}} = 0 \end{cases} \tag{4}$$

If m^i_{cs} is 0, there exists contact forces. There are two different conditions: the contact is sticking ($\forall j, m^{i,j}_{\text{ss}} = 0$), and sliding ($\exists j, m^{i,j}_{\text{ss}} \neq 0$).

When the contact is sticking, $\lambda^i_{t_1}$ and $\lambda^i_{t_2}$ are the contact tangent forces in x and y axis. The contact force should be in the polyhedral friction cone of the friction coefficient μ:

$$\begin{bmatrix} C_T & \mu \\ -C_T & \mu \end{bmatrix} \cdot \lambda^i > 0 \tag{5}$$

If the contact is sliding, due to the maximum dissipation law, the tangent force should be in the opposite direction of the sliding velocity. Let $\{h^{i,j}\}$ to be the edge(s) of the 1D/2D contact tangent velocity cone of m^i_{ss}, $\lambda^i_{t_1}$ and(or) $\lambda^i_{t_2}$ are contact force magnitudes in the $\{h^{i,j}\}$ direction(s). In our polyhedral approximation, the contact tangent force f^i_t should be in the opposite cone of the contact sliding velocity cone:

$$f^i_t = -\sum_j \lambda^i_{t_j} h^{i,j}, \lambda^i_{t_j} > 0 \tag{6}$$

where the edge(s) $\{h^{i,j}\}$ for all m^i_{ss} can be precomputed by C_T. From the Coulomb friction law, we have:

$$\begin{gathered} \lambda^i_{t_j} > 0, \forall j \\ \mu\lambda^i_n - \sum_{j=1}^{K} \lambda^i_{t_j} = 0 \end{gathered} \tag{7}$$

Putting the velocity and force constraints for all contacts together (Equation 1,2,3,4,5,7), letting λ be the magnitudes of all contact force directions, we get a set of linear equations and inequalities from contact mode constraints:

$$A_{\text{ineq}} \begin{bmatrix} v & \lambda \end{bmatrix}^T > b_{\text{ineq}}, A_{\text{eq}} \begin{bmatrix} v & \lambda \end{bmatrix}^T = b_{\text{eq}} \tag{8}$$

C. Quasistatic Assumption

In quasistatic manipulation, the object should always be in a force balance. The force balance equation is written as:

$$\begin{bmatrix} G_1 h_1, G_2 h_2, \ldots \end{bmatrix} \cdot \begin{bmatrix} \lambda_1, \lambda_2, \ldots \end{bmatrix}^T + F_{\text{external}} = 0 \tag{9}$$

where $\begin{bmatrix} \lambda_1, \lambda_2, \ldots \end{bmatrix}^T$ are the magnitudes of forces along active contact force directions $\begin{bmatrix} h_1, h_2, \ldots \end{bmatrix}^T$ determined by contact modes as described in Section IV-B. $\begin{bmatrix} G_1, G_2, \ldots \end{bmatrix}^T$ are the contact grasp maps. F_{external} includes other forces on the object, such as gravity and other applied forces.

D. Quasidynamic Assumption

Quasidynamic assumption relaxes the requirement for object being in force balance, allowing short periods of dynamic motions. We assume accelerations do not integrate into significant velocities. In numerical integration, the object velocity from the previous timestep is 0. The equations of motions become:

$$M_o \dot{v^o} = \begin{bmatrix} G_1 h_1, G_2 h_2, \ldots \end{bmatrix} \cdot \begin{bmatrix} \lambda_1, \lambda_2, \ldots \end{bmatrix}^T + F_{\text{external}} \tag{10}$$

In discrete time, the object acceleration $\dot{v^o}$ can be written as $\frac{v^o}{h}$, where h is the step size.

E. Solve for Desired Motions at Every Timestep

In Section V-C, we use numerical integration to compute the object trajectories. Here we describe how we find the object motion at each timestep using the force and motion models derived above. We solve a quadratic program to obtain the optimal motion given a desired velocity v^o_{des}:

$$\begin{aligned} &\min_{v^o, \dot{q}, \lambda,} \ \|v^o_{\text{des}} - v^o\|^2_2 + \epsilon \lambda^T \lambda \\ &\text{s.t.} \quad \text{Equation 8 (contact mode constraints)} \\ &\qquad \text{Equation 9 (quasistatic)/Equation 10 (quasidynamic)} \end{aligned} \tag{11}$$

Algorithm 1 the CMGMP algorithm

Input: q_{start}, q_{goal}
Output: tree $\mathcal{T}$

$\mathcal{T}$.add-node(q_{start})
while (*Time limit has not been reached*) **do**
  $q_{\text{rand}} \leftarrow$ SAMPLE-OBJECT-CONFIG(q_{goal})
  $q_{\text{near}} \leftarrow$ NEAREST-NEIGHBOR($\mathcal{T}, q_{\text{rand}}$)
  $c^{\text{env}} \leftarrow$ COLLISION-DETECTION(q_{near})
  $\mathbf{M}_{\text{cs}} \leftarrow$ CS-MODE-ENUMERATION(c^{env})
  for $m_{\text{cs}} \in \mathbf{M}_{\text{cs}}$ **do**
    // Iterate all CS modes
    EXTEND($m_{\text{cs}}, q_{\text{near}}, q_{\text{rand}}, c^{\text{env}}$)
return $\mathcal{T}$

where $\epsilon\lambda^T\lambda$ is a regularization term on the contact forces.

V. The CMGMP Algorithm

A. Planning Framework Overview

Algorithm 1 presents our planning framework. The function SAMPLE-OBJECT-CONFIG samples an object configuration q_{rand}, with a user-defined possibility p of being a random sample and $1-p$ of being q_{goal}. For q_{rand}, its nearest neighbor q_{near} in the tree $\mathcal{T}$ is found through a weighted $SE(3)$ metric:

$$\mathrm{d}(q_1, q_2) = \mathrm{d}_{\text{trans}}(q_1, q_2) + w_r \cdot \mathrm{d}_{\text{angle}}(q_1, q_2) \tag{12}$$

where w_r is the weight that indicates the importance of rotation in the system, and d_{trans} is the translation and d_{angle} measures the rotation angle between two configurations. Collision checking is performed to obtain environment contacts c^{env}. The function CS-MODE-ENUMERATION enumerates all feasible CS modes $\mathbf{M}_{\text{cs}}$ for c^{env}. Under each CS mode $m_{\text{cs}} \in \mathbf{M}_{\text{cs}}$, the function EXTEND expands the tree from q_{near} towards q_{rand} for a user-defined maximum distance.

The EXTEND function has three major steps: (1) BEST-SS-MODE: chooses a best SS mode given the desired object motion; (2) RELOCATE-MANIPULATOR: tries to relocate fingertip contacts when necessary; (3) PROJECT-INTEGRATE: generates motions that move towards q_{rand} under a contact mode. A successful extension adds a new node q_{new} and a new edge to $\mathcal{T}$. The following subsections include more details of these subcomponents in EXTEND.

B. Extend for Every CS Mode and Filter SS Modes

A CS mode indicates a transition between two contact states. For every q_{near}, we extend all its CS modes, so that the planner can explore all the next contact states from the current contact state. For each CS mode, we don't need to explore all its SS modes. The SS modes are approximations to the infinite number of sliding directions and polyhedral friction cones. We only need to select the SS mode in the best approximation cone for a desired object motion. In addition, to find solutions that lie in the lowest-dimensional space for a CS mode, such as an object pivoting around two sticking contacts, we also extend the all-sticking SS mode for each CS mode.

Algorithm 2 EXTEND function

Input: m_{cs}, q_{near}, q_{rand}, c^{mnp}, c^{env}

$m_{\text{ss}} \leftarrow$ BEST-SS-MODE($q_{\text{near}}, q_{\text{rand}}, c^{\text{mnp}}, c^{\text{env}}$)
$m \leftarrow$ FULL-MODE($[m_{\text{cs}}, m_{\text{ss}}]$)
$q^{\text{mnp}} \leftarrow$ PREVIOUS-MANIPULATOR-CONFIG(q_{near})
if not MOTION-FEASIBLE($m, q_{\text{near}}, q^{\text{mnp}}, c^{\text{env}}$) ...
or MANIPULATOR-FEASIBLE(q^{mnp}) **then**
  $q^{\text{mnp}}_{\text{new}} \leftarrow$ RELOCATE-MANIPULATOR($q_{\text{near}}, q^{\text{mnp}}, m$)
else
  $q^{\text{mnp}}_{\text{new}} \leftarrow q^{\text{mnp}}$
$q_{\text{new}} \leftarrow$ PROJECT-INTEGRATE($q_{\text{near}}, q_{\text{rand}}, q^{\text{mnp}}_{\text{new}}, m$)
if $q_{\text{new}} \neq q_{\text{near}}$ **then**
  ASSIGN-FINGER-CONTACT-TO-NODE($q_{\text{new}}, q^{\text{mnp}}_{\text{new}}$)
  $\mathcal{T}$.add-node(q_{new})
  $\mathcal{T}$.add-edge($q_{\text{near}}, q_{\text{new}}$)
return

We select the best SS mode that has the smallest cost. In the function BEST-SS-MODE, for every SS mode, a quadratic programming is solved to get the closest velocity to the goal velocity under the contact mode velocity constraints (Equation 11). In practice, for the cost function in Equation 11, we assign a weight for angular velocity, often the same as w_a in the SE(3) metric in Equation 12, so that the cost function becomes:

$$cost = \|{v^o_{\text{des}}}^{(v)} - v^{o(v)}\|_2^2 + w_a\|{v^o_{\text{des}}}^{(\omega)} - v^{o(\omega)}\|_2^2 \tag{13}$$

where $v^{o(v)}$ is the translational velocity and $v^{o(\omega)}$ is the angular velocity.

C. Projected Integration

Starting from an object configuration q_{near}, all reachable object configurations under the constraints by a contact mode m form a manifold with boundary in the object configuration space $\mathcal{M}_m$. This projected integration process finds the furthest configuration q_{new} that the object could reach from q_{near} towards q_{rand} on $\mathcal{M}_m$.

At time-step k, we first update the desired object velocity v^o_{des} as the body velocity between q_k and q_{rand} in the twist form. Then, Equation 11 obtains v^o_k by substantially projecting v^o_{des} onto the tangent space at q_k of $\mathcal{M}_m$. We integrate v^o_k by the first order forward Euler method:

$$q_{k+1} = Tr(q_k, hv^o_k) \tag{14}$$

where h is the size of the time-steps, Tr is the rigid body transformation computed from the body velocity hv^o_k, applied on q_k [29]. The environment contacts in the constraints of Equation 11 are updated every time-step by collision checking.

The projected integration stops when: (1) v^o_{des} is zero: q_k is the closest configuration on $\mathcal{M}_m$ to q_{rand}. (2) No feasible v^o_k: moving towards q_{rand} is not feasible with current contact mode under the quasistatic or quasidynamic assumption. (3) The robot collides with the environment, or new object-environment contacts are made. (4) No solution exists for the robot inverse kinematics (IK). (5) User-defined maximum translation or rotation for a step has been reached.

To address the constraint drift problem, we project the object configuration back to the contact manifold every several time-steps through a correction velocity v_{cor}:

$$\min_{v_{\text{cor}}} \|G_N v_{\text{cor}} - d^c\| + \epsilon \|v_{\text{cor}}\| \tag{15}$$

we have $G_N = \left[C_N G_1^T, \ldots, C_N G_i^T, \ldots\right]^T$ where $C_N = [0, 0, 1, 0, 0, 0]$ and G_i is the grasp matrix of the ith contact. $d^c = [d_1^c, \ldots d_i^c]^T$ are the contact distances. The solution is $v_{\text{cor}} = (G_N^T G_N + \epsilon I)^{-1} G_N^T d^c$.

D. Planning Finger Locations

The function RELOCATE-MANIPULATOR explores new finger contacts and robot configurations when: (1) the initial robot configuration is not assigned; (2) the current robot configuration will be in a collision or out of workspace; (3) current finger contacts no longer provides desired motions; (4) being randomly sampled to.

This function is based on rejection sampling. We pre-compute a set of evenly distributed finger placements based on the object mesh model using supervoxel clustering [30]. In practice, we store about 200 finger placements per object. During the planning, we first randomly select some fingers to relocate to new contact locations on the object. The new contacts should provide the desired motion under the current contact mode. The transition is feasible if the remaining finger contacts and new contacts from previously unlocated fingers can keep the object in force balance (quasistatic and quasidynamic) or can still provide the desired motion (quasidynamic). Next, we solve the robot IK. If there is a reachable collision-free robot configuration (a robot-specific finger relocation sub-planner is required), the new sample is accepted. Otherwise, the algorithm repeats this process until it gets a feasible sample or reaches the iteration limit.

VI. RESULTS

A. Planning Results for Simulated Manipulation Tasks

We test our planner on manipulation tasks that vary in object shapes, environments, model assumptions, manipulator types, etc. All the tasks need dexterous maneuvers due to specific task constraints. Table I gives brief descriptions on 14 representative simulated tasks. Figure 2 shows the real-life scenarios for Task 1-4. The supplementary video[1] and Figure 4 visualize some planning results.

Task 1 (peg-out-of-hole), as explained in Section I, is commonly encountered as unpacking in our daily life. Task 2 (bookshelf) requires the book to be first slid/pulled out to form a grasp. We filmed a human participant taking a book with all the fingers and with only two fingers. The strategies by our planner are surprisingly similar to human strategies. In Task 3 (pick up a blade), our planner consistently generates a strategy of sliding the blade to the edge of the table to expose its bottom for grasping, similar to the slide-to-edge grasp in [31]. Task 4 (take a bottle) is inspired by a human grasping behavior, "simultaneous levering out and grasp formation", observed in [8] when a human took a bottle of drink from the fridge. Our planner finds similar strategies: the manipulator

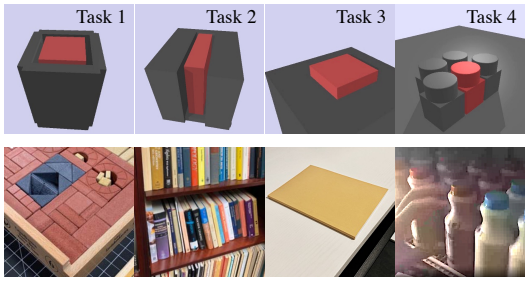


Fig. 2: Task 1-4. The lower row shows the corresponding real-life scenarios. Task 1: get the first block out of packed blocks. Task 2: take a book from a bookshelf. Task 3: pick up a very thin book from the table. Task 4: grab a bottle from the fridge full of bottles (the photo is from [8]).

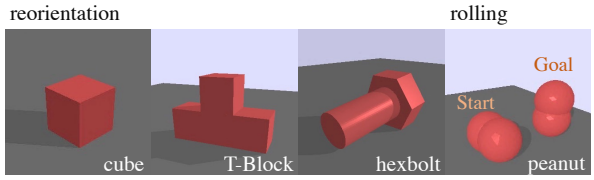


Fig. 3: Test objects for object reorientation (Task 5-7) and rolling (Task 8).

first pivots or pulls out the object to create more space to form a three-finger grasp. All the strategies generated by our planner for Task 1-4 leverage environmental contacts; they may be seen as constraint-exploiting grasps [31] commonly observed in human grasping behaviors. Our planner also generates strategies to reorient and to roll objects of different shapes in Figure 3. Task 8-9 demonstrate that this planner can plan over many contact changes, although in practice it should not be directly given such long horizon tasks. Task 11 - 14 exhibit the quasidynamic strategies that exploit the gravity of the objects. For example, the robot can move the fingers away and let the object drop. Another example is Task 13 inspired by the flip-and-pinch strategy from [32].

B. Implementation and Planning Statistics

We implement the algorithm using C++ and Dart [33], with Bullet [23] for collision detection. The code is available at `https://github.com/xianyicheng/cmgmp`. Table II shows the planning statistics provided by a computer with the Intel Core i9-10900K 3.70GHz CPU. Our planner generates most contact-rich plans in several seconds. The number of nodes in the tree is small, indicating that the exploration is efficient.

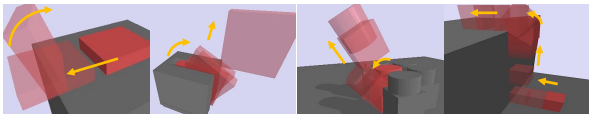
To set up new tasks, the users need to adjust some parameters to reasonable ranges according to the tasks (no need for careful tuning): (1) w_a for the distance metrics used in Equation 12 and 13, which should be the product of the object characteristic length and the estimated importance of rotations. (2) the maximum translational and rotational distance for EXTEND to move towards q_{rand}.

#	Task Name	Assumption	Robot	Description
1	Peg-out-of-hole	Quasistatic	3 free-moving balls	The small gaps prevent a direct pickup
2	Bookshelf	Quasistatic	3 free-moving ball a parallel jaw gripper the DDHand	Take a book from a bookshelf
3	Pick up a card	Quasistatic	3 free-moving ball the DDHand	The object is too thin or too wide to grasp
4	Grab a bottle	Quasistatic	3 free-moving balls	The bottle is surrounded by other bottles
5-7	Object Reorientation	Quasidynamic	2 free-moving balls	Cube (Task 5), T-block (Task 6), Hexbolt (Task 7)
8	Rolling	Quasistatic	2 free-moving balls	Reorient an object with a smooth surface
9	Cube-and-Wall	Quasistatic	One free-moving ball	The object is too heavy to pick up
10	Cube-and-Stairs	Quasistatic	2 free-moving ball	The object is too heavy to pick up
11	Regrasp a Cube	Quasidynamic	A parallel jaw gripper	Use a parallel jaw gripper to reorient an object
12	Placedown	Quasidynamic	2 free-moving balls	Place down a thin object in a grasp
13	Flip-and-Pinch	Quasidynamic	2 free-moving ball the DDHand	Flip then pinch a thin object on the table with two fingers
14	Object Reorientation	Quadidynamic	A robot arm + a rod finger	Reorient a cube considering robot kinematics and collision

TABLE I: Brief description of some selected test tasks.

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Success	10/10	10/10	10/10	10/10	30/30	27/30	29/30	10/10	9/10	6/10	30/30	10/10	10/10	10/10
Time (second)	5.4±1.1	2.7±1.3	3.9±1.0	1.7±0.8	4.8±2.8	11±10	2.8±1.7	0.6±0.3	8.3±3.9	46±11	14±9.0	12±7.0	9.0±2.8	23±14
Nodes in Solution	9±2	7±1	8±2	6±1	7±3	11±6	15±8	7±3	12±8	42±12	6±2	12±4	12±5	4±0
Nodes in Tree	55±16	28±10	38±16	27±20	52±23	78±62	77±38	18±8	82±65	437±95	41±20	68±27	75±26	34±19

TABLE II: Planning statistics. A run is successful if a solution can be found within 100 iterations (200 for Task 10). "Time", "Nodes in Solution" and "Nodes in Tree" are in the format of "mean"±"standard deviation" for successful runs.



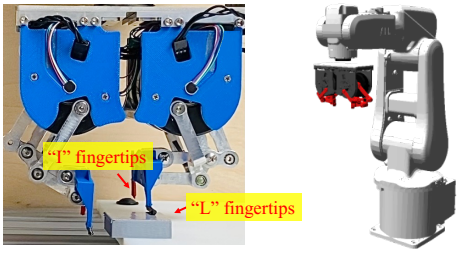


Fig. 5: Left: The Dexterous DDHand executing Task 3. Right: the hand-robot system.

C. Real Robot Experiments

To validate our models, we planned for Task 2 (bookshelf), Task 3 (pickup a blade), and Task 13 (flip-and-pinch) on a Dexterous Direct Drive Hand (DDHand) [34] mounted on an ABB IRB-120 robot arm, as shown in Figure 5. The hand has two fingers, each with two degrees of freedom. There are two types of fingertips, the "L" type for Task 3 and 13, and the "I" type for Task 2. We provide the planner with the hand IK model and contact models for the fingertips. An "L" finger has a line contact model approximated by two point contacts. An "I" finger has a patch contact model, approximated by three point contacts. In the supplementary video, we run the planned robot trajectories only with robot position control. Although some runs are successful, the system is sensitive to uncertainties like object initial position errors. Robust executions will need controllers with force control [35] and force and vision feedbacks.

VII. Conclusion and Discussion

We present the CMGMP framework that uses contact mode as guidance to generate quasistatic and quasidynamic dexterous manipulation motions in 3D. The strategies by our planner effectively leverage the environment as an external source for manipulation. For more general manipulation planning, we conclude several directions for future work: (1) Planning for fully dynamic manipulation plans will double the dimensions of the search space, which is impractical in our current framework; (2) We assume sticking finger contacts. Local planners for finger rolling/sliding can be considered for further dexterity; (3) The RRT is non-optimal, while the optimal RRT-star is not pratical as it is almost impossible to directly connect two nodes in our problem. Post-processing with trajectory optimization could be used to enhance the solution quality. Solutions from our planner can also be used as warm-start for CITO [17]; (4) We observed that some plans are fragile under uncertainty while others are very robust to execute even in an open-loop manner. It is important to have criteria over motion stability [36], [37] to increase the overall solution qualities; (5) It is possible to adapt ideas and techniques of the CMGMP into other constrained sampling-based planning frameworks [7].

References

[1] N. Chavan-Dafle, A. Rodriguez, R. Paolini, B. Tang, S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[2] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[3] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d," *IEEE International Conference on Robotics and Automation*, 2021.

[4] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 3684–3691.

[5] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning." in *Robotics: Science and Systems*, vol. 2, 2018.

[6] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 625–632.

[7] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[8] Y. C. Nakamura, D. M. Troniak, A. Rodriguez, M. T. Mason, and N. S. Pollard, "The complexities of grasping in the wild," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 233–240.

[9] J. Xiao and X. Ji, "Automatic generation of high-level contact state space," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 584–606, 2001.

[10] J. C. Trinkle and J. J. Hunter, "A framework for planning dexterous manipulation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 1991, pp. 1245–1251 vol.2.

[11] X. Ji and J. Xiao, "Planning motions compliant to complex contact states," *The International Journal of Robotics Research*, vol. 20, no. 6, pp. 446–465, 2001.

[12] P. Tang and J. Xiao, "Automatic generation of high-level contact state space between 3d curved objects," *The International Journal of Robotics Research*, vol. 27, no. 7, pp. 832–854, 2008.

[13] G. Lee, T. Lozano-Pérez, and L. P. Kaelbling, "Hierarchical planning for multi-contact non-prehensile manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 264–271.

[14] M. Yashima, Y. Shiina, and H. Yamaguchi, "Randomized manipulation planning for a multi-fingered hand by switching contact modes," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 2689–2694.

[15] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 2012, pp. 137–144.

[16] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[17] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[18] J. Sleiman, J. Carius, R. Grandia, M. Wermelinger, and M. Hutter, "Contact-implicit trajectory optimization for dynamic object manipulation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6814–6821.

[19] F. H. N. Doshi and A. Rodriguez, "Hybrid differential dynamic programming for planar manipulation primitives," in *ICRA*, 2020.

[20] B. Aceituno-Cabezas and A. Rodriguez, "A global quasi-dynamic model for contact-trajectory optimization," in *RSS*, 2020.

[21] C. Chen, P. Culbertson, M. Lepert, M. Schwager, and J. Bohg, "Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8262–8268.

[22] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018. [Online]. Available: https://doi.org/10.21105/joss.00500

[23] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[24] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001.

[25] ——, "On the scope of quasi-static pushing," in *International Symposium on Robotics Research, 1986*, 1986, pp. 229–233.

[26] S. Berard, K. Egan, and J. C. Trinkle, "Contact modes and complementary cones," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 5280–5286.

[27] E. Huang, X. Cheng, and M. T. Mason, "Efficient contact mode enumeration in 3d," in *Workshop on the Algorithmic Foundations of Robotics*, 2020.

[28] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[29] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[30] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation-supervoxels for point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2027–2034.

[31] C. Eppner, R. Deimel, J. Alvarez-Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1021–1038, 2015.

[32] L. U. Odhner, R. R. Ma, and A. M. Dollar, "Open-loop precision grasping with underactuated hands inspired by a human manipulation strategy," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 625–633, 2013.

[33] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[34] A. Bhatia, A. M. Johnson, and M. T. Mason, "Direct drive hands: Force-motion transparency in gripper design," in *Robotics: science and systems*, 2019.

[35] Y. Hou and M. T. Mason, "Robust execution of contact-rich motion plans by hybrid force-velocity control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1933–1939.

[36] Y. Hou, Z. Jia, and M. T. Mason, "Manipulation with shared grasping," 2020.

[37] A. M. Johnson, J. E. King, and S. Srinivasa, "Convergent planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1044–1051, 2016.