Task-specific Manipulator Design and Trajectory Synthesis

Julian Whitman[1] and Howie Choset[2]

***Abstract*—This paper addresses the challenge of determining the optimal design of a customizable robot for a given task. We present a motion planner that not only prescribes a path, but also synthesizes the robot design to follow that path. Our approach treats design variables as part of an inverse kinematics problem to jointly optimize manipulator design parameters and trajectory. We apply our method to two distinct problems where rapid prototyping and customization are desirable: an arm prototyped for a future Mars mission, and a wearable backpack-mounted arm. We then propose a novel method to minimize the number of joints in the mechanism while maintaining its ability to reach workspace task poses. We combine these methods into a framework in which we iteratively optimize and simplify task-specialized manipulator designs.**

***Index Terms*—Cellular and Modular Robots, Kinematics**

I. Introduction

MODULAR robotic systems have the potential to be adapted to varying tasks using a single platform and enable customizable robots to be developed faster and more economically than conventional robots. One challenge in using such robots is determining the optimal design to specialize a robot for a given task. An optimal custom modular manipulator would be more effective and require fewer redundant components than would a generic manipulator, and one could easily reconfigure its assembly to respond to a change in task, supporting rapid redesign and prototyping.

In this work, we introduce a method to jointly plan a path and optimize a manipulator's continuously adjustable design parameters. For high dimensional problems, it is tempting to use entirely stochastic approaches [1], [2], [3], [4]. Instead, we formulate the design and planning problem as a constrained nonlinear program (NLP) that can be solved with standard deterministic quasi-Newton algorithms. Our approach can utilize analytical gradients, resulting in lower computational costs than related stochastic approaches. We optimize over configuration and design variables within the same minimization problem, effectively solving the inverse kinematics (IK) at all task targets jointly with the design. Our objectives and constraints include end-effector position and orientation, joint torques, and obstacle collision avoidance. We apply our method to two distinct problems where rapid prototyping and customization are desirable: an arm prototyped for a potential future Mars mission [5], and a wearable collaborative arm.

State-of-the-art methods which vary the continuous design parameters of the robot [1], [6] assume a fixed number of

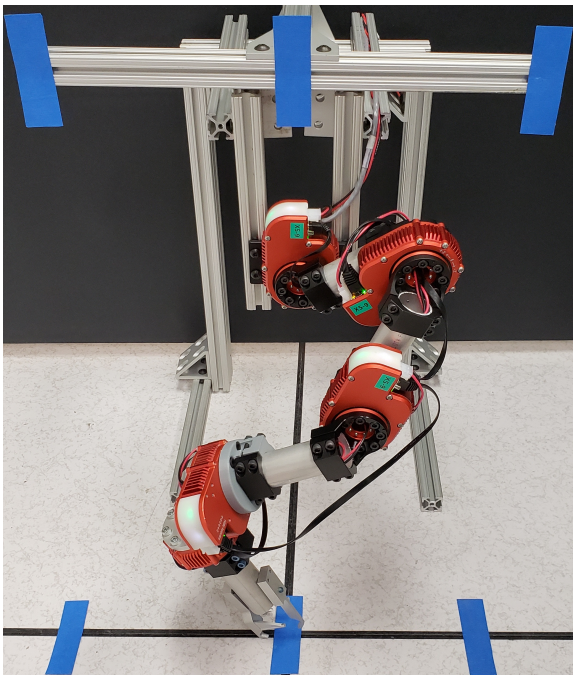

Fig. 1: Rover arm prototyped with modular hardware, reaching one of the target waypoints (every other target marked in blue).

Fig. 2: Our wearable collaborative robot arm reaching an overhead workpiece. The customizable modular arm, attached to a backpack providing computation and power, lends the user a helping hand.

Manuscript received: September 10, 2018; Revised December 15, 2018; Accepted December 21, 2018

This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by NASA Space Technology Research Fellowship NNX16AM81H.

[1]Julian Whitman is with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. jwhitman@cmu.edu

[2]Howie Choset is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. choset@ri.cmu.edu

Digital Object Identifier (DOI): 10.1109/LRA.2018.2890206

joints. We extend our design framework into a novel method to minimize the number of joints in the mechanism, reducing its complexity and mass while maintaining its ability to reach task workspace poses. With these methods, we iteratively optimize and eliminate joints to synthesize a task-specialized, minimized degree-of-freedom, manipulator.

This paper is outlined as follows: Section II reviews related work. Sect. III describes our problem formulation and proposed approach. Sect. IV applies our method to a rover arm prototype and a wearable arm, including hardware validations of each. Sect. V presents our method to reduce the number of joints in an arm. Sect. VI discusses limitations and directions for future work.

II. RELATED WORK

A variety of approaches that optimize robot design and trajectory exist in prior work. Geometric methods, which derive closed-form optimum designs, require substantial assumptions about the robot's workspace, number of joints, and their assembly [7], [8], [9]. Evolutionary algorithms, which randomly vary design parameters while favoring those with high fitness, have been used to create designs with fewer assumptions [2], [3], [4]. However, they are computationally expensive, and their results often vary between runs, even when started from the same initial seed.

A recent approach [1] used a purely stochastic search to synthesize manipulator design parameters and trajectory. They designed a planar manipulator arm and a concentric tube robot using a rapidly-spanning random tree as an inner loop motion planner, and a simulated annealing outer loop on design parameters. Their algorithm is probabilistically complete, and so asymptotically converges to the optimal solution with run time correlated with the size of the end-effector target regions. They considered targets in $\mathbb{R}^3$, without end-effector orientation.

Stochastic methods are computationally burdensome, and do not make use of domain-specific knowledge. For instance, analytical gradients of the robot pose with respect to design and configuration variables are inexpensive to calculate for standard fixed-base rigid-bodied manipulators, and can, for instance, reveal how to modify the robot to escape collisions rather than discarding samples that collide with obstacles. The above methods also have an inner loop for motion planning and an outer loop for design parameter variation, which ignores the relationship between the two within the objective function, i.e., that changing the configuration and design variable simultaneously can decrease objective value. Our formulation, described in Sect. III, uses this observation to optimize both design and configuration variables.

Some existing methods use a matrix containing design parameters and a pseudo-inverse update rule to move through the design space towards a locally optimal design [10], [6]. [6] observed that task constraints implicitly define a manifold of valid robot designs. The authors searched along this manifold to concurrently adjust the design and motion parameters toward optimizing the robot's performance. Their methods iterated only among feasible designs, requiring a user-selected feasible initial design and trajectory, and so cannot initialize from random seeds.

III. PROBLEM FORMULATION

Here we describe a motion planner that simultaneously prescribes a trajectory and synthesizes the manipulator design. We exploit the computational benefits of analytical gradients, and propose a formulation that allows us to use random, often infeasible, initial seeds. To create a motion plan between the task targets, we then present our approach to avoiding obstacle collisions, ensuring smooth motion plans, and lastly task-specific objectives. The main innovations in this section are: 1) posing the combined design and motion planning problem as a nonlinear program that can be input to deterministic solvers, and 2) the formulation of gradients with respect to both configuration and design parameters which allow the solver to smoothly push the design out of obstacles and away from torque limits.

A. Overall optimization problem

We optimize concurrently over the design parameters of any continuously adjustable parts of the robot and the configuration at each waypoint. *Target waypoints* (or "targets") are poses in the workspace that the robot must reach to fulfill the task, and *intermediate waypoints* contain the path the arm moves through between targets. The design and trajectory are optimized for a robot with a fixed set of modules with n joints and m variable design parameters. We denote the design parameters $d \in \mathbb{R}^m$, which can include link lengths, twists about link axes, base locations, or other offsets. The design parameters are properties of the modules chosen; each module may have any number of design variables.

We define the configuration variable matrix $\Theta = [\theta_1, \dots \theta_{n_W}]$, where each $\theta \in \mathbb{R}^n$ is the configuration vector (joint angles) at one of the n_W waypoints. Of these waypoints, n_T will be associated with targets, such that the configurations associated with those waypoints must result in forward kinematics which reach workspace targets. The remainder are intermediate waypoints, for which the end effector does not have a desired target.

The targets are a series of poses in the workspace, which can be specified in terms of a position, orientation, and/or tip axis. The robot will be designed such that its end-effector can reach these targets, in order, without colliding with obstacles. The form of the minimization problem is a weighted sum of n_O objectives with n_C constraints:

$$\begin{aligned} \underset{\Theta, d}{\text{minimize}} \quad & \sum_{i=1}^{n_O} w_i f_i(\Theta, d) \\ \text{subject to} \quad & g_k(\Theta, d) \leq 0, \quad k = 1, \dots, n_C. \end{aligned} \tag{1}$$

The objectives f_i, weights w_i, and constraints g_k are described in the following sections. These functions are inspired by, although not identical to, those in existing multi-objective optimization-based motion planners [11], [12].

B. Reaching the targets

First we formulate inequality constraints which direct the end-effector to reach the workspace targets. Let FK : $\mathbb{R}^n \times \mathbb{R}^m \mapsto \mathrm{SE}(3)$ be the forward kinematic map where $\mathbb{R}^m$ is

the space of design parameters, $\mathbb{R}^n$ the configuration space of the mechanism, and the robot workspace (or task space) is in SE(3), with $[p_{fk}, R_{fk}] = \text{FK}(\theta, d)$. $p_{fk} \in \mathbb{R}^3$ is the position of the end-effector in the world frame, $R_{fk} \in \text{SO}(3)$ is the orientation of the end-effector in the world frame. We further define $\hat{n}_{fk} \in \mathbb{R}^3, |\hat{n}_{fk}| = 1$, as the third column of R_{fk} corresponding to the z-axis of the end-effector.

A position constraint is added for each target waypoint,

$$g_{\text{pos}} = ||p_{T,i} - p_{fk,i}(\theta_i, d)||^2 - \epsilon_{\text{pos}}, \quad i = 1, \ldots, n_T, \quad (2)$$

where there are n_T targets with positions $p_{T,i} \in \mathbb{R}^3$. This constraint will cause n_T of the configuration vectors (n_T of the n_W columns in Θ) to reach their workspace targets. $\epsilon_{\text{pos}} \in \mathbb{R}^+$ is a small position threshold. Each target can also be given an orientation constraint, such as on tip axis,

$$g_{\text{tip}} = 1 - \hat{n}_{fk}(\theta_i, d) \cdot \hat{n}_{T,i} - \epsilon_{tip}, \quad i = 1, \ldots, n_T, \quad (3)$$

where each target has an axis $\hat{n}_{T_i} \in \mathbb{R}^3, |\hat{n}_{T,i}| = 1$ and a small threshold $\epsilon_{\text{tip}} \in \mathbb{R}^+$. When the end-effector tip axis aligns with the target axis, $g_{\text{tip}} = -\epsilon_{\text{tip}} \leq 0$. Orientation can also be fully constrained,

$$g_{\text{rot}} = ||I - R_{fk}(\theta_i, d) R_{T,i}^T||_F - \epsilon_{rot}, \quad i = 1, \ldots, n_T, \quad (4)$$

where each target has an orientation $R_{T,i} \in \text{SO}(3)$ and small threshold $\epsilon_{\text{rot}} \in \mathbb{R}^+$. The end effector orientation is an output of the forward kinematics; we do not include a parameterization of SO(3) in the decision variables.

We also tried posing these distance constraints as costs with high weights (soft constraints). However, we found that in practice this resulted in many undesirable local minima in which the robot did not reach the targets.

C. Obstacle collision avoidance

To avoid collisions with obstacles, we use obstacle penetration penalties based on previous motion planners [12]. Before optimization begins, we discretize the workspace into voxels and precompute a signed distance field, used to define a workspace potential function smoothed around the edges of obstacles, as in [12], and the objective gradients along the trajectory are calculated similarly. At each iteration, the voxels occupied by the robot at each of the waypoints are approximated using ray casting [13] between the input and output of each module. The obstacle avoidance objective f_{obs} is then the sum of the obstacle penetrations of the occupied voxels over all waypoints. We choose this method because it allows gradients to be computed not only for joint angle variables but also for the design parameter vector d. At each waypoint i, we observe that the instantaneous variables θ_i and d can both be varied to effect f_{obs}, and an analytical gradient can be calculated that relates the changes in θ_i and d which result in motion reducing f_{obs}.

Each evaluation calculates the forward kinematics, manipulator Jacobian, collision cost, and collision cost gradient at each occupied voxel. By summing the contributions from d over all waypoints, we can create a gradient $\frac{\partial f_{\text{obs}}}{\partial d}$. The full gradient $[\frac{\partial f_{\text{obs}}}{\partial \Theta}, \frac{\partial f_{\text{obs}}}{\partial d}]$ allows the optimization to efficiently avoid or smoothly escape collisions with obstacles.

D. Waypoint smoothness

We ensure trajectory smoothness by imposing costs on the path length in joint space,

$$f_{\text{joints}} = \sum_{i=1}^{n_W - 1} 1 - \cos(\theta_i - \theta_{i+1}), \quad (5)$$

where the cosine function serves to prevent penalization of joint angles with a 2π difference. This differs from previous joint motion penalties [11], [12] in that it frees the optimizer to use any joint angle value; the final joint angles can be converted to be within $[-\pi, \pi]$ in a post-processing step. If the joints have limits, this penalty could be replaced by a simpler difference cost. f_{joints} helps create paths that do not become contorted in the joint space while attempting to follow a straight path in the workspace.

The path of the end-effector is smoothed by an objective,

$$f_{\text{path}} = \sum_{i=1}^{n_W - 1} ||p_{fk,i} - p_{fk,i+1}||^2, \quad (6)$$

which penalizes distance between subsequent waypoints' end-effector positions.

E. Other objectives and constraints

Some additional penalties and limits were included to ensure feasibility of the solution. The static torque on each joint at each waypoint, τ_i was calculated given the current design and configuration. Torque limits at all waypoints were included based on the maximum continuous torque output of the modules τ_{max},

$$g_{\tau,\text{max}} = \tau_i - \tau_{\text{max}}, \;\; g_{\tau,\text{min}} = -\tau_i - \tau_{\text{max}}, \;\; i = 1, \ldots, n_W. \quad (7)$$

A cost on torques was also added,

$$f_\tau = \sum_{i=1}^{n_W} \tau_i^2. \quad (8)$$

The constraint and cost on torques promote the discovery of designs that hold forces structurally when necessary. Static torque against gravity is a continuous function of design and joint angles, even around singular configurations.

Each design variable corresponding to a link length was marked as a "length variable" and a small penalty f_{length} on these variables was included to prevent unnecessarily long links. This regularization sometimes shortens links until the arm is fully extended to singular configurations. Future work will consider adding a cost to penalize approaching singularities, as in some numerical IK algorithms [14]. Links lengths are also assigned upper and lower bounds based on their corresponding hardware.

Our formulation allows for additional constraints and objectives to be added simply when needed for an application. In the example in Sect. IV-A, each target was assigned a tip axis direction constraint, but a second orientation constraint was needed to make the problem physically realistic; the gripper must grasp objects while approaching from the front but not the back of the object. We added a constraint on another axis of the end-effector frame, similar to the constraint in

(3), with a much larger threshold. Additional objectives and constraints, such as end-effector force application, or self-collision avoidance, can be added as long as their weights are tuned to avoid poor conditioning of (1).

F. Solution procedure

All of the above objectives and constraints have analytical gradients which are provided to the optimizer. Since most are sparse, this provides significant speed improvements over approximating gradients with finite differences. However, the cost landscape for this NLP is highly non-convex. As such, some local minima may still intersect with obstacles or have unnecessarily long trajectories between targets. We run the optimization from multiple random initial seeds in parallel, and use the best local minimum solution.

We implemented (1) in MATLAB version R2016A, and solved it with `fmincon`. We compiled inner loop functions into MEX code for speed, but recognize that we may achieve a 1-2 order of magnitude speed increase if the same functions were implemented in C++. In the following results a four-core Intel i5 desktop with 16 GB RAM was used.

IV. APPLICATIONS AND RESULTS

We present the results of two distinct applications: a Mars rover arm prototype and a supernumerary backpack-mounted limb. For each, we explain our motivation for the use of customized modular design, the problem-specific setup, the design optimization results, and validations of the design with HEBI modular actuators [15]. The variable design parameter vector d includes base position, link lengths, and twist offset between joint axes along links. The module sequence and targets were selected from task requirements and engineering experience, then the design parameters and trajectory were synthesized.

In these results we use optimization weights as follows, where each $w_{(\text{label})}$ corresponds to its cost $f_{(\text{label})}$: $w_{\text{path}} = 500$, $w_{\text{joints}} = 200$, tuned to balance direct paths in the workspace and smooth paths in configuration space. $w_{\text{length}} = 1$, a small regularizer to prevent arbitrarily long links. $w_\tau = 20$ as power-saving was not a priority in our applications, and this objective can conflict with path smoothness. The cost due to these objectives depends on the number of intermediate and target waypoints, as they are sums over the discretized path, as well as the number of joints. To prioritize obstacle avoidance as a soft constraint, we set a high value $w_{\text{obs}} = 5000$. These weights were tuned by hand, but we found that the optimizer results were not sensitive to the exact weight value, but rather on weight order of magnitude. We used the constraint tolerances based on acceptable deviation from the target for the task, $\epsilon_{\text{tip}} = 10^{-3}$ and $\epsilon_{\text{pos}} = 5 \times 10^{-4}$. $\tau_{\text{max}} = 9$ N·m, the max continuous output of the modules. We used convergence tolerances OptimalityTolerance = FunctionTolerance = ConstraintTolerance = StepTolerance = 0.01, tuned to match the order of magnitude of the cost, which impact solver time to convergence.

We used voxels with edge length 2 cm in the signed distance field, based on the size of the smallest obstacle edge present.

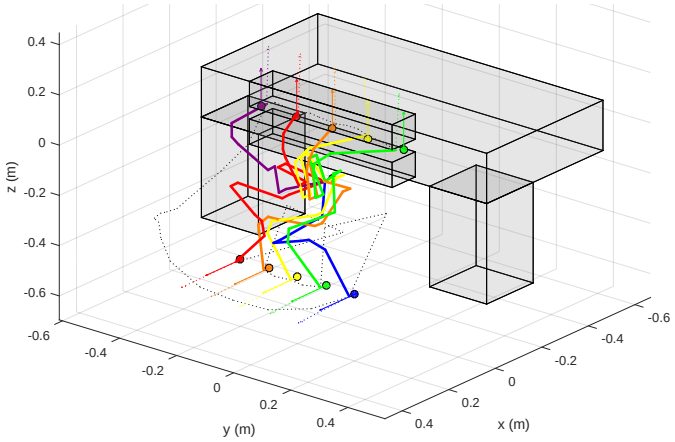

Fig. 3: Rover robotic manipulator design and trajectory synthesized in this work. Each color depicts the arm reaching a correspondingly colored target. We model to rover chassis, wheels, and sample storage area as bounding boxes (grey), and avoid collision with the rover and ground ($z = -0.5$ m). The trajectory between the targets is shown with a dotted black line. The base location is near the origin. The end-effector first reaches the bottom left red target, moves through each target on the ground from left to right, moves up to the top left purple target, then moves through the top row of targets from left to right.

We assigned to each set of n_T targets a trajectory with a total of $4n_T$ waypoints, then redistributed the waypoints such that each path segment between targets had a number of waypoints proportional to the distance between the targets in the workspace. Additional waypoints and higher resolution voxels could be used for a finer path resolution at higher computational cost.

A. Rover arm prototyping

The Mars 2020 robotic rover mission plans to core and collect samples, storing them in sealed capsules and leaving them on the surface of Mars [5]. A future mission may deploy a smaller rover to collect the capsules and deliver them to a launch site. Engineers at NASA Jet Propulsion Laboratory (JPL) are currently prototyping the design of the sample return rover, including its manipulator arm.

We consulted with JPL engineers and learned the approximate layout planned for the arm, which will differ from those of previous missions. Its task will be to grasp sample tubes from the ground, and reach back to place them in a storage rack on the rover chassis. We identified a need for automated design synthesis to assist in rapidly prototyping manipulators, so we applied our design optimization to their current top design concepts.

We approximated the rover body as a collection of rectangular prisms for the front wheels, chassis, and sample storage rack, all of which are treated as obstacles. The ground is also set as an obstacle. Five targets (sample tube locations) were spread on the ground in front of the rover and five in between the two parts of the rack. The targets were assigned tip axes such that the sample capsules would be axially aligned with each axis, and the end-effector would grip the capsules along this axis. In addition to this tip axis constraint, we also require that the grasp approach from $\pm 30^\circ$ from the ground surface

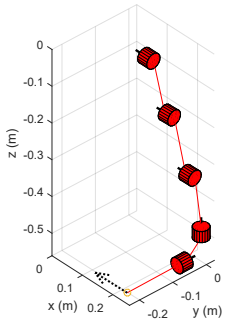

Fig. 4: Rover arm design shown at zero configuration, where red cylinders represent HEBI rotary actuated joint modules, and the dashed arrow shows the end-effector axis of interest. The optimization results in a design with the first three joints spaced nearly evenly, with nearly the same axis directions.

normal so the end-effector could grasp the capsule without colliding with the ground.

We solved (1) with eight random initial points. The modules chosen had a total of nine design parameters and five rotary joints. The mean solution time was 85 minutes. We show the results of the optimization at zero configuration in Fig. 4, its synthesized configurations and design within the workspace in Fig. 3, and its validation in hardware in Fig. 1. The five degree-of-freedom arm reaches ten targets (positions with tip axis constraints). It moves along a trajectory connecting the targets without colliding with obstacles, for which it passes through the narrow channel containing the five targets on the storage rack.

B. Backpack arm

Wearable robots have been designed for specific overhead manufacturing tasks [16], assisting with balance [17], or collaboration with another person via tele-operation [18]. These devices were built for a specific task using engineering expertise and domain knowledge. However, one might expect the task desired for a wearable robot to change frequently. Further, the robot's users might vary in their height or arm span, so one robot may not be sufficient for every variation of the task or every user. We propose to design a customized modular wearable robot for each task and user. The robot can be optimized with fewer actuated joints, resulting in lower mass to be carried, fewer complex and expensive actuators, and lower power consumption.

We optimized the design parameters of a wearable manipulator for an overhead assembly task. In order to affix a workpiece to an overhead plate, the workpiece is held in place by the robot while the user places screws and operates power tools. The task targets were set based on the difference between the height of a fixed overhead plate and the user's height. The targets consist of three positions with tip axes. To avoid collisions with the user and ceiling, a sphere and two rectangular prisms were set as obstacles, shown in Fig. 6. The end-effector is a rubberized plate with spring-loaded hinge.

We solved the NLP (1) with eight random initial seeds. The modules chosen had eight design parameters and three rotary joints. The mean solution time was 1.8 minutes. We show the results of the optimization at zero configuration in Fig. 5, its synthesized configurations and design within the workspace

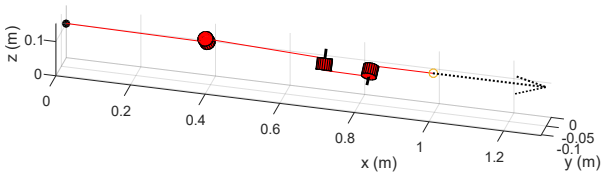

Fig. 5: Wearable arm design shown at zero configuration, where red cylinders represent HEBI rotary joint modules, and the dashed arrow shows the end-effector axis of interest. The optimization result places the joint axes at irregular intervals so that it can reach the targets.

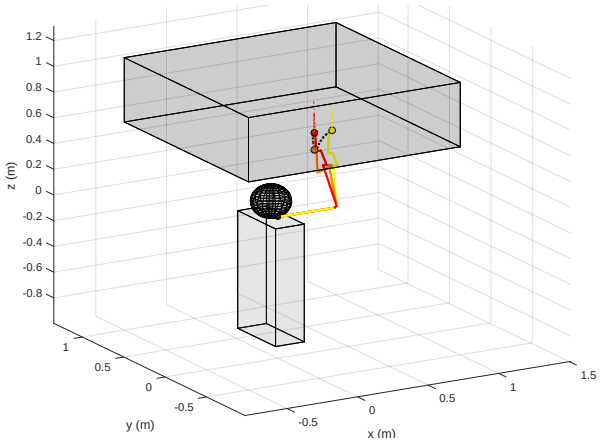

Fig. 6: Wearable arm design, where each color represents the arm reaching its correspondingly colored target. Grey regions show obstacles (the user and the ceiling). The trajectory between targets is shown with a dotted black line.

in Fig. 6, and its validation in hardware in Fig. 2. If we then desire to modify the arm for small changes in the task, like a change in user/ceiling height, this solution can be reused as an initial seed to search for the feasible design closest to the original solution.

C. Sensitivity

To test the sensitivity of the optimization with respect to the workspace targets, we randomly perturbed target locations for the backpack application. We varied each target by a small displacement in a random direction, and re-solved the synthesis using the previous solution as the initial seed, recording the resulting design parameters d_{new}. The difference between the original design variables d_0 and those for the perturbed problem's solution were then compared with a vector norm $||d_{new} - d_0||$. Each perturbation magnitude was repeated 256 times. The results, shown in Figs. 7 and 8, reveal that the design parameters most often vary smoothly with small changes in target locations. Larger changes in the locally optimal design occur more frequently with larger target perturbations, when the solver converges on a distinct IK solution or flips the joint axes to better reach the targets.

V. JOINT ELIMINATION

In this section we seek to design a robot with the minimal number of degrees-of-freedom (DoF). A manipulator with fewer joints may have a more limited workspace, but in return offers lower mass, power use, computation, and complexity. In the above sections, we optimized a design given an initial

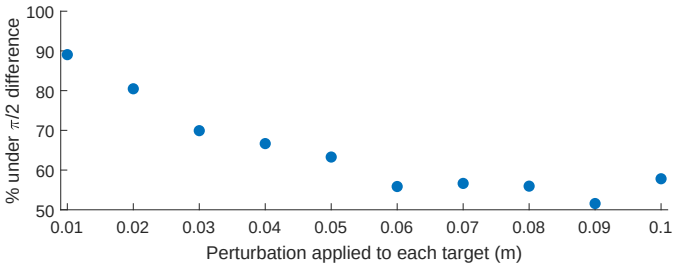

Fig. 7: The results of an experiment described in IV-C. The difference between the original design and the re-optimized design falls as the target locations are farther perturbed. The target locations for the backpack arm application were shifted by 256 random directions for each perturbation value. This plot shows the percent of the results with a norm of the design difference less than $\pi/2$. Target changes which cause a substantial change in the locally optimal design occur more frequently with larger perturbations We set $\pi/2$ as the threshold because it indicates that no joint axis directions flipped. Such a flip would cause a π change in a design variable.

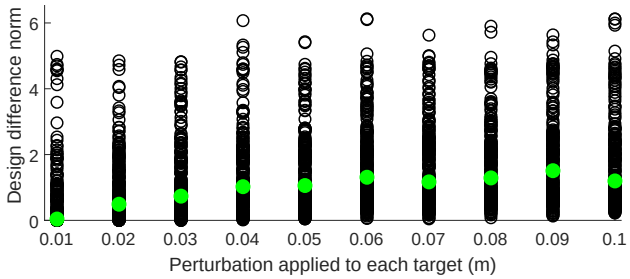

Fig. 8: The distribution of results from Fig. 7, showing the spread (black circles) and median (green filled circles).

guess at the necessary number of joints. The next step in our proposed framework is a novel method to eliminate joints. We can begin with a complex mechanism and iteratively simplify the design.

Our method is similar in principle to that used in [6], but with a significantly different goal. Rather than use the null space/implicit function theorem to optimize design parameters, we substitute a link for a set of abstract design parameters, then drive those design parameters to zero such that two subsequent joints of the same type can be merged into one. We use the results of the elimination process as an initial seed to re-optimize the design and trajectory.

A. Method

In the above optimization procedure, we calculated the configuration and design variables to reach each target, i.e., we jointly calculated the design and IK for a series of end-effector targets. The manipulator Jacobian J describes the instantaneous spatial velocity $V^s \in \mathbb{R}^6$ of an end-effector given the rate of change of its DoF [19]. For a configuration θ and design d, treated momentarily as degrees-of-freedom, we construct the Jacobian at each of the n_T targets,

$$V_i^s = J_i(\theta_i, d) \begin{bmatrix} \dot{\theta}_i \\ \dot{d} \end{bmatrix} \quad i = 1, \ldots, n_T, \quad \dot{\theta} \in \mathbb{R}^n, \quad \dot{d} \in \mathbb{R}^m. \tag{9}$$

Next, inspired by the idea of a *hand Jacobian* [19], we stack the Jacobian matrices,

$$\begin{bmatrix} V_1^s \\ \vdots \\ V_{n_T}^s \end{bmatrix} = \begin{bmatrix} J_1(\theta_1, d) \begin{bmatrix} \dot{\theta}_1 \\ \dot{d} \end{bmatrix} \\ \vdots \\ J_{n_T}(\theta_{n_T}, d) \begin{bmatrix} \dot{\theta}_{n_T} \\ \dot{d} \end{bmatrix} \end{bmatrix}, \tag{10}$$

and observe that the columns of each Jacobian can be rearranged to separate the columns caused by joint angle variation and those caused by design variable variation,

$$J_i \begin{bmatrix} \dot{\theta}_i \\ \dot{d} \end{bmatrix} = \begin{bmatrix} J_{i,\theta}\dot{\theta}_i & J_{i,d}\dot{d} \end{bmatrix}. \tag{11}$$

Each design parameter change $\dot{d}$ effects the arm at all targets, so we can rearrange (10) to combine the effects of the design and configuration variations on the end-effector velocity,

$$\begin{bmatrix} V_1^s \\ \vdots \\ V_{n_T}^s \end{bmatrix} = \begin{bmatrix} J_{1,\theta} & \ldots & 0 & J_{1,d} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & J_{n_T,\theta} & J_{n_T,d} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_{n_T} \\ \dot{d} \end{bmatrix}$$
$$V_{\text{stack}}^s = J_{\text{stack}} \begin{bmatrix} \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_{n_T} \\ \dot{d} \end{bmatrix}. \tag{12}$$

We will use this equation in Algorithm 1 to reduce the number of joints in the mechanism. Beginning with a manipulator with n joints, we select a segment to eliminate, which we will call the "elimination segment." We attempt to reduce its transformation to the identity, such that joints at its input and output can be merged. As input to Algorithm 1, we remove all design parameters in d associated with the elimination segment, and denote the remaining design parameters d'.

Algorithm 1 uses the notation in Sect. III with additional auxiliary variables and operators:

- $T_j \in \text{SE}(3)$ is the transformation induced by the elimination segment at the start of the algorithm.
- SE3toXYZRPY: $\text{SE}(3) \mapsto \mathbb{R}^6$ converts a transformation into a Euler angles parameterization, with Cartesian components and roll, pitch, and yaw.
- $(\cdot)^\vee$ is the vee operator and $(\cdot)^\wedge$ the skew operator [19].
- α is a small step size.

We seek to drive the elimination segment parameters d_j to zero while avoiding movement of the end-effector away from its targets. We iteratively invert (12) in a method reminiscent of a Jacobian pseudo-inverse algorithm for IK [20]. In line 5, J_{stack} is separated into columns corresponding to d_j in one matrix J_{dj}, and the remainder of the columns into $J_{\theta,d'}$. In line 13, the $J_{\theta,d'}$ term serves to vary the design based on the combined design and configuration null space. In lines 6–11, V_{stack}^s is calculated, and then is used within line 13 to move the end-effector back toward the targets. This term is included because changes in elimination segment parameters may not be possible without some movement of the end-effector, and

Algorithm 1 Joint Elimination Algorithm

Require: Initial design variables d' and elimination segment transformation T_j

$d_j = \text{SE3toXYZRPY}(T_j)$
iter $= 0$
while (iter $<$ max iter) and ($||d_j|| >$ threshold) **do**
  Calculate J_{stack} in Eq. (12)
  Separate columns of J_{stack} into J_{dj} and $J_{\theta,d'}$
  for $i \leftarrow 1, n_T$ **do**
    $\Delta p_i = p_{T,i} - p_{fk,i}, \quad \omega_i = \hat{n}_{fk,i} \times \hat{n}_{T,i}$
    $\Delta g_i = \begin{bmatrix} (\omega_i)^\wedge R_{fk,i} & \Delta p_i \\ 0 & 0 \end{bmatrix}$
    $g_i = \begin{bmatrix} R_{fk,i} & p_{fk,i} \\ 0 & 1 \end{bmatrix}$
    $V_i^s = (\Delta g g_i^{-1})^\vee$
  $V_{\text{stack}}^s = [V_1^s, \ldots, V_{n_T}^s]$
  $\Delta d_j = -\alpha d_j$
  $\begin{bmatrix} \Delta\theta_1 \\ \vdots \\ \Delta\theta_{n_T} \\ \Delta d' \end{bmatrix} = J_{\theta,d'}^\dagger (V_{\text{stack}}^s - J_{dj}\Delta d_j)$
  Clamp $\Delta\theta_i$ and $\Delta d'$ ▷ Prevent overly large steps
  $\theta_i \leftarrow \theta_i + \Delta\theta_i$, $d' \leftarrow d' + \Delta d'$, $d'_j \leftarrow d_j + \Delta d_j$
  Clamp d' ▷ Keep design parameters within bounds
  iter $\leftarrow$ iter + 1
for $i \leftarrow 1, n_T$ **do**
  $\theta'^i = [\theta_i^1, \ldots, \theta_i^j + \theta_i^{j+1}, \ldots, \theta_i^n]$ ▷ Merge joints
  $[p_{fk,i}, \hat{n}_{fk,i}] = \text{FK}(\theta'^i, d')$
$\theta' = [\theta'^1, \ldots, \theta'^{n_T}]$
$\delta = \sum_{i=1}^{n_T} \left(||p_{T,i} - p_{fk,i}||^2 + (1 - \hat{n}_{fk,i} \cdot \hat{n}_{T,i}) \right)$
return d', θ', δ

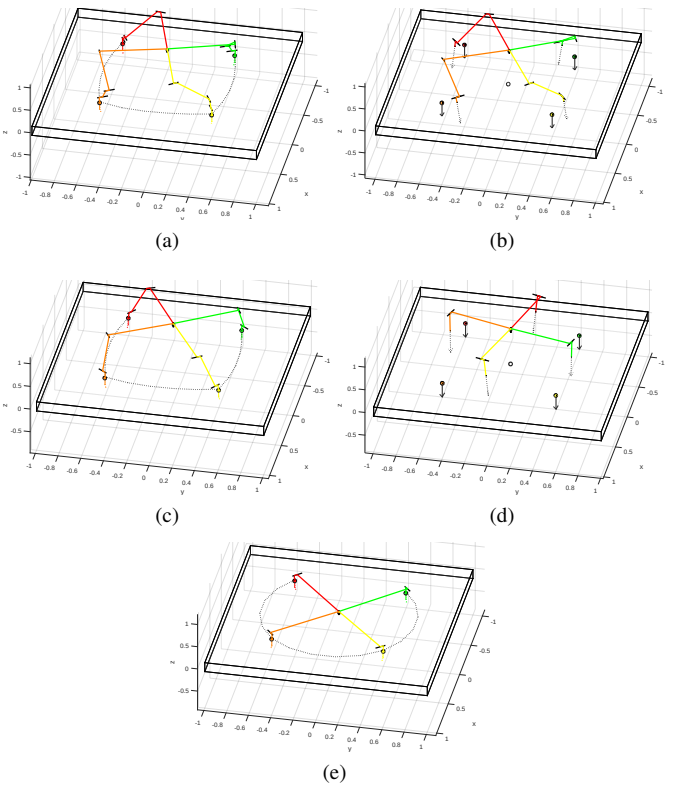

Fig. 9: In this example we iteratively eliminated degrees-of-freedom, beginning with four joints and ending with two joints. Each color (red, orange, yellow, green) represents the same arm reaching a different target. The solid black lines mark the joint axes. (a) We first design an arm with four joints with a motion path (black dashed line) to reach four targets. (b) We then use our joint elimination procedure to merge the third and fourth joints, although some drift from the targets is introduced in the process. (c) Our design and path synthesis procedure is used to refine the reduced mechanism with three joints. (d) The second and third joints are merged. (e) The reduced mechanism with two joints is refined.

numerical drift will accumulate due to finite step sizes taken on an equation that is only exact differentially.

We apply this computationally inexpensive algorithm on each joint $j = 2, \ldots, n$ to check which joint, when eliminated, returns the lowest end-effector error δ. Then the returned design parameters d' and configurations θ' are used as initial point in the optimization process in Sect. III to solve for the trajectory and to correct for numerical drift.

We now have a complete framework for iterative design synthesis and degree-of-freedom minimization. We select a series of modules with variable design parameters, including as many actuated joints as would be acceptable. We then alternate between solving the NLP (1) for design and trajectory, and using Algorithm 1 to eliminate joints.

B. Results and Discussion

We demonstrated our design synthesis and simplification procedure in a simple example. We selected four target points forming a square on a plane, with target end-effector tip axes pointing opposite the plane normal. We optimized the design parameters for a manipulator with four rotary actuators to reach these targets using our approach in Sec. III.

We then used the joint elimination procedure to identify and remove joints. Beginning with the four-joint mechanism, we first merged the third and fourth joint. We re-optimize the design using the joint elimination output as initial point, resulting in an optimized three-joint mechanism. We repeat this process once more, merging the second and third joint.

The results of this process are shown in Fig. 9. We used 500 iterations for each joint removal stage, with a step size $\alpha = 10^{-2}$, clamped V^s terms at 0.5 for linear quantities and 0.05 for angular quantities, and clamped incremental design changes $\Delta d'$ at 0.1 for linear quantities and 0.01 for angular quantities. These parameters were tuned experimentally by hand. Smaller values tend to increase algorithm stability but require more iterations to converge. We began with an arbitrary guess at the number of necessary joints, synthesized a manipulator, then eliminated two joints without preventing it from reaching the workspace targets.

Under restrictive geometric, task, and free space assumptions, the number of joints needed to kinematically reach a set of points can be determined analytically [9]. However, we seek to design robots given some design parameter limits, torque limits, and obstacles, making analytical approaches impractical. After each joint elimination, we re-optimize the design with fewer DoF, using an initial seed derived directly from the solution with more DoF. The solver terminates at

the nearest local minima, enforcing feasibility of the reduced design. This method allows the user to explore varying DoF with less computation time than attempting to solve (1) for each possible number of DoF.

VI. LIMITATIONS AND FUTURE WORK

We have identified directions for future work both in our design synthesis and joint elimination procedures. One challenge in the synthesis process is the existence of multiple IK solutions or distinct self-motion manifolds [21], each of which may result in a significantly different objective function value. A recent planning method samples multiple IK solutions at each point along the trajectory then searches for the most efficient and obstacle-free joint trajectory [22]; we hope to incorporate these ideas into our future methods.

Another challenge is that expert knowledge, or trial and error, is required in selecting the set of modules and target locations. An open research area is how to combine selection of the discrete set of modules [23] with synthesis of continuous design parameters. In this work we require the user to input task workspace targets. In the future we hope to allow the user to select targets based on higher level task descriptions or demonstrations [24].

The results of the synthesis are not always physically realizable, either from self-collisions (which could be implemented as in [12]), or because the solver did not find a feasible solution when starting from any of the initial seeds. The user must therefore check that the solution is implementable. Future work will consider using analytical solution criteria to provide bounds on whether a solution is possible for a given problem, or to provide initial seeds based on a simplified version of the problem.

Our joint elimination method is similar in principle to methods in [6]. They observed that moving through the manifold of valid designs via iterative Jacobian inversion made it difficult to incorporate nonlinear constraints or obstacles, so they replaced part of their algorithm with constrained optimization. We may similarly want to replace the inner loop of Algorithm 1 with an optimization subroutine so that we may include the objectives from (1) which we do not yet consider within the joint elimination step.

VII. CONCLUSIONS

In this work we explored a design philosophy in which we gain robot versatility not by designing a robot that can execute a wide range of different tasks, but by the ease with which we can specialize a simple robot for a single task. We presented a new method to co-optimize the design parameters and trajectories for a serial manipulator in the presence of obstacles, and validated it in hardware for two distinct applications. The key to the success of this approach lies in the use of computationally inexpensive objective and constraint gradients. Our method can constrain joint torques, reach many goal targets, and start from infeasible initial seeds. We then developed a method, based in screw theory, to systematically eliminate joints in a manipulator. We used our design optimization and joint elimination procedures to iteratively design and simplify robotic manipulators. These tools can help a user explore the design space for a given task, as a step toward full automation of the design process.

REFERENCES

[1] C. Baykal and R. Alterovitz, "Asymptotically optimal design of piecewise cylindrical robots using motion planning," in *Robotics: Science and Systems*, 2017.

[2] K. M. Digumarti, C. Gehring, S. Coros, J. Hwangbo, and R. Siegwart, "Concurrent optimization of mechanical design and locomotion control of a legged robot," in *Mobile Service Robotics*. World Scientific, 2014, pp. 315–323.

[3] C. Lehnert, T. Perez, and C. McCool, "Optimisation-based design of a manipulator for harvesting capsicum," in *Robotics and Automation (ICRA), 2015 IEEE Int. Conf. on*, 2015.

[4] C. Leger, *Darwin2K: An evolutionary approach to automated design for robotics*. Springer Science & Business Media, 2012, vol. 574.

[5] J. Papon, R. Detry, P. Vieira, S. Brooks, T. Srinivasan, A. Peterson, and E. Kulczycki, "Martian fetch: Finding and retrieving sample-tubes on the surface of mars," in *2017 Aerospace Conf.* IEEE, 2017.

[6] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Computational co-optimization of design parameters and motion trajectories for robotic systems," *The Int. Journal of Robotics Research*, p. 0278364918771172.

[7] F. C. Park and R. W. Brockett, "Kinematic dexterity of robotic mechanisms," *The Int. Journal of Robotics Research*, vol. 13, no. 1, pp. 1–15, 1994.

[8] C. Mavroidis, E. Lee, and M. Alam, "A new polynomial solution to the geometric design problem of spatial RR robot manipulators using the denavit and hartenberg parameters," *Journal of Mechanical Design*, vol. 123, no. 1, pp. 58–67, 2001.

[9] J. M. McCarthy and G. S. Soh, *Geometric design of linkages*. Springer Science & Business Media, 2010, vol. 11.

[10] G. S. Chirikjian, "Kinematic synthesis of mechanisms and robotic manipulators with binary actuators," *Journal of Mechanical Design*, vol. 117, no. 4, pp. 573–580, 1995.

[11] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros, "Interactive design of 3d-printable robotic creatures," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, 2015.

[12] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The Int. Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[13] A. Williams, S. Barrus, R. K. Morley, and P. Shirley, "An efficient and robust ray-box intersection algorithm," in *ACM SIGGRAPH 2005 Courses*. ACM, 2005.

[14] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[15] Hebi Robotics. (2018) Hebi robotics. [Online]. Available: https://www.hebirobotics.com/

[16] Z. Bright and H. H. Asada, "Supernumerary robotic limbs for human augmentation in overhead assembly tasks," in *Robotics: Science and Systems*, 2017.

[17] F. Parietti and H. H. Asada, "Supernumerary robotic limbs for human body support," *IEEE Trans. Robotics*, vol. 32, no. 2, pp. 301–311, 2016.

[18] T. Sasaki, M. Saraiji, C. L. Fernando, K. Minamizawa, and M. Inami, "Metalimbs: Metamorphosis for multiple arms interaction using artificial limbs," in *ACM SIGGRAPH 2017 Posters*. ACM, 2017.

[19] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[20] S. R. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, 2004.

[21] J. W. Burdick, "On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds," in *Advanced Robotics: 1989*. Springer, 1989, pp. 25–34.

[22] R. Holladay, O. Salzman, and S. Srinivasa, "Minimizing task space Fréchet error via efficient incremental graph search," *arXiv preprint arXiv:1710.06738*, 2017.

[23] S. Ha, S. Coros, A. Alspach, J. M. Bern, J. Kim, and K. Yamane, "Computational design of robotic devices from high-level motion specifications," *IEEE Transactions on Robotics*, no. 99, 2018.

[24] A. Giusti, M. J. Zeestraten, E. Icer, A. Pereira, D. G. Caldwell, S. Calinon, and M. Althoff, "Flexible automation driven by demonstration," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/18, 2018.