2017-04-01

Design Optimization and Motion Planning For Pneumatically-Actuated Manipulators

Daniel Mark Bodily
Brigham Young University

Design Optimization and Motion Planning For Pneumatically-Actuated Manipulators

Daniel Mark Bodily

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Marc D. Killpack, Chair
Mark B. Colton
John D. Hedengren

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Design Optimization and Motion Planning For Pneumatically-Actuated Manipulators

Daniel Mark Bodily
Department of Mechanical Engineering, BYU
Master of Science

Soft robotic systems are becoming increasingly popular as they are generally safer, lighter, and easier to manufacture than their more rigid, traditional counterparts. These advantages allow an increased sense of freedom in both the design and operation of these platforms. In this work, we seek methods of leveraging this freedom to both design and plan motions for two different serial-chain, pneumatically actuated manipulators developed by Pneubotics, a small startup company based in San Francisco. In doing so, we focus primarily on two related endeavors: (1) the optimal kinematic design of these and other similar robots (i.e., choosing link lengths, base positioning, etc.), and (2) the planning of smooth paths in joint space that enable these robots to perform useful tasks.

Our method of design optimization employs a genetic algorithm in combination with max-imin multi-objective optimization techniques to efficiently generate a diverse set of Pareto optimal designs. This set represents the optimal region of the design space and highlights inherent tradeoffs that designers must make when choosing a particular set of design parameters for manufacture. In our work, we have chosen to optimize inflatable robots to be both dexterous, and to be able to support loads near the ground with limited deflection. We have also applied our framework to optimize a plastic manipulator to perform painting motions.

In our approach to motion planning we simultaneously optimize the base position and joint motions of a robot in order to enable its end effector to follow a smooth desired trajectory. While this method of path planning generalizes to any kind of robot, we envision it to be especially applicable to soft robots and other mobile robots that can be quickly and easily repositioned to perform tasks in varying environments. Our method of path planning works by moving a set of virtual robot arms (each representing a single configuration in a sequence) branching from a common base, to a number of assigned target poses associated with a task. Additional goals and hard constraints (including joint limits) are naturally incorporated. The optimization problem at the core of this method is a quadratic program, allowing constrained high-dimensional problems to be solved in very little time. We demonstrate our method by planning and performing painting motion on two different systems. We also demonstrate in simulation how our planner could be used to perform several common tasks including those involving, pick-and-place, wiping and wrapping motions.

Keywords: Mobile Robots, Soft Robots, Design Optimization, Motion and Path Planning, Kinematics

ACKNOWLEDGMENTS

I'd like to express my thanks to my wife whose patience and ongoing support has sustained and uplifted me throughout this time. She's given me a smile and a laugh when I've needed one as well as a patient ear when I've felt like sharing an abstract idea.

I am also indebted to my parents and siblings for their support. Their guidance at critical junctures in my life gave me the confidence I needed to press forward with my talents and pave my own path of success. My mother's sustained optimism has been particularly helpful and inspiring.

My appreciation extends to those who enabled this work. In this regard it is especially appropriate to express heartfelt appreciation to the Intelligent Robotics Group at NASA who has funded this work. Also included in this category are my collaborators at Otherlab. A special thanks goes to Kevin Albert, Maria Talleria and Thomas Allen for their guidance and support of this project. It is truly a pleasure to work with such a bright and visionary team.

My thanks goes to those at BYU who have supported this research. This includes my fellow peers in the lab. In addition to providing valuable insight relating to my research, they have become close friends. I'd also like to express thanks to those serving on my graduate committee, including John Hedengren and Mark Colton for their willingness to oversee my work.

Finally, it's been my pleasure to work with Marc Killpack who has served as my graduate advisor. Marc has displayed every quality I look for in an advisor. His wisdom, congeniality, and approachability made him easy and valuable to work with. His diligent work in ascertaining funding was core to our lab's success. Finally, aside from his work as a technical advisor, Marc has been very supportive of me in my personal life and has helped me navigate times of both extreme difficulty and great success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

General

θ	All controllable degrees of freedom (DOF) (base DoFs included in planning section)
$\Delta\theta$	A small change in joint parameters
θ_i	Controllable degrees of freedom (DOF) of the ith configuration
θ_{ij}	The jth joint angle corresponding to the ith configuration
$\bar{\theta}$	The full set of joint angles, including non-controllable degrees of freedom ψ and v
M	The homogenous transform between the world and robot base frames (also denoted as $g_{w_{b_0}}$)
R_{ab}	The 3x3 rotation matrix marking the orientation of frame b with respect to frame a
p_{ab}	A 3x1 vector marking the position of frame b with respect to and in terms of frame a
g_{ab}	The 4x4 homogenous transform locating frame b with respect to reference frame a
$L_{i=1,2,...,n}$	A set of n link lengths
J	The Jacobian matrix mapping changes in joint variables to associated twists of a frame
J_s	The Jacobian of the arm's tool frame as expressed with respect to the world frame
J_B	The Jacobian of the tool frame as expressed with respect to the current tool frame
J_H	Similar to J_B, however the body frame orientation is rotated to be that of the world
$Ad_{g_{ab}}$	The adjoint operator that transforms twists from frame B to equivalent twists in frame A
W_A	A 6x1 wrench vector containing both linear and angular components expressed in frame A
f	The 3x1 vector of linear force components of a wrench
τ	The 3x1 vector of angular components of a wrench; τ is also used to denote joint torques
ξ	A 6x1 twist vector representing positional and angular velocities of a homogenous transform
v	A 3x1 velocity vector expressing linear velocity with respect to a reference frame
ω	A 3x1 angular velocity vector expressed as the axis about which a frame is rotating

Simple Rotational and Continuous Curvature Joint Modeling

w	A vector $[u, v, 0]$ about which a continuous curvature joint bends
u	The component of w representing pure bending about the controlled degree of freedom
v	The component of w that corresponds to bending about the off-axis of a joint
$\tilde{u}$, $\tilde{v}$	Normalized u and v components, u/ϕ and v/ϕ
ϕ	The l_2-norm of w. It corresponds to the total amount of bending about the vector w
s_ϕ, c_ϕ, σ	The $sin(\phi)$, $cos(\phi)$ and $\cos(\phi) - 1$ respectively
ρ	The vector perpendicular to w that locates w in and with respect to the joint base frame
h	The spine (or arc) length for each continuous curvature joint
ψ	A rotational DOF included to account for torsional deflection in the inflatable robot platform
J_s^j	A Jacobian mapping $\dot{u}$, $\dot{v}$ to twists of the jth joint's top frame with respect to its bottom

Stiffness Modeling

K_q	A single joint stiffness model relating deformations in joint parameters to applied torques
K_G	A stiffness matrix composed of joint stiffness matrices stacked diagonally
K_H	An end effector stiffness matrix relating deformations to applied loads in the hybrid frame
W^L	The max sustainable load of a design, keeping within deformation or joint torque limits
V	A deformation twist representing positional and angular movement
u_L	A 6x1 unit vector in the direction of the external loading wrench variable W^L
α	The magnitude of the maximum sustainable W^L

α_F	The magnitude of wrench W^L sustainable assuming active DOFs are infinitely stiff
α_{limit}	The magnitude of wrench W^L sustainable while staying within all joint torque constraints

Path Planning

T_t	Target corresponding to time t where t ranges from 0 (beginning of path) to 1 (end of path)
χ	A set of targets (homogenous transforms) that describe an end effector path in $SE(3)$
$\theta_{min/max}$	Vectors representing the lower and upper limits of θ (joint constraints)
$\lvert\Delta\theta_{max})\rvert$	The maximum step size of any single θ_{ij} allowed in the QP problem each iteration
J_{B_i}	The body Jacobian for the ith configuration
$J^l_{B_i}$	The body Jacobian for a frame placed in the center of link l on configuration i
$J^M_{B_i}$	The body Jacobian accounting for degrees of freedom added to the base of the robot
J^G_B	A body Jacobian for a robot with many configurations branching from a single base
$\bar{V}$	A set of concatenated deformation twists
$\tilde{V}$	A set of concatenated, desired twists taken between end effectors and desired path targets
P	A 6x6 weighting matrix added on quadratic terms (subscripts denote weighting on subterms)
L	A 6x6 weighting matrix added on linear terms (subscripts denote weighting on subterms)
κ	A scalar applied to the objective prioritizing path smoothness
γ	A scalar applied to the objective prioritizing path manipulability
β	A scalar applied to a term in the QP problem encouraging joints to be far from joint stops
G	A set of convex geometric objects representing collision barriers (points, lines, planes)
c	A scalar applied to the objective prioritizing collision free paths
d	The distance a link is from touching a collision element
ε	The distance at which a term is added to drive the violating link away from a nearby collision
e_p	The sum of weighted errors of configurations from their path targets
Ω	The improvement tolerance between iterations under which secondary weights are scaled by λ
λ	If e_p is not improving, $\lambda \in [0,1]$ is used to scale secondary objectives automatically
α_{step}	A step size found by performing a line search in the optimal direction identified by the QP
Φ	Tolerance of error on e_p under which the algorithm terminates
ζ	Parameterized path distance tolerance under which refinement is deemed unnecessary

Subscripts, superscripts, and other indicators

$\hat{[\,]}$	The hat operator as defined in Eq. 2.4 in [2] for 3x1 vectors, and Eq. 2.31 in [2] for 6x1 twists
$[\,]^\vee$	The inverse hat operator
$\lvert\lvert\ \rvert\rvert_{2,P}$	A weighted 2-norm. For example $\lVert x\rVert_{2,P} = \sqrt{x^T P x}$ where P is a weighting matrix

CHAPTER 1. INTRODUCTION

Soft robots are becoming increasingly popular as they provide the necessary compliance for systems operating around humans and in sensitive environments [3–6]. These robots have the strong advantage of being able to safely collide and interact with their environment with less risk of causing damage or injury. Whereas rigid robots are traditionally confined to cages for safety reasons, these systems have the potential of working alongside humans in performing everyday tasks like cleaning, cooking, or painting. They can also be easily and cheaply transported from one task region to another because of their low overall weight.

While potential applications are diverse in nature, these soft robotic platforms come with a host of new technical challenges in regards to their optimal design and control. As this field of research is so new, those addressing these challenges are forced to, in many cases, fallback on intuition gained from working with traditional, rigid hardware or on trial and error methods. In this work we have sought to build on traditional robotic theory and develop design and path planning methodologies specifically tailored for soft robotic platforms. In doing so we focus our efforts on addressing two related problems of importance; namely, that of optimizing the design of a soft robot based on a set of user-defined metrics, and that of planning motion to enable soft robots to complete useful tasks. Both these problems are introduced in the following sections.

1.1 Problem Description: Kinematic Design Optimization

The first problem we seek to address is that of deciding how soft robots should be designed. Unencumbered by many of the traditional components associated with rigid hardware (motors, gear trains, heavy linkages, etc.), the structure of soft robots can often be easily modified to target specific application objectives. As an example, consider the structure of the inflatable arm developed by Pneubotics in Fig. 1.1. This manipulator is characterized by a sequence of inflated

Figure 1.1: This fully inflatable manipulator developed by Pneubotics was designed using optimization techniques presented in this work.

links and pneumatically actuated joints that, when filled with pressurized air, give the manipulator a suitable structure and ability to control.

Without constraints imposed by traditional hardware, the structure of this robot can be radically altered in order to meet application-specific requirements. Link lengths can be quickly changed to give the robot a completely new kinematic structure. Additionally, because of its light weight (about 30 lbs total), its mounting position and orientation can be easily adjusted, thus granting it a high degree of mobility. This ability to be so easily redesigned and rapidly re-positioned grants a remarkable amount of freedom to those designing these kinds of platforms.

This increased amount of freedom in design is characteristic of emerging soft robot designs. Rus and Tolley introduce a host of similar systems [7]. These systems are composed of many different light-weight materials (e.g., polymers, rubbers, fabrics, etc.) and are controlled by various actuator mechanisms (e.g., hydraulics, pneumatics, electrically activated soft actuators, etc.). Marchese et al. present a compliant continuum manipulator entirely composed of silicone rubber and actuated by fluidic elastomer channels [8]. Voisembert et al. introduce an inflatable robot actuated by pneumatics [9]. These systems are similar in that they are lightweight and provide many degrees of freedom in design that may be leveraged for optimization.

With this increased freedom in design, a framework for optimally choosing application-dependent parameters (e.g., joint types, link lengths, base configuration, etc.) for these kinds of systems is desirable. In this work we present an optimization methodology suited to do this. This method relies on a genetic algorithm in which optimal designs are continually propagated from one generation to the next. In each generation, each design is scored by simulating it through many randomly-generated configurations and computing metrics of interest at each configuration. These metrics are then combined in a multi-objective fitness function that assigns a single score to each design based on both its optimality (as measured by chosen design metrics) and its uniqueness within its gene pool. Child designs are then generated from well performing designs and the process repeats. The final result is a set of diverse designs lying along a Pareto front spanning the optimal region of the design space. This result is particularly useful in that it highlights fundamental trade-offs inherent within the design space of these systems. This feature can aid hardware designers in making critical decisions regarding robot structure.

In addition to introducing this framework, we also define a set of custom metrics for the inflatable arm we seek to optimize (see Fig. 1.1). These metrics describe a design's capacity to support loads near the ground with acceptable deflection as well as its dexterity within its reachable workspace. We also discuss other metrics potentially helpful in design optimization. These methods and other relevant background information are presented in more detail in Chapter 3.

1.2 Problem Description: Mobile Motion Planning

The second problem we seek to address in this work is that of optimal motion planning for soft robots and other mobile platforms. Mobile manipulation robots are becoming more common and require planning algorithms that can effectively and efficiently coordinate both how a robot is positioned in relation to the task it is asked to perform, as well as how it actually goes about performing that task. The inherent compliance, light weight, and potential cost of soft robots makes them viable for many mobile manipulation tasks that were previously too difficult or had low likelihood of success.

We are primarily interested in the problem of planning motions that track paths in the operational workspace of a manipulator. Specifically, we desire a sequence of joint configurations $\theta_i \in R^n$ as well as a base pose $M \in SE(3)$ that will allow a manipulator's end effector to successfully

and optimally navigate a chosen path $\chi(t) \in SE(3)$. One of the specific applications we target in this work is planning motion for painting tasks and similar surfacing operations.

Motion planning with a predetermined end-effector trajectory may be performed by beginning at a starting configuration and using inverse kinematics methods [10, 11] to move along the desired trajectory. However this method is prone to getting stuck in local minima or requiring significant (sometimes unacceptable) re-configurations of the arm in order to complete the desired path. Figure 1.2 illustrates this problem with a simple example in which it is desired that an end effector of a two-link robot paints a wall. In this example, the manipulator fails to perform the task when a poor starting configuration is chosen and inverse kinematics is performed.

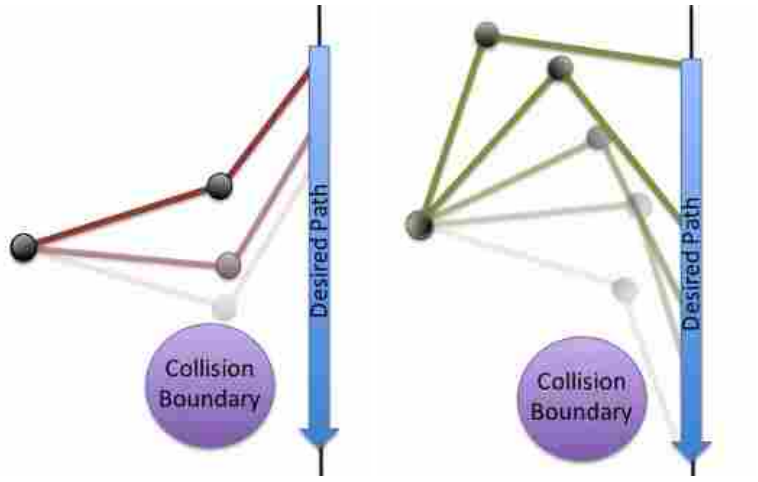


Figure 1.2: Left: Inverse Kinematics. Right: Holistic Optimization-Based Manipulator and Behavior Planning.

In this work we introduce a novel method for path planning that solves this problem. It works by branching a set of virtual arms stemming from a common base (each representing a single configuration in time) to targets along the desired path. The branching motion is determined by repeatedly solving a quadratic programming problem constructed of terms describing different motion objectives. By strategically scaling terms within this function, optimal motion along a chosen path may be achieved. In this work we focused on optimizing motion to be smooth, avoid collisions, and remain in the manipulable workspace of the robot.

1.3 Problem Motivation

The Robotics and Dynamics lab at Brigham Young University (BYU) has partnered with Pneubotics, a startup company in San Francisco, to develop and control a serial open-chain, pneumatically actuated, inflatable manipulator to test for space applications on NASA's lunar rover K-REX. Several tasks have been targeted for this manipulator to perform including holding a small jackhammer steady for sample collection, moving objects on the ground of arbitrary weight and shape, and lifting and moving objects in collaboration with humans.

The primary goal of our collaboration with Pneubotics is to aid in developing and controlling the first robotic arm that relies on pneumatics for both actuation and structure. The compliance and low-mass of this arm allows for efficient mobile manipulation in environments that are otherwise difficult to operate in. Our specific target application in this work is to design a pneumatic arm for mobile manipulation around the K-Rex rover shown in Fig 1.3. The problems and methods presented in this work are largely a result of this collaboration.

Figure 1.3: The K-Rex Mars rover during an engineering field test in October 2012 [1].

1.4 Specific Contributions

The specific contributions made by the author were:

- Developed models relating deflections of inflatable joints to associated loading conditions

- Formulated optimization metrics describing a soft robot's dexterity and capacity to support loads
- Developed and optimized a design evaluation routine used to score each candidate design
- Designed an evolutionary algorithm used to generate Pareto Fronts of optimal soft robot designs
- Designed a path planning algorithm that optimizes both an arm's base positioning and subsequent joint angle motion to a set of user-defined objectives
- Developed simulation software to test and visualize both optimization and path planning routines
- Prepared detailed documentation and published two papers in the 2017 IEEE International Conference on Robotics and Automation (see [12, 13]).

1.5 Thesis Overview

In Chapter 2 we further introduce inflatable robots and describe our methods of modeling their kinematics using constant curvature joint models and screw theory. Chapter 2 does **not** review the technical backgrounds or relevant literature of kinematic optimization or motion planning. As previous work done on these topics is fairly distinct in nature, this background is provided in Chapter 3 and 4 respectively.

We discuss our approach to kinematic design optimization in Chapter 3. After reviewing relevant literature, we describe our method in detail, discussing discretization strategies, metrics, evaluation techniques, and options available within the genetic algorithm. We also present two demonstrative examples of applying our algorithm to optimize two distinct robotic systems.

Path and motion planning for mobile and general high-degree-of-freedom robots is presented in Chapter 4. As before a technical background is provided reviewing related work, after which we discuss our method of path planning using quadratic programming techniques. The structure of the quadratic programming problem at the core of our algorithm is discussed in detail, and secondary objective terms describing path smoothness, collision avoidance, and manipulability are derived. Strategies for scaling objectives within the algorithm are also discussed. Several applications are shown to demonstrate the capabilities of the framework including painting, manipulation in cluttered environments and grasping motion.

CHAPTER 2. BACKGROUND

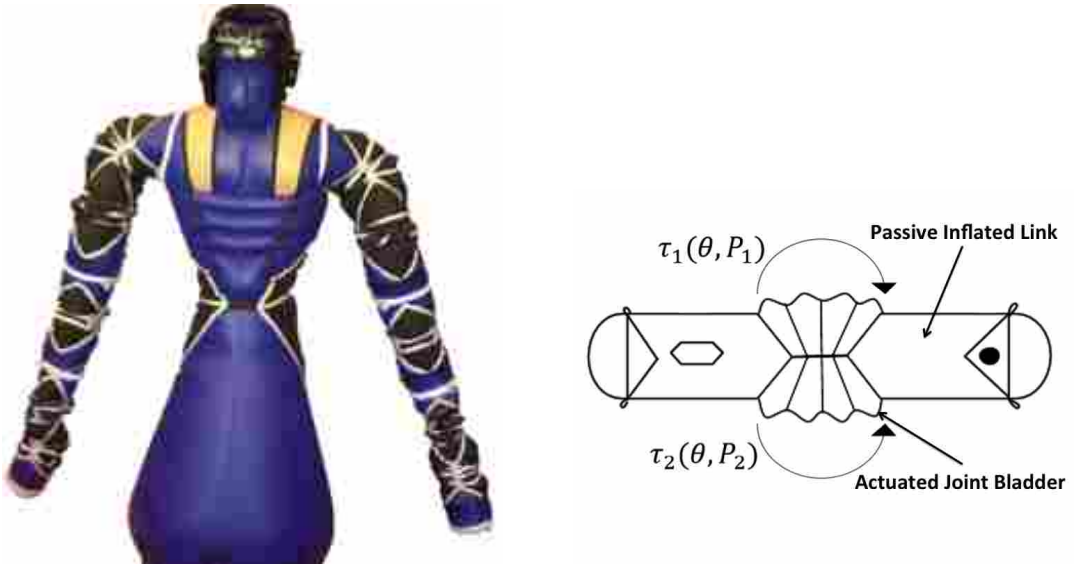
This section is meant to provide a brief overview of supporting theory used as part of this work. Background work in the areas of path planning and optimization are provided at the beginning of Chapter 3, and Chapter 4 respectively.

2.1 Soft Robotic Platforms

Traditional robotic manipulators employ rigid links for accurate position and force control. While these rigid links provide the manipulator with robust structure and precision, they often introduce problems associated with their weight. They perform poorly, for example, in unstructured environments, where unexpected collisions can cause damage to sensitive equipment and, in some cases, human injury. Their weight also often precludes their use in mobile applications (e.g., space exploration or search and rescue).

In attempts to solve these and other related problems, many have begun using soft materials in the passive structures and joints of robotic manipulators [3], [4]. These soft manipulators leverage compliant materials that are orders of magnitude lighter, cheaper, and more compact than their traditional counterparts (including actively compliant robots that are sometimes referred to as soft robots as in [14]). Their low mass enables them to manipulate safely around or in collaboration with humans in performing tasks and allows them to be used in a host of applications where robotic technologies have previously been restricted. As an example, King Louie, the first inflatable robot used by BYU's Robotics and Dynamics Lab as a testbed for research in soft robotics, is approximately 10x lighter than the similar sized robot used by NASA's Robonaut 2 (approximately 33 lbs vs 330 lbs [15]). Moreover it can be quickly deflated and stored in a compact 3'x2'x1' storage box for efficient transport.

In this work we investigate design and control methods for an arm similar to that of one of King Louie's arms (see Fig. 1.1). This robotic arm, like that of King Louie's, is completely



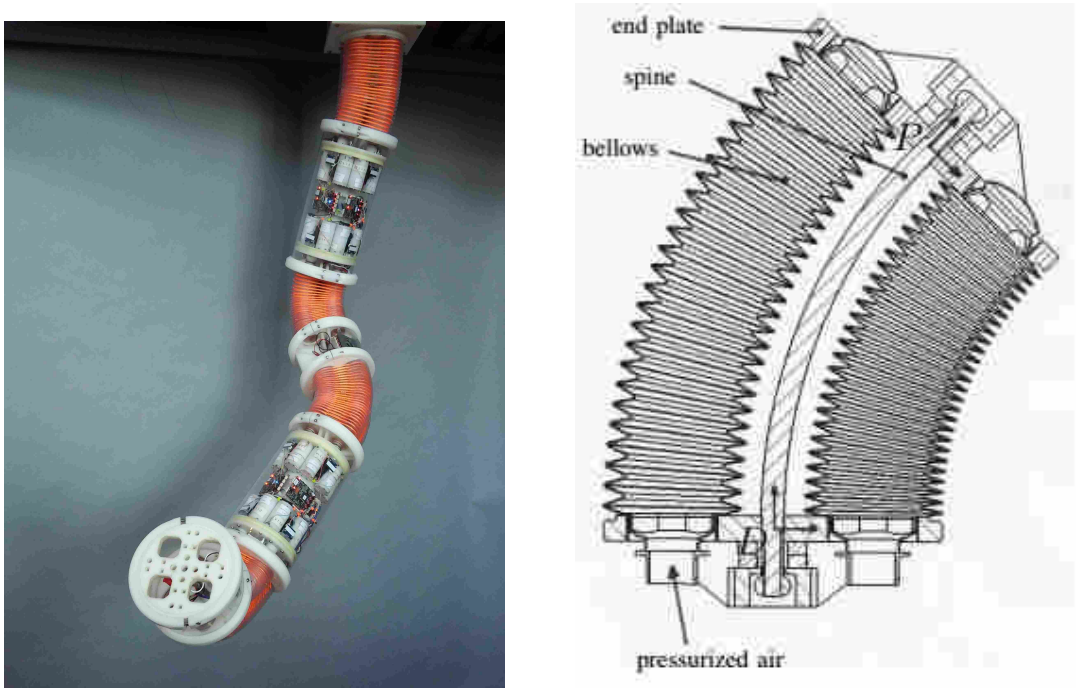
(a) King Louie (b) Joint Bladders

Figure 2.1: King Louie is a research robot developed by Pneubotics and used in BYU's Robotics and Dynamics Lab. It is completely inflatable, relying on pressurized air for both structure and control.

inflatable and is comprised of passive and active bladders as shown in Fig. 2.1b. Each pneumatic joint is controlled to a desired angle or torque by adjusting the air pressures of two antagonistic bladders within that joint. Neighboring joints are each offset by 90° to grant the manipulator a greater range of motion (e.g., a neighboring joint of that depicted in Fig. 2.1b would bend in and out of the page). By simultaneously adjusting the bladder pressures for each joint, the arm can be accurately controlled to perform coordinated motion and carry out meaningful tasks (see [6, 16] for specifics on the control strategies).

While this inflatable platform offers significant advantages over its counterparts, many challenges related to its optimal design and control are still unresolved. One challenge we seek to address in this work is that of accurately modeling structural deflections induced by loading conditions on this inflatable system. In early testing of a prototype arm it was found that external forces primarily caused deflections at the inflatable manipulator's joints, and that its links remained approximately rigid. We have accordingly added uncontrollable degrees of freedom at the joints in our models to account for these deflections.

A second platform we investigate in both the optimization and path planning chapters is shown in Fig. 2.2a. This manipulator is characterized by plastic links connected by novel pneu-



(a) Full manipulator

(b) Bellows design of a single joint

Figure 2.2: This blow-molded joint robot developed by Pneubotics is lightweight and safer to work around the traditional robots.

matic joints. Each joint is comprised of four blow-molded bellows that, when filled with pressurized air, allow it to bend continuously along its length as depicted in Fig. 2.2b. Similar to the inflatable platforms previously introduced, this pneumatically actuated arm is lightweight making it easier to transport and safer to work around than traditional rigid robots. Additionally, its plastic links grant the manipulator a more rigid structure (in comparison to the fabric-based arm in Fig. 1.1) that lends itself to simpler control and estimation schemes.

2.2 Kinematic Modeling

Both the optimization and path planning routines introduced in Chapters 3 and 4 require kinematic models of a manipulator in order to predict and evaluate its behavior. In this work we model both inflatable and blow-molded joints in the same manner. Our kinematic modeling follows from Murray et al. [2] with joint models taken from Allen et al. [17]. Assignment of homogeneous

transforms and forward kinematics, joint models and the computation of the manipulator's Jacobian is reviewed here.

2.2.1 Robot Structure Definition and Forward Kinematics

A kinematic model is constructed by assigning coordinate frames of interest at distinct locations along the robot as shown in Fig. 2.3. The position and orientation of one frame with respect to another may be conveniently expressed using homogeneous transforms. These are 4x4 matrices, each comprised of a 3x3 rotation matrix $R \in SO(3)$ describing changes in orientation and a 3x1 vector $p \in R^3$ describing changes in position relative to another frame's location. In this work, we denote the transform locating frame b with respect to and in terms of frame a as,

$$g_{ab}(R_{ab}, p_{ab}) = \begin{bmatrix} R_{ab} & p_{ab} \\ 0 & 1 \end{bmatrix} \tag{2.1}$$

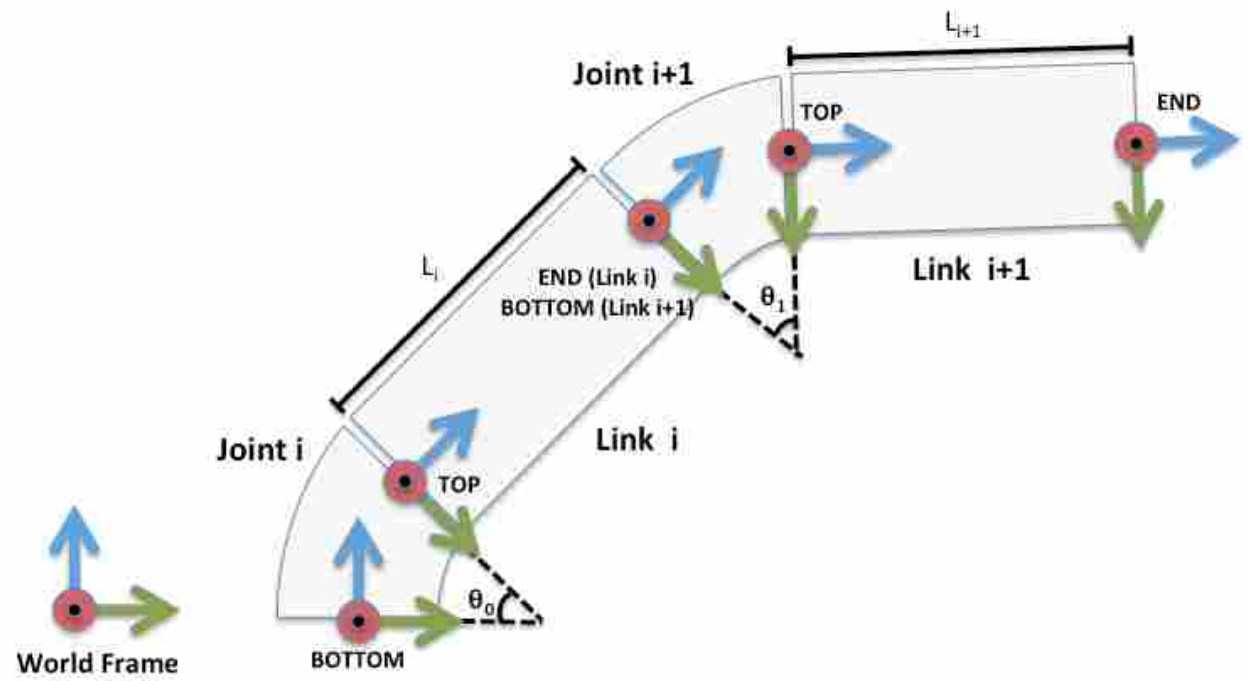


Figure 2.3: Important frames of reference labeled on our robot with red, green and blue axes representing the x, y and z axes of each frame respectively. Red axes point out of the page.

Homogeneous transforms may be multiplied together to move between multiple frames. For example, knowing the transform g_{wb_i} locating the bottom of the ith joint (see Fig. 2.3) in relation to the world frame (as expressed in the world frame), we can find a frame fixed to the top of that joint, g_{wt_i}, in the world frame by computing,

$$g_{wt_i} = g_{wb_i} g_{bt_i} \tag{2.2}$$

where g_{bt_i} is a homogeneous transform locating the top frame with respect to the bottom of that joint (as expressed in the bottom frame).

Following a sequence of similar multiplications, we can locate the position and orientation of the robot's end effector with respect to a fixed world frame. To do this, every link and joint pair is assigned three frames of interest; one placed at the bottom of the joint, one placed at the top of the joint, and one placed at the link's end as shown. Transforms between these coordinate frames are denoted as g_{bt_i} and g_{te_i} respectively. For inflatable joints we introduce a simple rotary joint before each curvature joint to model torsional deflection (deflection about the +z axis of each joint bottom frame). As inflatable joints are stacked orthogonally, we also add a nominal 90° offset at each of these same transitions. Both the offset and torsional bending are modeled together as $g_{e_{i-1}b_i}$.[1] Assigning a transformation from the world frame to the base of the robot, g_{we_0}, we can then perform forward kinematics from the world frame to the end of the nth link as,

$$g_{we_n} = g_{we_0} \prod_{i=1}^{i=n} g_{e_{i-1}b_i} g_{bt_i} g_{te_i} \tag{2.3}$$

In Eq. 2.3, g_{we_0} and g_{te_i} are assumed to be fixed given a mounting position $M \in SE(3)$ and a set of link lengths $L_{i=1,2,\ldots,n}$. This assumes link bodies to be rigid and the mounting position to be unchanging. This assumption is appropriate as initial testing on prototype arms has shown load-induced deflections to be primarily present at joints rather than at links. These deflections are thus modeled within the joint transformations g_{bt_i} and within the offset transformations between joint-link pairs $g_{e_{i-1}b_i}$. We discuss these transformations in the following sections.

[1] This transformation is left as the identity matrix for blow-molded joints as these joints do not exhibit similar deflection patterns.

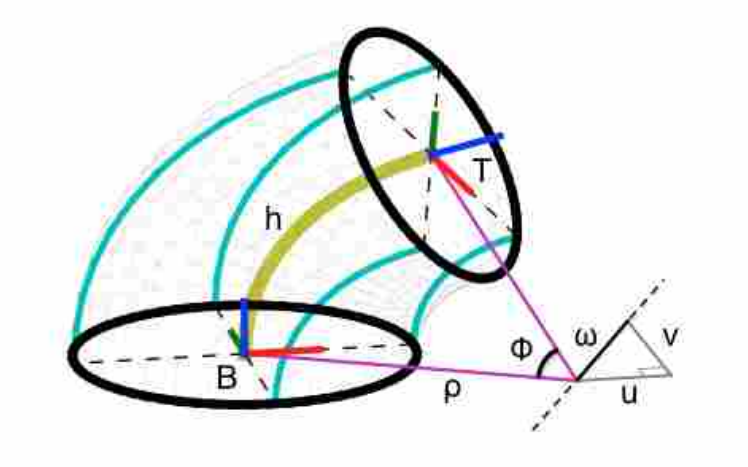


Figure 2.4: Parameterization of a joint with constant curvature

2.2.2 Continuous Curvature Joint Models

For this thesis, we model both inflatable and blow-molded joint types as continuous curvature joints as outlined by Allen et al. [17]. In this model a joint spine of fixed length h curls in a circular arc around a vector $w = [u, v, 0]$ lying in the plane tangent to the bottom of the joint as shown in Fig. 2.4. Encapsulated within this vector are two orthogonal joint state variables, u and v, that are used to model the degrees of freedom of the joint. For a blow-molded joint, we assume both these variables are controllable and that the joint can bend any direction desired by commanding appropriate pressures in its billows. For inflated joints however, the state v is not controllable and is included only to represent uncontrolled bending occurring about the joint's off axis due to loading conditions. In Fig. 2.1b this would represent bending motion in and out of the page.

Resulting motion can be described as the manipulator bending $\phi = \sqrt{u^2 + v^2}$ radians about the normalized unit vector $u_w = w/||w||$. The vector ρ locates this normalized vector from the joint base center and is found by noting that $||\rho|| = h/\phi$ where ρ is perpendicular to w. Using a simple cross product gives the relation,

$$\rho = ||\rho|| u_\rho = ||\rho||(\begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \times u_w) = \frac{h}{\phi^2} \begin{bmatrix} v \\ -u \\ 0 \end{bmatrix}. \tag{2.4}$$

With u_w, ρ, and ϕ all defined in u and v coordinates, we can compute the transformation from the base of the ith joint to the top of that same joint $g_{bt_i}(u_i, v_i)$. By substitution of these variables into Eq. 2.40 in [2] which defines a transformation for screw-like motion we get, [2]

$$g_{bt_i}(u_i, v_i) = \begin{bmatrix} e^{i\hat{u}_w\phi} & (I - e^{i\hat{u}_w\phi})\rho \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

where the hat operator on a 3x1 vector is defined as,

$$\hat{a}_{3x1} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.6}$$

Expanding the matrix exponentials and simplifying[3] we reach a closed form solution. Denoting the normalized u and v vectors as $\tilde{u} = u/\phi$ and $\tilde{v} = v/\phi$ respectively, $\cos(\phi) - 1$ as σ, and expressing $sin(\phi)$ and $cos(\phi)$ terms as s_ϕ and c_ϕ, we get,

$$g_{bt_i} = \begin{bmatrix} \sigma\tilde{v}^2 + 1 & -\sigma\tilde{u}\tilde{v} & \tilde{v}s_\phi & -\sigma h\tilde{v}/\phi \\ -\sigma\tilde{u}\tilde{v} & \sigma\tilde{u}^2 + 1 & -\tilde{u}s_\phi & \sigma h\tilde{u}/\phi \\ -\tilde{v}s_\phi & \tilde{u}s_\phi & c_\phi & hs_\phi/\phi \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

This joint transformation contains division by zero in several terms when both u and v tend to zero. However, in each case, these terms come in the form of $sin\phi/\phi$ or $(cos\phi - 1)/\phi^2$ which can be evaluated in the limit by expressing them as analytic, converging, infinite series. Another means of deriving this same transformation is by exponentiating an appropriate twist associated with this motion. We discuss this process in the next section.

[2] ρ is denoted as q in Eq. 2.40 in [2] and the variable h in Eq. 2.40 represents pitch which, in this case, is zero.

[3] The following identity known as Rodrigues' formula is useful in simplifying this expression.

$$e^{\hat{a}\theta} = I + \hat{a}sin\theta + \hat{a}^2(1 - cos\theta)$$

2.2.3 Joint Transformation Derivation (Eq. 2.7) Using Twists

Any rigid body motion can be thought of as motion consisting of a rotation about a fixed axis in space, and then subsequent translation parallel to that axis. This motion is referred to as screw motion. The infinitesimal version of a screw is known as a twist and is a convenient representation of a rigid body's linear and angular velocities. It is expressed as a 6x1 vector as,

$$\xi = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.8}$$

Twist vectors allow us to track the instantaneous relative motion of a frame of interest with respect to another reference frame. Twists will be important in our derivation of the manipulator Jacobian as will be shown later.

It can be shown that the relative motion of a moving frame as seen from the viewpoint of a fixed reference frame can be found by exponentiating an appropriate twist. Specifically, if we let $g_{ab}(0)$ be the initial configuration of a rigid body B relative to and expressed in frame A, then after a constant twist of ξ applied over a range of Θ radians (assuming $||\omega|| = 1$), its resulting position can be expressed as,

$$g_{ab}(\Theta) = e^{\hat{\xi}\Theta} g_{ab}(0) \tag{2.9}$$

where the hat operator performed on a 6x1 vector maps a it to a 4x4 matrix and is defined in Eq. 2.31 in [2] as,

$$\hat{a}_{6x1} = \begin{bmatrix} \hat{a}_2 & a_1 \\ 0 & 0 \end{bmatrix} \tag{2.10}$$

where a_1 and a_2 represent the first three and second three components of a respectively.

The case of pure rotation about an axis is of special interest in this work (see Fig. 2.4). In this case, given a vector ρ from the frame of interest to the axis of rotation, the unit normalized axis u_ω and the amount of rotation ϕ about this axis, the transformation is given in Eq. 2.9 with

$$\xi = \begin{bmatrix} -u_\omega \times \rho \\ u_\omega \end{bmatrix} \qquad \Theta = \phi \tag{2.11}$$

Exponentiating this twist using the matrix exponential (Eq. 2.9) gives Eq. 2.7 as previously noted.

2.2.4 Modeling Torsional Deflection (Inflatable Manipulators Only)

For inflatable manipulators, the offset between joint-link pairs is nominally set at 90° about the Z-axis of the previous link end frame as previously mentioned. To account for uncontrolled torsional deflection occurring at each curvature joint, we include an additional rotational degree of freedom at each joint's base with a deflection of ψ_i radians from the nominal offset. We model these additional rotary joints as separate from the curvature joint model, thus granting three degrees of freedom for each joint-link pair. Transformations associated with these joints are a simple rotation about z. With the 90° offset included, the joint-link pair interface transformation becomes,

$$g_{e_{i-1}b_i}(\psi_i) = \begin{bmatrix} cos(\psi_i + \pi/2) & -sin(\psi_i + \pi/2) & 0 & 0 \\ sin(\psi_i + \pi/2) & cos(\psi_i + \pi/2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

With known link lengths $L_{i=1,2,\ldots,n}$ and a fixed joint arc length h, a sequence of joint parameters,

$$\bar{\theta} = [\psi_1, u_1, v_1, \psi_2, u_2, v_2, \ldots, \psi_n, u_n, v_n] \tag{2.13}$$

can now be used to perform the forward kinematic routine given in Eq. 2.3. For inflatable manipulators, only the joint variables u_i within this set are controllable whereas for blow-molded arms, both u_i and v_i are controllable (ψ_i is not included for blow-molded arms). In this work we denote the controllable subset as θ. Other degrees of freedom are included only when modeling deflection from loading.

2.2.5 Computing the Manipulator Jacobian Matrix

We now shift our focus to modeling a manipulator's velocity. This is primarily done by computing the Jacobian matrix which represents a linear mapping between differential movements in a manipulator's joint states and corresponding twists of a frame fixed to the robot. For an in depth study of the Jacobian and related screw theory, the reader is advised to study Chapters 2 and 3 in [2]. This section only briefly outlines equations and supporting theory used in this work.

Computing the Jacobian For a Continuous Curvature Joint

We derive the Spatial Jacobian J_j^s as defined by Murray et al [2] for a single continuous curvature joint. This Jacobian maps differential movements in joint parameters into twists ξ_S of the top frame of that joint with respect to (and in terms of) its bottom frame as,

$$\xi_S = J_S^j[\dot{u} \quad \dot{v}]^T \tag{2.14}$$

and is computed as,

$$J_S^j = \begin{bmatrix} \left(\frac{\partial g}{\partial u} g^{-1}\right)^\vee & \left(\frac{\partial g}{\partial v} g^{-1}\right)^\vee \end{bmatrix} \tag{2.15}$$

where g denotes the joint transformation (e.g., g_{bt_i} in Eq. 2.7). The vee operator ($^\vee$) transforms a 4x4 matrix into a 6x1 twist vector and is defined as the inverse hat operator given in Eq. 2.10 (also defined in Eq. 2.30 in [2]). For the continuous curvature joint model this expression is solved in closed form as, [4]

$$J_S^j = \begin{bmatrix} 0 & \sigma h/\phi^2 \\ -\sigma h/\phi^2 & 0 \\ h\tilde{u}(\phi - s_\phi)/\phi^2 & h\tilde{v}(\phi - s_\phi)/\phi^2 \\ \tilde{u}^4 + \tilde{u}^2\tilde{v}^2 + \tilde{v}^2 s_\phi/\phi & \tilde{u}\tilde{v}(\phi - s_\phi)/\phi \\ \tilde{u}\tilde{v}(\phi - s_\phi)/\phi & \tilde{v}^4 + \tilde{u}^2\tilde{v}^2 + \tilde{v}^2 s_\phi/\phi \\ \sigma\tilde{v}/\phi & -\sigma\tilde{u}/\phi \end{bmatrix} \tag{2.16}$$

This represents the Jacobian for a single continuous curvature joint. Each column of this Jacobian gives a twist vector ξ generated from differential movement in the joint's associated state u and v. These twists are expressed with respect to the joint's base frame, a fact that will be important to remember in the next section.

Putting It All Together

Having found the continuous-curvature joint transformation g_{bt_i} in Eq. 2.7 by exponentiating the appropriate twist in Eq. 2.11, and then having derived the Jacobian of this transformation in Eq. 2.16 by plugging this result into Eq. 2.15, we are now ready to derive the manipulator's

[4] This representation breaks down when ϕ approaches zero. For an appropriate approximation of both g and J_S^j near this point, the reader is referred to Allen et al. [17].

Spatial Jacobian, J_S. This Jacobian linearly relates differential movements in all joint parameters $\bar{\theta}$, to differential movements of a frame attached to the manipulator's end effector,

$$\xi_S = J_S \dot{\bar{\theta}} \tag{2.17}$$

The resulting twist ξ_S is expressed with respect to (and in terms of) a common world frame (often the base of the robot).

To derive this Jacobian, we must express the contributions of twists generated by each joint in a single, common reference frame. To change the reference frame and location from which a twist is expressed, we use the 6x6 adjoint transformation introduced in Eq. 2.58 of [2],

$$Ad_{g_{ab}} = \begin{bmatrix} R_{ab} & \hat{p}_{ab} R_{ab} \\ 0 & R_{ab} \end{bmatrix} \tag{2.18}$$

where $\hat{p}$ is taken as a 3x3 skew-symmetric matrix (see Eq. 2.6) taken from the position vector locating the old frame of reference (frame b) relative to and in terms of the new frame (frame a). This adjoint is directly computable from the components of the homogeneous transform relating the new and old reference frames (see Eq. 2.1).

By horizontally concatenating twists generated by each joint parameter in a single matrix, and then transforming each twist to be expressed in reference to a common point in a common world frame using appropriate adjoint transformations, we arrive at the Spatial Jacobian,

$$J_S = \begin{bmatrix} \xi_1' & \xi_2' & \dots & \xi_n' \end{bmatrix} \tag{2.19}$$

$$\xi_j' = Ad_{g_{wb_j}} \xi_j \tag{2.20}$$

The transformation from the world to the base of each joint g_{wb_j} can be found by performing forward kinematics up to that joint (see Eq. 2.3). For constant curvature joints, untransformed joint twists ξ_j are given in Eq. 2.16. Rotary joints have the simple jacobian, $\xi_R = [0,0,0,0,0,1]^T$ and must also be transformed using the appropriate adjoint operation.

To obtain the manipulator's Body Jacobian, we can transform all twists in J_S to be expressed instead at the end effector frame of the current configuration,

$$J_B = Ad_{g_{we_n}^{-1}} J_S \tag{2.21}$$

This Jacobian relates joint velocities to corresponding twists of the body-fixed end effector frame with respect to and in terms of the end effector's current end effector frame $g_{we_n}(\bar{\theta})$. We will use the Body Jacobian extensively in this work when formulating the quadratic programming problem for path planning.[5]

In addition to the Body Jacobian, we will also use Link Body Jacobians J_B^l describing the movement of frames placed at the center of mass of each link. These come into play in the quadratic programming problem when implementing collision avoidance and grasping. They can be found by transforming twists in the Spatial frame to these respective frames in a similar manner as done in Eq. 2.21. In doing this, twists corresponding to distal joints relative to the link of interest must be set to zero as they do not generate additional motion in this frame.

Finally, a natural frame of interest is one attached to the end effector of the robot that maintains a constant orientation equal to that of the world. This hybrid Jacobian J_H can be found by rotating the twists in J_B into the world frame's orientation,

$$J_H = \begin{bmatrix} R_{we_n} & 0 \\ 0 & R_{we_n} \end{bmatrix} J_B \tag{2.22}$$

The Spatial, Body, and Hybrid Jacobian frames are labeled in Fig. 2.5.

[5] In this work, when a Jacobian (J_B, J_H, or J_S) is used with θ (the controllable subset of $\bar{\theta}$), columns of the Jacobian corresponding to uncontrollable degrees of freedom are dropped.

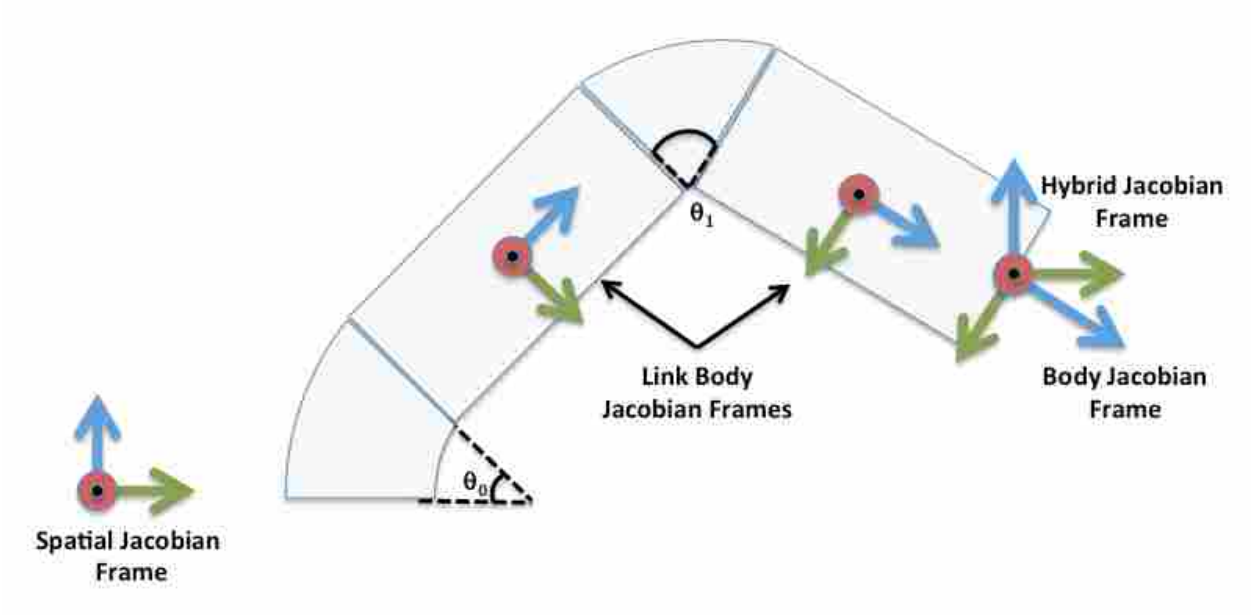


Figure 2.5: The Jacobian frames are labeled here. Twists mapped from the Spatial Jacobian, J_S, are those of the tool frame as expressed relative to and in terms of the Spatial frame (or world frame). The Body Jacobian, J_B maps joint movement to twists of the tool frame relative to and in terms of the current Body frame (or tool frame). Similarly, the Link Body Jacobians J_B^l map to twists of frames attached to the link bodies relative to and in terms of their current respective frames. The Hybrid frame is one attached to the end effector with constant orientation equal to that of the world frame. The Hybrid Jacobian J_H maps joint movement to twists of this frame as expressed in its current frame.

2.2.6 Wrenches

In this work we will make use of wrenches to describe loads applied to a soft manipulator. Wrenches denote generalized forces comprised of both linear and angular components as,

$$W = \begin{bmatrix} f \\ \tau \end{bmatrix} \tag{2.23}$$

Similar to twists, wrenches depend on the frame in which they are expressed. Wrenches can be naturally combined in a single reference frame using Eq. 2.65 in [2],

$$W_b = Ad_{g_{ab}}^T W_a \tag{2.24}$$

This equation transforms wrenches expressed in frame A to equivalent wrenches as applied in frame B. In performing this calculation, we assume frame A is rigid with respect to frame B. We will make use of these generalized force vectors extensively when we discuss stiffness modeling in Chapter 3.

CHAPTER 3. DESIGN OPTIMIZATION

In this section we discuss our method for optimizing the kinematic design of soft manipulators. We first discuss previous methods of robot design optimization, after which we present our design methodology. We demonstrate our methods on two separate types of soft robotic platforms and discuss our results.

3.1 Previous Research In Robot Design Optimization

Optimization methods applied in the field of robotics have often lagged behind the current state of the art of the broader optimization community and in practice only very simple problems are solved. As a result the optimization potential of many current systems is not fully exploited and designs are suboptimal [18]. In an effort to bridge the gap between design and optimization, the IEEE Robotics and Automation Society (RAS) Technical Committee (TC) on Model-Based Optimization for Robotics was founded in October of 2012. One of the committee's primary research objectives is to "optimize the design of robots for given tasks (parameter optimization and structural optimization)." We share this same objective in this work and focus on the application of appropriate optimization methods to the compliant platforms previously introduced.

3.1.1 Design Optimization Methods

Many different methods of design optimization for a broad range of robotic applications have been explored. Oral and Idler utilize sequential quadratic programming (SQP) to minimize the weight of a robot's links for a particular high speed movement [19]. Paredis and Khosla optimize the kinematic assembly of joint modules using simulated annealing (SA) techniques [20]. Rao and Waghmare use teaching-learning-based optimization (TLBO) to find optimal geometrical dimensions of a robot gripper [21]. Ramezan et al. employ particle swarm optimization (PSO) to reduce singularities within the workspace of a parallel manipulator [22].

Of these methods, genetic algorithms (GA) and similar stochastic techniques have been found to be particularly useful [23–26]. They are especially good for optimizing highly nonlinear problems as they are less prone to getting stuck in local minima and do not require expensive gradients. However establishing an appropriate fitness function to combine desired design objectives can be difficult, especially when these objectives are competing.

Several researchers have extended genetic algorithms to incorporate a multi-objective optimization framework, seeking to explore a Pareto front of optimal designs rather than zeroing in on one particular design [27, 28]. Finding the Pareto front is often difficult however, especially when applied to systems requiring expensive objective function calls as is the case in this work. One method is to solve an assortment of optimization problems, each weighting objectives differently in a single fitness function (see ParEGO optimization [29]). This method is slow however as many different GA problems must be solved. Additionally, scaling weighting vectors appropriately to provide a diverse, Pareto optimal set can be challenging.

Other methods such as the one presented in this work seek to operate directly in the multivariate objective function space by making use of the maximin objective function [30]. Coello et al. use a weighted form of this function in order to optimize counterweight balancing for robot arms [28]. This methodology is especially powerful because it requires only a single GA problem to be solved in order to approximate the Pareto Front.

To our knowledge, none of these optimization methods have been applied to optimize the design of inflatable robots. Part of the difficulty in performing any such optimization is establishing accurate design metrics to adequately characterize the behavior of these difficult to model systems. Our goal in this work is to establish appropriate metrics and utilize the aforementioned maximin framework to optimize the design of a fully inflatable, pneumatically-actuated manipulator. We also optimize the design of an arm comprised of blow-molded joints using this same framework as will be shown.

3.1.2 Metrics for Design Evaluation

A critical step in optimizing the design parameters of a robotic manipulator is to identify metrics that adequately reflect desired behavior. In other relevant work, metrics commonly employ the Analytical Jacobian J_A which relates joint velocities and torques to end-effector velocities and

forces respectively [31], [32], [33], [34]. Common metrics used that employ the Jacobian include the inverse of the condition number of J_A ($\kappa = \sigma_{min}/\sigma_{max}$), the smallest singular value of J_A (σ_{min}), and the volume of manipulability ellipses relating joint and end-effector velocities (proportional to $det(J_A^T J_A)$) [35, 36]. These metrics are commonly used to assess the dexterity of a robot within a workspace of interest.

The Jacobian can also be used to characterize other behaviors of interest, including how the joint torques of a manipulator scale in relation to the forces they generate at its end effector [35]. In a similar fashion a velocity transmission ratio can be defined relating joint and corresponding end effector velocities at a single pose.

These metrics provide a valuable starting point for optimization routines but have limitations that may severely restrict their use. This is especially the case with soft, deformable robots, where rigid body kinematic assumptions may be rendered invalid. Many recent advancements in soft robotic technologies fall in this latter category including electroactive polymers [37], pneumatics [6], and shape-memory alloys [38]. We present a new metric in this work that characterizes the load bearing capacity of manipulators with flexible joints.

3.1.3 Methods of Collecting Metrics Within a Manipulator's Workspace

The design metrics we have discussed are configuration dependent and thus need to be evaluated over a workspace of interest in order to provide a global sense of fitness for a particular robot design. This can be done analytically if a closed form expression describing manipulator kinematics can be obtained [34, 39]. However for high dimensional systems such a model may be unobtainable. Kumar et al. [40] propose a sampling based approach using inverse kinematics to evaluate the dexterity of a manipulator at distinct points within its workspace. Their inverse kinematic formulation relies on specific design geometry however and is not readily generalizable to other platforms. We focus instead in this paper on sampling techniques that are readily extendable to any platform, including those with many degrees of freedom. We present three sampling strategies in particular: random forward kinematic (FK) sampling, sampling using local inverse kinematics (IK), and hybrid FK-IK algorithms. We will discuss the benefits and drawbacks of each.

Figure 3.1: K-REX is a planetary rover developed by NASA for exploration in areas of rough terrain.

3.2 Methods

The motivating problem for this work is the design of a 6DOF, inflatable manipulator (similar to that shown in Fig. 1.1) that is to be mounted on the side of NASA's prototype planetary rover K-REX at NASA Ames. A picture of a preliminary design for this inflatable arm mounted to K-REX is shown in Fig. 3.1 which was designed using a single objective function for optimizing the dexterous workspace.

For this platform, we specifically seek a set of optimal structural parameters including a sequence of six optimal link lengths L_{1-6}, a vertical mounting height b_h from the ground, and a horizontal mounting angle b_θ that will allow this arm to be dexterous while also being able to manipulate heavy objects (e.g., rocks) near the ground. As these two objectives are likely competing, we wish to explore the region of optimality in the design space in which one of these objectives cannot be improved without sacrificing the other. This is commonly referred to as the Pareto front.

3.2.1 Genetic Algorithm

In this work we use maximin optimization techniques within the framework of a genetic algorithm to find the Pareto front. To do this we first generate an initial set of designs by uniformly sampling design parameters within constraints. To satisfy practical limitations and hardware constraints, designs not meeting the following design constraints are immediately discarded:

- **Armspan**: Armspan must be below $L_a \leq 1.5m$. Each joint is modeled with constant curvature and with joint length $6cm$. Thus, $\sum_{i=1}^{6} L_i \leq 1.14$
- **Max Link Length**: Single link length, $L_i \leq .5m$
- **Valves**: Consecutive links must accommodate valves used for control, $L_i + L_{i+1} \geq .18m$
- **Mount Height**: Mounting height from ground must be in range, $.3m \leq h_m \leq 1.5m$
- **Mount Angle**: The mount is allowed a fixed vertical tilt angle, $-60^\circ \leq b_\theta \leq 60^\circ$ (towards or away from the ground)

Generated designs are randomly paired and child designs are generated by performing crossover. In our application this is done by randomly assigning each child's *i*th link length to be either that of one of its corresponding parents, or the average length of the parents' *i*th link. Base mounting parameters are handled similarly. Crossover that produces a child design violating constraints is re-performed until a valid design is obtained.

We allow the possibility of mutation when generating a child design in order to avoid local minima and encourage global diversity within each generation. This is done by randomly swapping two links in the child design and then assigning a new mounting position by uniform sampling as before. Mutation is re-performed if a resulting design violates constraints. After crossover (and in some cases mutation), children and parents are evaluated using methods that will be later discussed. The best scoring design from each parent-child grouping is then passed to the next generation. To aid in local refinement of the genetic algorithm, a perturbed copy of the winner is also passed to the next generation. This copy is generated by slightly adjusting a child's link lengths and base mounting parameters (while staying within constraints). The steps of the genetic algorithm are summarized in Fig. 3.2.

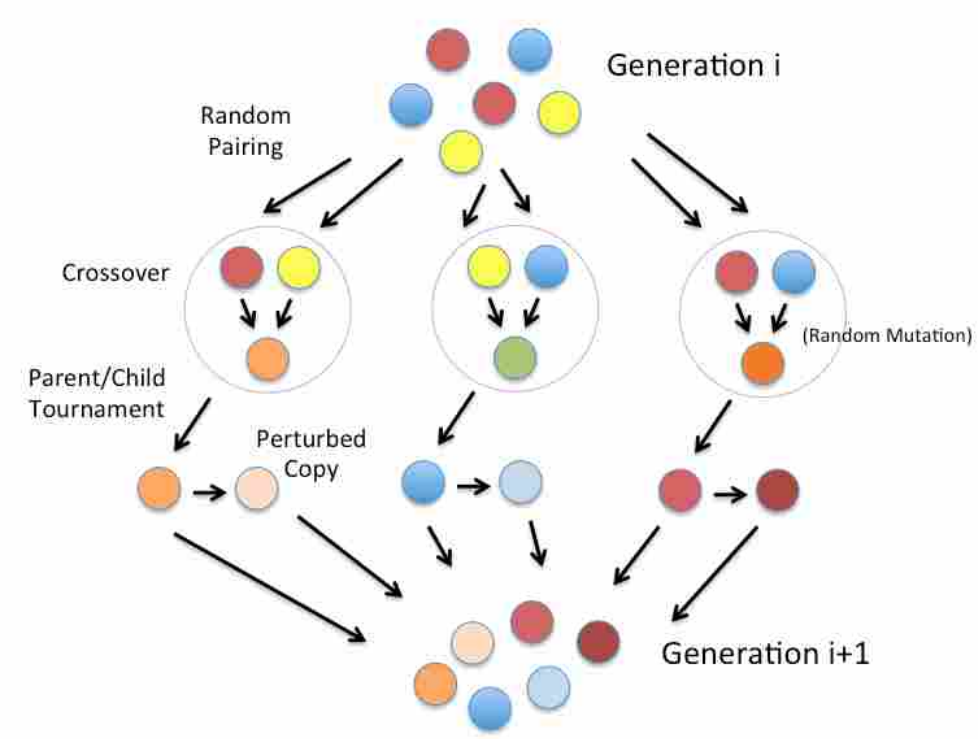


Figure 3.2: Designs are perpetuated through the genetic algorithm in a sequence of steps that include random pairing, crossover including mutation, tournament evaluation, and design perturbation.

3.2.2 Design Evaluation Methods

During the genetic algorithm every design (parents and children) undergoes an evaluation routine in which it is scored on multiple user-defined metrics. To find these scores, metrics are calculated throughout the workspace at many different simulated configurations for each design. Once the end effector has been simulated through many different configurations, these pose-specific scores are summed across the entire reachable workspace of each manipulator providing a measure of optimality for each metric of interest. We discuss this process more fully next.

Discretizing The Workspace of a Manipulator

To track how well a manipulator performs at various regions of its reachable workspace, we must first discretize that space appropriately. To do this, we first note that the homogeneous transform that marks an end effector's pose with respect to a world frame $g_{we_n}(\bar{\theta})$ can be broken

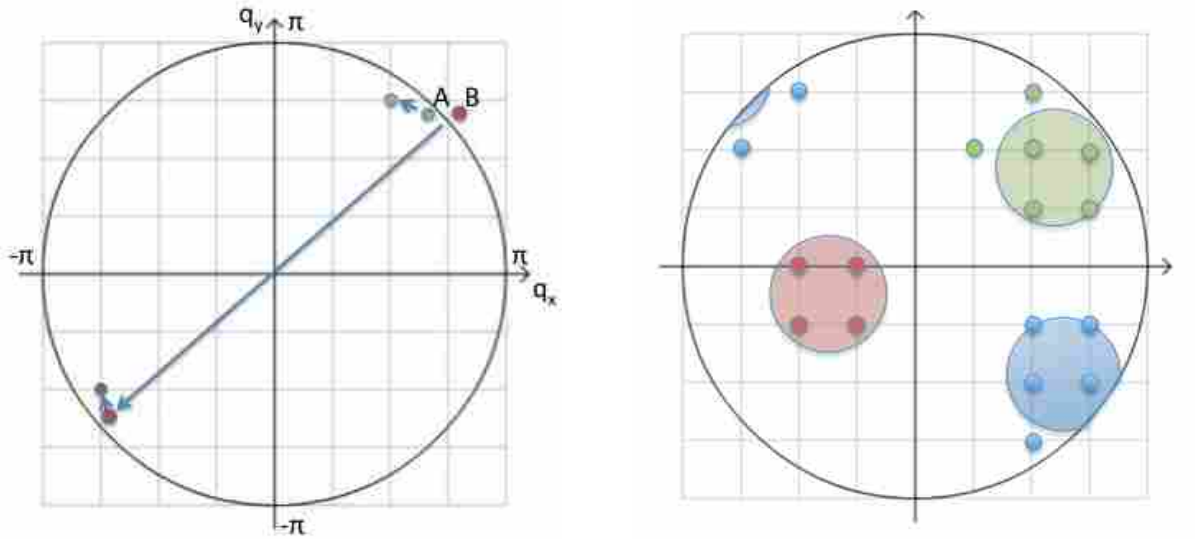
down into three positional components x, y, and z and three orientation components q_x, q_y, and q_z (as expressed in axis-angle coordinates [2]),

$$g_{WE} = f(x, y, z, q_x, q_y, q_z) \tag{3.1}$$

This six dimensional representation of an end effector pose provides a natural basis from which we may discretize the robot's dexterous workspace. We accordingly discretize a six dimensional space into a rectangular grid of size $M^3 \times N^3$, where M^3 is a 3D tensor representing discretization of translation in Cartesian space, and N^3 is a 3D tensor representing an angular discretization present at each discrete Cartesian location. Each time a new configuration is generated, the twist components of its end effector pose are computed and rounded to the nearest point in this grid. Subsequent computed metrics are then computed and stored at this discretization point.

One weakness we have encountered in pursuing this method of discretization arises in trying to impose a rectangular grid on the space of orientations $SO(3)$. This space is a sphere that wraps around itself and does not naturally lend itself to rectangular segmentation. This is because every axis-angle orientation vector is only unique in magnitude up to a factor of $2\pi n$, where n is any integer. To remove this ambiguity within our discretization process, vectors with magnitude greater than π are first expressed instead as their equivalent vectors within a ball of radius π before they are discretized. This can be seen in Fig. 3.3a which depicts the cut plane $q_z = 0$ of $SO(3)$. Here an orientation vector B is expressed as an equivalent orientation within the sphere of radius π before it is discretized. In contrast, the orientation vector A is inside the ball and can instead be directly binned.

The true region of reachable orientations at a single cartesian location can now be represented as some region contained within the SO(3) ball. As we find more orientations reachable at this location in the workspace, we sample more orientations within this subregion, marking nearby discretized orientations as being found. With a coarse rectangular discretization, the relative location of this subregion with respect to the discretization (directly related to the reference frame from which orientations are expressed) affects how many discretized orientations will be found. We have found this effect to be especially pronounced when the orientable subregion intersects the boundary of SO(3). This is shown in Fig. 3.3b.



(a) SO(3) Ambiguity Resolution

(b) Effects of Discretization and Reference Frame

Figure 3.3: Left: If the magnitude of an orientation vector lies outside a ball of radius π, it is first converted into an equivalent rotation within the ball by subtracting 2π from its magnitude. Right: The number of orientations found at a particular point in the workspace of a robot depends on how the true region of reachable orientations lines up with the discretization imposed on SO(3). In this image the blue, green, and red subregions represent the space of orientations that could be found at a single point in the workspace of a robot, as measured from different reference frames. This shows that the reference frame from which rotations are reported directly impacts the number of discretized samples found.

We have found this latter effect to introduce undesirable asymmetries in the reported dexterous workspace of a robot. In Fig. 3.4a we show an example of this by visualizing a heat map of the percentage of discretized orientation vectors (as measured from a frame at the origin) reportedly found at the cut plane $x = 0$ of the workspace of a robot. This robot is mounted horizontally and is perfectly symmetrical about the plane $z = 1.5m$, leading us to expect a symmetric heat map of pose orientations about this plane.

However we find that at points where the end effector can point downwards and wrap around itself ($q_x \geq \pi$ and the true region of orientations intersects the SO(3) surface), an artificially high number of orientations are reported. To remedy this problem, we chose to discretize end effector position with respect to the world frame as before, but orientation with respect to the mount of the base, $g_{b_0 e_n}$. Doing so reduces the number of orientation vectors falling near the boundary of SO(3) significantly and eliminates asymmetries from our optimization.

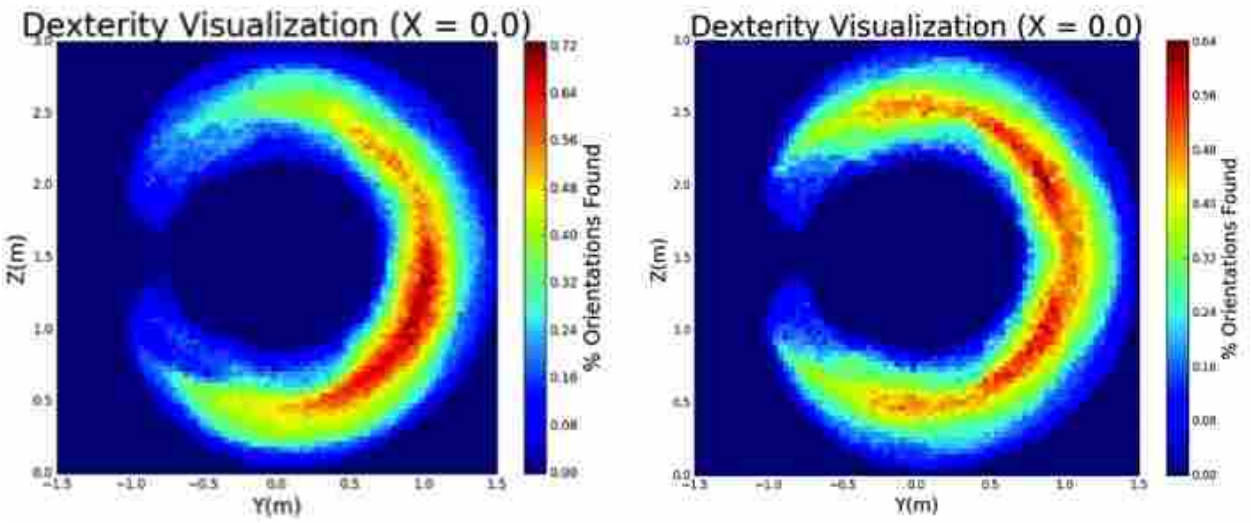


Figure 3.4: Imposing a rectangular grid over the space of orientations introduces asymmetries into the optimization (left). To correct for this, rotations are instead tracked with respect to the manipulator's base frame $g_{b_0 e_n}$ (right).

Workspace Sampling Techniques

Having defined our method of storing metrics within the reachable workspace of the manipulator, we next investigate methods of efficiently sampling the manipulator's controllable configuration space θ. As many designs must be evaluated within the evolutionary algorithm, we wish this process to be as fast as possible while providing a diverse set of manipulator configurations. In our work we have investigated three strategies: random forward kinematic (FK) sampling, inverse kinematics (IK), and hybrid FK-IK methods.

To perform random FK sampling, joint parameters are randomly sampled within their constraints and forward kinematics is performed using Eq. 2.3 (see Fig. 3.5). Because the forward kinematics routine is a simple series of matrix multiplications (see Eq. 2.3) this process can be done very quickly. The major drawback in doing this however, is that sampling becomes redundant as the algorithm progresses, thus leading to fewer new poses actually being found later in the algorithm.

We also tried using direct inverse kinematics (IK) to search the discretization space. In particular, three different methods of inverse kinematics were attempted: IK using a simple damped pseudo-inverse (DPI), IK using selectively damped least squares (SDLS) [41] and IK using quadratic programming (QP) [11]. Because each of these methods relied on expensive computations (e.g.,

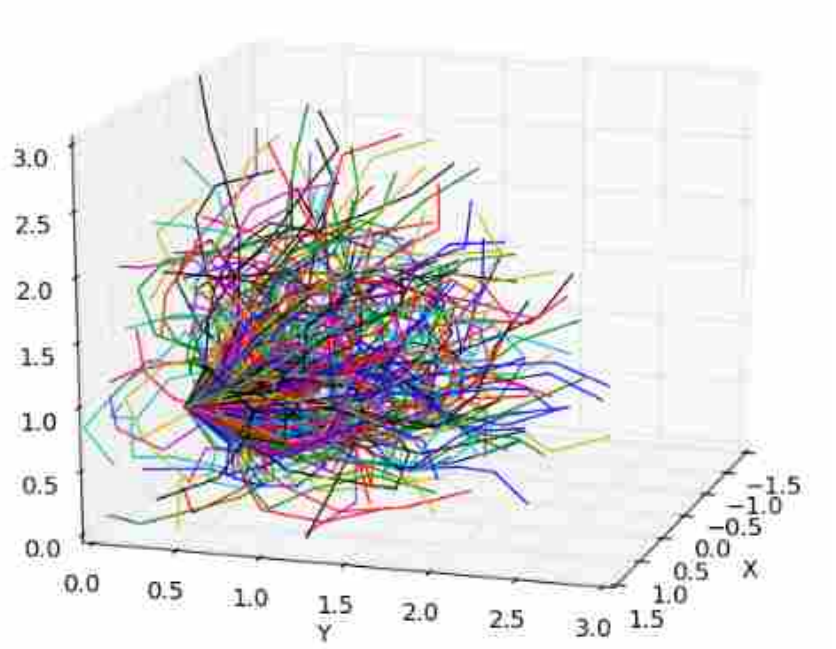


Figure 3.5: A single design mounted horizontally (+Y) is simulated and plotted at 500 random joint angle configurations. A typical design evaluation in our genetic algorithm would be simulated through two to four million random configurations and would be completed in 15-20 seconds.

updating the Jacobian matrix, performing singular value decomposition, and/or solving a QP problem), they were much slower in searching the manipulator's configuration space than fast FK sampling.

The fastest of these IK methods (in terms of identifying new poses) was by far IK using a damped pseudo inverse. In this case, a manipulator is initialized at a previously identified configuration and then driven to nearby, unidentified poses in the discretization by iteratively computing a step in the configuration space as,

$$\Delta\theta = J_B^T (J_B J_B^T + kI)^{-1} V_{g_{ent}} \tag{3.2}$$

Here, J_B represents the manipulator body Jacobian, $V_{g_{ent}}$ represents a desired deformation twist taken from the homogeneous transform between the end effector frame and a target pose, and k is taken to be a small damping constant ($k \approx .01$). As the Jacobian represents a linear mapping that holds only locally within the configuration space of a manipulator, only small steps in the direction of computed $\Delta\theta$ are taken, before a new Jacobian is computed and the process is iterated. This

is repeated until the orientation and position of the end effector align nicely with the target pose within a tolerance or a maximum number of allowable steps have been taken without convergence.[1]

While this method of IK is simple and fast, it still pales in comparison to the speed at which random fast FK sampling reveals new discretization poses. This is manifest in Fig. 3.6 in which random FK is performed on a 6DOF design for +80 seconds with discretization parameters $M = 50$ and $N = 7$. At every 100,000 random samples, each method (both random FK sampling and IK using a damped pseudo-inverse) is timed to see how long it takes to find a new pose within the discretization. Timing is averaged over 50 new discretization points found from each method. In this case IK is implemented to move from previously identified configurations to unfound, neighboring poses within the discretization with similar orientation.

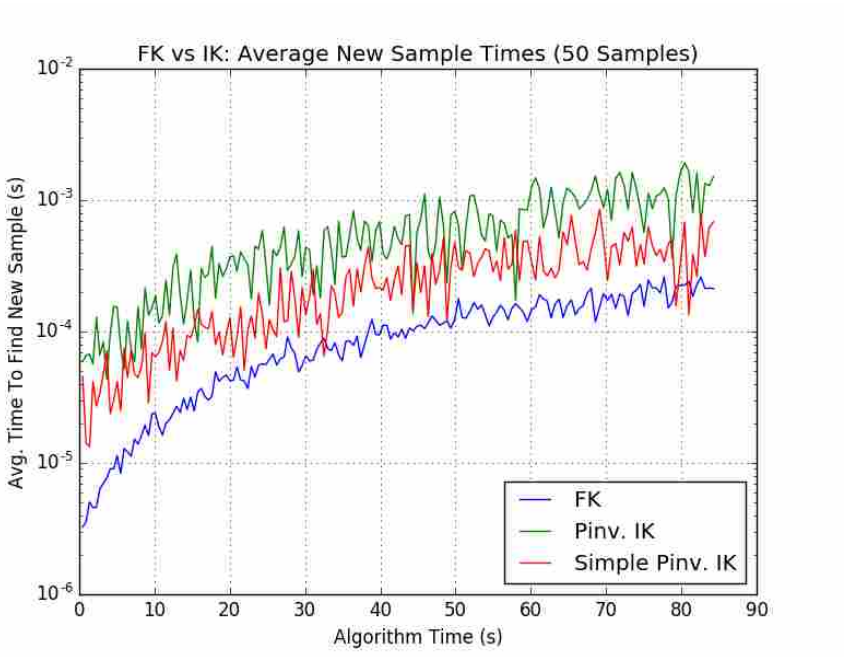


Figure 3.6: Randomly sampling joint angles and performing forward kinematics was found to be the fastest method of sampling a manipulator's workspace.

In Fig. 3.6 we see that simple FK sampling (blue) finds new samples about 10 times faster than the damped pseudo-inverse IK (green) throughout the entire characterization of the workspace. Code profiling tools revealed that the repeated computation of the Jacobian in the IK

[1]Error between the target frame and end effector frame is computed using a weighted norm of the differences in twist components of the two frames as expressed with respect to a world frame.

algorithm accounted for most of the time differential. In attempts to remedy this, a simple method of IK was implemented in which the Jacobian was not updated between IK steps (red) and only a few steps were allowed before the IK algorithm would terminate. This method somewhat improved the rate at which the IK method identified new samples, but still proved to be significantly inferior to random FK sampling.

The trends observed in Fig. 3.6 were confirmed over many different sets of evaluation parameters (e.g., max number of steps allowed to IK, step lengths, discretization sizes, etc.). Several hybrid FK-IK algorithms were also attempted. These hybrid methods begin by using FK, but resort to inverse kinematics when sampling becomes sufficiently redundant. Over longer periods of time these algorithms occasionally provided superior results over direct FK sampling but were difficult to tune. Additionally we observed that trends in design performance initially reported by simple FK sampling did not change with longer evaluations and/or extra IK sampling. We thus concluded that random sampling for relatively short periods of time was the most efficient means of comparing important trends for the purpose of design optimization.

Design Metrics

We now turn our attention to identifying appropriate metrics that can be computed at every new found pose, and then subsequently combined across the entire discretization at the end of the sampling routine to provide global measures of design optimality. For the inflatable robot we expect to test while mounted to the lunar rover K-REX at NASA Ames, we have identified two primary optimization objectives:

- That the manipulator be dexterous at all parts of its workspace
- That the manipulator be capable of lifting heavy loads near the ground with its end effector

A traditional metric used for dexterity is the volume of the six-dimensional ellipse which maps small joint velocities of a manipulator to twists at its end effector. This is noted by Yoshikawa et al. [36] as directly proportional to,

$$E = \sqrt{det(J_B J_B^T)} \tag{3.3}$$

This metric however requires the computation of the Jacobian at every new configuration found within the workspace of the robot. This operation is relatively expensive compared to a simple forward kinematics sampling, and should be avoided if possible. Instead, we favored using a brute force approach in which the number of unique poses γ_{sum} found within the discretization is taken directly as a metric for optimization.

In addition to being dexterous, we also preferred our robot to be capable of lifting heavy loads near the ground without undergoing excessive deflection. We observed in initial testing that our prototype arm experienced significant deflection when subjected to external loads. This deflection was primarily due to uncontrolled bending occurring at the manipulator's joints. To identify robots capable of lifting heavy loads, we needed to characterize this effect. To do this we assumed a rigid link model with flexible joints and build a linear model of joint stiffness as,

$$\begin{bmatrix} \Delta\tau_X \\ \Delta\tau_Y \\ \Delta\tau_Z \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} \Delta\theta_X \\ \Delta\theta_Y \\ \Delta\theta_Z \end{bmatrix} \tag{3.4}$$

This equation relates differential movements of the joint as expressed in twists coordinates $V_{g_{bt_i}}$ to external torques applied at the joint base (A coordinate frame is provided in Fig. 3.7 for reference).[2]

To find this joint stiffness model, we measured many different applied loads and tracked corresponding deformation twists of the joint transform g_{bt}.[3] Wrenches measured at the end effector were then expressed as equivalent wrenches directly about the joint base frame using Eq. 2.24. Resulting linear force components of transformed wrenches were dropped as they were not considered as deflection inducing.

With N_D samples, we can construct a $3xN_D$ matrix of horizontally stacked torques vectors τ_D and a $3xN_D$ matrix of corresponding twists deformation vectors $\Delta\theta_D$. We then perform a linear regression using a psuedo-inverse to find the 3x3 joint stiffness transformation K_{q_i} as,

$$K_{q_i} = \tau\Delta\theta_D^T(\Delta\theta_D\Delta\theta_D^T)^{-1} \tag{3.5}$$

[2]This model assumes only small deflections at each joint.

[3]For stiffness testing purposes, the joint arc length parameter h was assumed to be zero. The joint transformation g_{bt} then simplifies to a double pin joint model with an added rotational degree of freedom at its base. Measured deflection parameters $\Delta\theta_X$, $\Delta\theta_Y$, $\Delta\theta_Z$ are then assumed to be approximations for joint deflection parameters Δu, Δv, $\Delta\psi$ respectively.

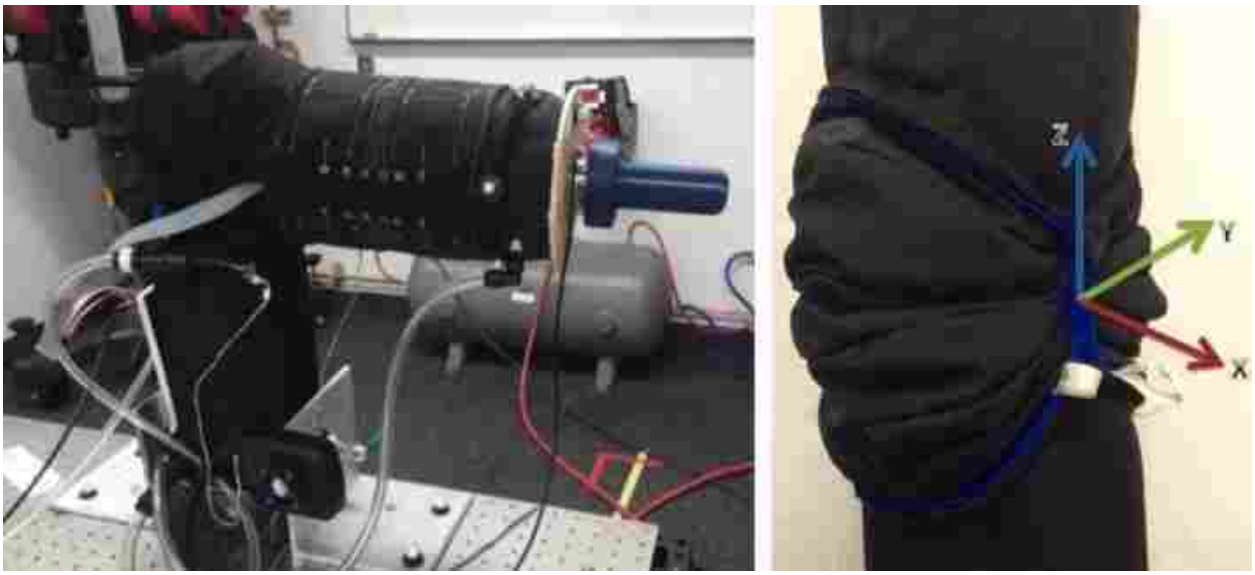


Figure 3.7: Left: An ATI force-torque sensor was attached to the end of a single inflatable joint and deflections were measured using a motion capture system. Right: A coordinate frame for a single joint is shown. Actuation generates joint torque about the x-axis.

In experimentation we found the joint stiffness model K_{q_i} to be approximately diagonal with significant deviations induced by varying bladder pressures and joint angle. As the joint hardware is still under development, we assumed it less important to do a full classification of this model and instead resorted to using a simplified joint stiffness model of diagonal structure. In future work we recommend extending stiffness models to be functions of both joint position and pressure as the joint hardware design becomes more developed.

For our purposes, we use the same diagonal joint model K_{q_i} for each joint and assume it to be independent of bladder pressures and configuration. Joint stiffness models are combined with the manipulator Jacobian to give a 6x6 equivalent end effector stiffness matrix,[4]

$$K_H = (J_H K_G^{-1} J_H^T)^{-1} \tag{3.6}$$

[4]To compute the end effector stiffness matrix as expressed in the body or spatial frames, use the body or spatial Jacobians respectively.

where

$$K_G = \begin{bmatrix} K_{q_1} & 0 & \dots & 0 \\ 0 & K_{q_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K_{q_n} \end{bmatrix} \tag{3.7}$$

The stiffness matrix given in Eq. 3.6 provides a linear mapping between external wrenches W_H applied at the manipulator's hybrid frame and resulting deformation twists V of that frame.

$$W_H = K_H V \tag{3.8}$$

We use this expression to obtain the max downward force that a particular configuration can support at its end effector without deflecting outside of a defined deformation tolerance V_{max}. By expressing the contributions of gravity and external loading forces to deformation at the end effector separately we have,

$$\left\| V^L + V^G \right\| \leq V_{max} \tag{3.9}$$

Using Eq. 3.8 to express resulting end effector deflection in terms of a corresponding applied wrench W_H^L, and summing known gravity wrenches in the end effector's hybrid frame as W_H^G using appropriate adjoint transformations (see Eq. 2.24), we can restate this equation as,[5]

$$\left\| K_H^{-1} W_H^L + K_H^{-1} W_H^G \right\| \leq V_{max} \tag{3.10}$$

$$W_H^L = \alpha_L u_L$$

Given a known loading direction unit vector u_L for W_H^L, Eq. 3.10 gives a quadratic expression in the loading force magnitude α_L, the roots of which represent the range of forces that maintain acceptable deflections. The maximum load supportable under the deflection limit is then taken to be the maximum root α_F of this expression with $u_L = [0,0,-1,0,0,0]$. As an example, Figure 3.8 shows a nominal configuration (black), a deflected configuration due to the manipula-

[5]For simplicity, orientation components are ignored and a standard two norm is used representing a spherical deflection limit in Cartesian space.

tor's weight alone (blue), and a deflected configuration due to the manipulator's weight as well as a critical force α_F applied downwards at the end effector (red).

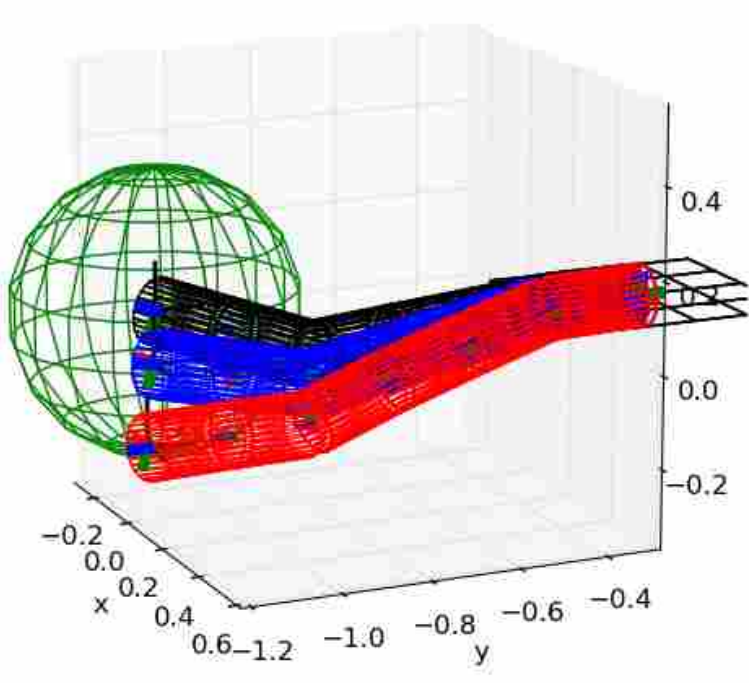


Figure 3.8: At each configuration a critical tolerance $V_{max} = .15m$ (shown in green) is chosen from the nominal end-effector position (shown in black). A critical load α_F is then computed that causes deflection (shown in red) out to this tolerance. The deflection of the arm from its weight alone is shown in blue for reference (control is used to prevent deflection in actuated degrees of freedom).

Our actuators allow us control about the x-axis (see Fig. 3.7 for reference) of each joint within a torque range of τ_- and τ_+. As long as loading conditions generate joint torques within this controllable range, we assume we can completely compensate for deflections in this degree of freedom with active control.[6] To factor in the capacity of control to compensate in this way, we can assume infinite stiffness in actuated degrees of freedom. This can be done by eliminating rows and columns of J_H and K_G associated with actuated degrees of freedom in Eq. 3.6.

Making this assumption we can solve for the max loading force α_F as before, and then cap it by the minimum load α_{limit} that would generate at least one unsustainable torque at a joint. Doing this ensures that our max sustainable load α always stays within a range compatible with our controllable degrees of freedom. Joint torques τ are related to loads applied at the end effector

[6]We neglect additional torque introduced by the dynamics of the system and assume the robot moves slowly.

as,

$$\tau_{=} J_H^T W_H^L + J_H^T W_H^G \tag{3.11}$$

$$\tau_{=} J_H^T u_L \alpha_L + \tau^G$$

To remain in an acceptable region of control, α_L must be set so every joint torque τ_i remains within its respective, controllable range. Solving for the max allowable force α_{limit_i} that can be applied within each of these ranges, and taking the minimum of the result gives,

$$\alpha_{limit} = \min_i(\max(\frac{\tau_{i_-} - \tau_{i_G}}{J_{H_i}^T u_L}, \frac{\tau_{i_+} - \tau_{i_G}}{J_{H_i}^T u_L})) \tag{3.12}$$

where J_{H_i} represents the column of the manipulator Jacobian corresponding to the ith actuated degree of freedom and τ_{i_G} represents the torque applied torque from gravity. If a joint cannot support torque induced by gravity (let alone any additional loading), we set the entire expression equal to zero.

The supportable force at the end effector that maintains acceptable deflection while staying within actuator torque limits is then,

$$\alpha = \min(\alpha_F, \alpha_{limit}) \tag{3.13}$$

This metric is computed for every configuration we find within 20 cm of the ground during the forward kinematic sampling routine. If multiple configurations converged on the same end effector grid pose, the maximum applicable load is taken.

Termination and Metrics Scoring

The forward kinematic sampling routine is terminated when a high percentage (+99%) of configuration samples taken generate poses previously found within the discretization. At this point we assume the manipulator has been sufficiently simulated through its workspace. The total number of end-effector poses found within the discretization is then counted, and the sustainable loading force α is summed across the discretization to provide separate measures of dexterity and load sustainability respectively.

In summing the loading force variable α, it was found that occasionally singular or near-singular configurations would generate very high sustainable forces that became dominating within the optimization. To overcome this affect we cap the maximum force α contributed by any single discretization point by a constant α_{max}. In this sense we care about finding the design that can support loads over large regions of its workspace.

Maximin Optimization

Each design's scores are now evaluated against others in the multi-objective maximin fitness function presented in [30]. Metric scores between designs in a generation are scaled to be between zero and one before evaluating the maximin function (e.g., the most dexterous design in a generation will receive a score of one for that metric).

$$\begin{aligned} fitness_i &= 1 - \max_{j \neq i} \left(\min_k \left(f_{ki} - f_{kj} \right) \right) \\ fitness_i &= \text{maximin fitness score of } i\text{th design} \\ f_{ki} &= k\text{th objective score at } i\text{th design} \\ f_{kj} &= k\text{th objective score at } j\text{th design} \end{aligned} \tag{3.14}$$

This function evaluates how optimal a particular design is in maximizing its scores in relation to other designs in that generation. Its formulation encourages both diversity and optimality as it selects designs to be propagated. By continually evolving designs and choosing those that score well in this way, we achieve a final set of optimal designs that allows us to explore the inherent trade-offs in the design space.

3.3 Results

3.3.1 Optimizing an Inflatable Robot

One hundred designs were randomly generated and evaluated for dexterity and stiffness performance near the ground. High performing designs were propagated through the genetic algorithm as discussed in Sec. 3.2.1 and this cycle repeated for 50 generations. Design evaluations

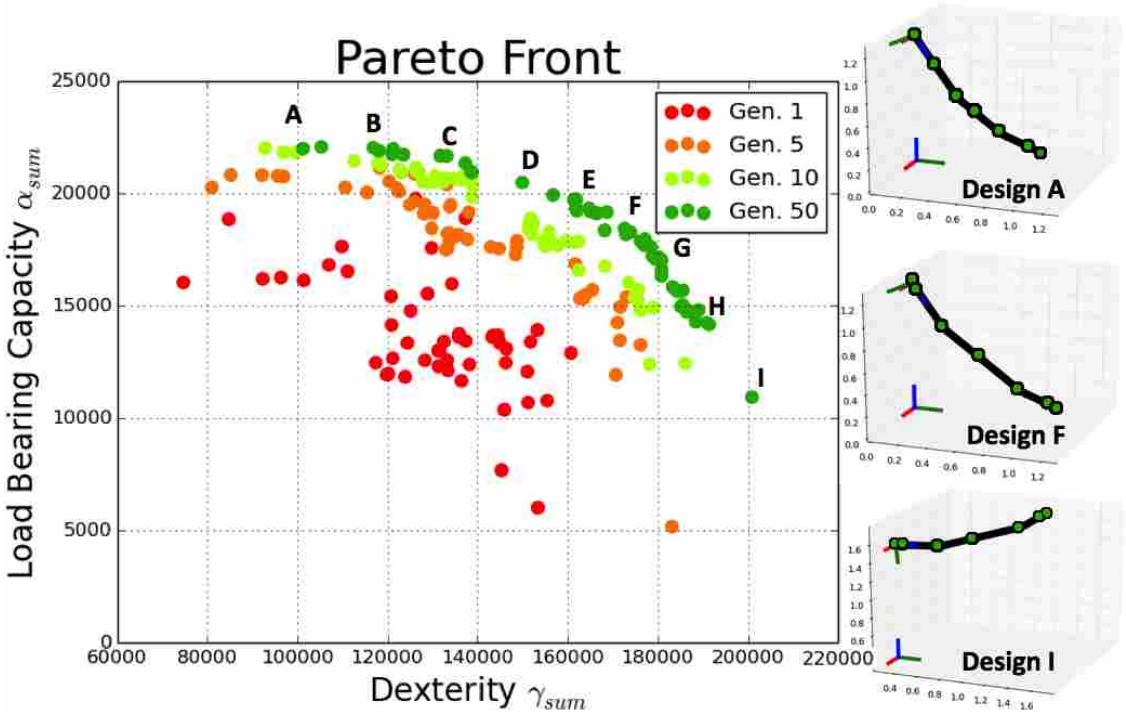


Figure 3.9: A Pareto front that characterizes fundamental trade-offs inherent in the design space of an inflatable manipulator.

were performed in parallel on eight cores of a 3.4 GHz Intel Core i7-4770 processor and the entire optimization took approximately five hours to complete. Significant speedups can be obtained using more processing cores (up to the number of designs evaluated each generation).

Processing time was highly dependent on desired discretization resolution. In this example we discretized a $27m^3$ space to the side of K-REX in a grid of 31x31x31 points representing possible end effector positions in 10 cm increments. Orientations were binned within a 5x5x5 grid at each positional grid point. Each design evaluation sampled between two to four million configurations (depending on design dexterity) and took about 15-20 seconds to run. For comparison, a lower resolution optimization running with 21x21x21 positional points and 3x3x3 orientation points was completed under an hour. This lower resolution optimization gave similar results to that of the higher resolution.

The resulting Pareto front shown in Fig. 3.9 visualizes how the optimization evolves designs from a set of randomly generated designs (red) to a set of Pareto optimal designs (dark green). The final generation of designs characterize the optimal region of the design space and

highlight fundamental trade-offs the designer must make. A table listing the parameters of a subset of optimal designs in the estimated Pareto front is shown in Table 3.1.

Table 3.1: Optimization Results

Des.	L_1	L_2	L_3	L_4	L_5	L_6	L_a	b_θ	b_Z
A	.23	.27	.13	.18	.19	.04	1.40	-60	1.19
B	.25	.27	.11	.19	.21	.06	1.45	-60	1.24
C	.20	.33	.15	.22	.19	.03	1.47	-60	1.26
D	.13	.34	.21	.24	.20	.00	1.48	-60	1.27
E	.07	.35	.25	.27	.19	.00	1.49	-60	1.24
F	.00	.30	.31	.34	.19	.00	1.50	-60	1.18
G	.00	.29	.27	.38	.19	.00	1.49	-54	1.27
H	.01	.26	.31	.36	.19	.00	1.49	-35	1.22
I	.00	.27	.28	.39	.19	.01	1.50	4	1.32

[a] Length measurements are given in meters, and the base angle is given in degrees.
[b] Zero link lengths denote collocated joints (90° offset included). In the case of the last link being zero, the last joint is directly attached to end effector tooling.

The tabulated results help us understand several key issues regarding the design space of these inflatable manipulators. We see that in general, stiffer arms (see design A for example) have parameters that reduce gravity torque loads near their base. They afford longer link lengths near the base to accommodate heavy valve hardware. Further, they have slightly shorter total arm spans than other, more dexterous arms to save on weight. By reducing gravity torque loads on critical joints near the base, these joints are then allowed to support heavier loads applied at the end effector. The result is a stiffer, though slightly less-dexterous arm.

On the other hand, dexterous arms like design I are significantly longer than stiffer arms. They have shorter link lengths at the beginning and end of the robot, offering shoulder and wrist-like structures to the robot that improve the dexterous workspace of the arm. Further, they are mounted slightly higher from the ground and point more horizontally, granting them a larger, ground-free workspace in which they can touch many points within the discretization. A 3D visualization of the reachable workspace as measured by the dexterity metric given in this paper is provided in low resolution (as seen and used by the optimization) at `https://youtu.be/bsk7NbaCb5E` and at a higher resolution for visualization and validation after the optimization is complete at `https://youtu.be/pqXF-ymJidk`.

3.3.2 Optimizing a Manipulator with Blow-Molded Joints

This same optimization routine was used in order to optimize the design of a blow-molded arm similar to that shown in Fig. 2.2. In this case, three blow molded joints (2DOF each) are connected by two rigid links. These links have fixed bends in them that our optimization can determine of up to 45°. Appropriate joint angle and arm length constraints are applied to describe the natural limitations of the system. Link lengths and fixed bend angles are included as design variables within the optimization. Additionally, as before, the mounting height b_Z and a tilt angle b_θ are also included as design variables in the optimization.

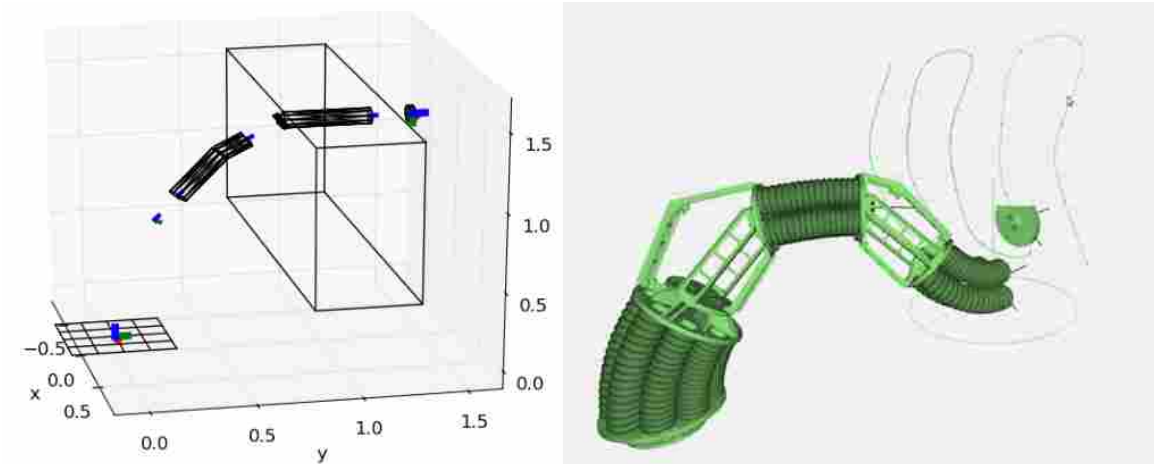


Figure 3.10: A blow-molded jointed manipulator is optimized in order to paint a wall. Left: A Python visualization shows the workspace of interest and an optimized design (with joints removed). Right: A screen shot of a similar robot performing painting motion.

The objective in this case was to optimize robot geometry in order to allow the robot to paint a wall. We accordingly used only the dexterity metric previously discussed and counted only poses with position lying within a 2x.5x1 meter box and with orientation within 60° of being perpendicular to the wall (+y axis in Fig. 3.10). This score is taken as the only metric of interest and used directly to compare designs within the optimization routine. Crossover and mutation strategies are analogous to those discussed earlier. An identified optimal design (joints not depicted) is shown relative to the workspace of interest in Fig. 3.10. This design could find almost all of the

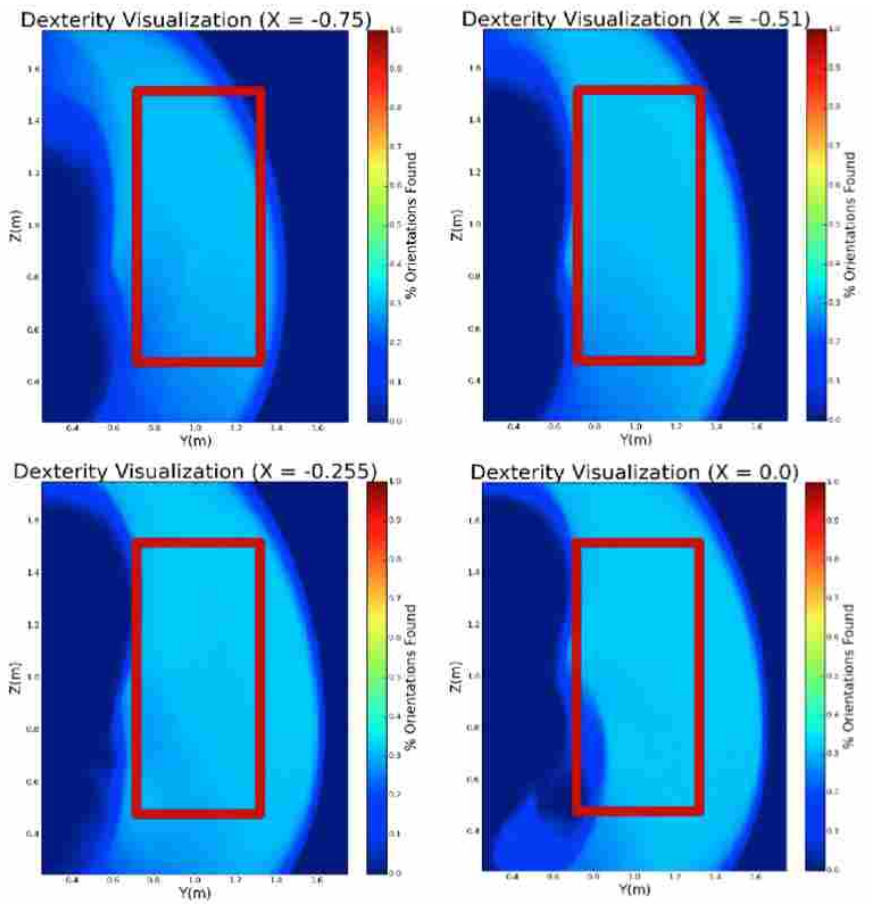


Figure 3.11: A sequence of yz-cut planes (left to right: X = -.75m, X = -.5m, X = -.25m, X = 0m) showing the percent of pose orientations counted towards the dexterity metric in those regions. The red box represents the workspace of interest at each cut plane. The dexterous workspace is symmetric about X = 0m.

orientations perpendicular to the wall within the desired region. This is shown in visualizations of cut planes of its dexterous workspace in Fig. 3.11.

3.4 Discussion

The genetic algorithm presented in this work provides a robust, scalable framework that can be used to optimize a host of robotic technologies. It is especially applicable to systems that can be quickly manufactured and that feature many degrees of freedom in their design space, such as the inflatable arm presented. The metrics provided herein are offered as examples for those designing similar, soft-robot systems. Other systems will require appropriate metrics reflecting desirable traits relevant to their application.

CHAPTER 4. PATH PLANNING

In this section we discuss our method for performing motion planning. As in the last chapter we will begin this chapter with a discussion on previous approaches used to solve this problem. We then discuss our method of planning and present motivating examples.

4.1 Previous Work

Sampling-based motion planners such as RRT or PRM [42–48] are probabilistically complete and very effective for many problems. However, they require starting and ending robot states, are not well suited to following continuous end-effector trajectories, typically require a path smoothing step, and do not scale well to high degree-of-freedom systems.

Local inverse kinematic methods [10, 11] are effective methods for generating configurations of a robot given an end-effector goal, and these algorithms could be combined with the above path search or trajectory optimization algorithms to yield start and end configurations with a path linking them. Similarly Berenson, Kuffner and Choset [49] use evolutionary algorithms to find start and end configurations which are then connected with an RRT-based planner. However, failing to consider the path between objects when choosing start or end configurations can make the problem more difficult than necessary or impossible.

Trajectory optimization algorithms such as CHOMP [50], STOMP [51] or TrajOpt [52] are able to generate a trajectory for an initial guess (which may be infeasible or in collision), and scale well to high-dimensional problems. Each utilizes a distinct optimization method to minimize an objective function comprised of path smoothness and collision avoidance terms. Our formulation most closely follows TrajOpt in which the original cost function is iteratively approximated and a path update is identified using a quadratic programming solver. Our method differs however, in that we address the problem of planning directly in the task space of the manipulator with a predetermined end effector target path specified. Additionally we introduce functionality in our

formulation that enables a fixed base position of the robot to be optimized simultaneously with path motion.

Several works [53–55] consider planning problems such as opening a door with a mobile robot. However, the approach is feasible because they plan in the low-dimensional space of the robot's base position and orientation, allowing them to discretize the space. This does not scale well to the higher degree-of-freedom systems that we wish to consider.

This work is a direct extension of the authors in [11], extending it from a single end effector goal to an arbitrary number of goals. It utilizes a hierarchical weighting scheme to handle competing objectives, similar to that given in [56], but applied over an entire trajectory. This work is also related to general work on redundancy resolution. However, most other past work on redundancy resolution is focused on velocity, acceleration, or torque formulations without moving the end-effector or while achieving a secondary constraint or task instead of reaching a pre-defined trajectory with smooth motion (see chapter 11 in [57] for a summary of these approaches).

4.2 Methods

In this section we provide a detailed list of inputs that the planner requires. We also discuss the details of how the algorithm uses kinematics to optimally move arms towards their final targets. Finally, we discuss objective weighting strategies, termination criterion and our process of recursive path refinement for continuous and smooth motion.

4.2.1 Inputs

Our planner requires the following inputs:

- **Arm Kinematic Model:** The planner requires a defined kinematic model with availability of body Jacobian matrices J_{B_i} which linearly map differential movements in joint space to corresponding twists at the ith end effector (in the ith end effector's frame of reference). Similarly, body Jacobian matrices $J^l_{B_i}$ at the center of mass of each link l are required. An efficient means of computing forward kinematics is also assumed, that is, given a configuration θ_i, we can find a homogeneous transform at any frame of interest attached to the

manipulator. While we used the kinematic model introduced in Ch. 2, our path planning framework is general enough to encompass any kinematic model with these simple features.

- **Target Path Parameterization:** A function $\chi(t)$ representing a parameterized target path must be available to the path planner. This function maps a parameter t, ranging from 0 to 1 to a desired end effector pose $T_t \in SE(3)$ on the desired path. A continuous equation for computing these targets is important to the path refinement routine.

- **Constraints:** The inverse kinematics routine allows for constraints that represent the fundamental limits of the manipulator including joint stops ($\theta_{min.}$, $\theta_{max.}$). Custom constraints including maximum allowable parameter deviations $abs(\Delta\theta_{max})$ at any single iteration in the optimization are also included. Finally, a set of geometries G representing potential collisions the manipulator could make with the world must be known to the path planner. These collision geometries are included as optional inputs.

4.2.2 Quadratic Programming Problem Formulation

The inverse kinematics problem for a single manipulator is outlined in [11] where a local search direction in joint space is identified by solving a quadratic programming (QP) problem that stems from the optimization problem of the form,[1]

$$\begin{aligned} \underset{\Delta\theta}{\text{argmin}} \quad & \left\| V_{WE}^{B} - \tilde{V}_{WE}^{B} \right\|_{2,P\chi}^{2} + \|\Delta\theta\|_{2,P\theta}^{2} \\ \text{noting that} \quad & V_{WE}^{B} \approx J_B(\theta)\Delta\theta \end{aligned} \tag{4.1}$$

where V_{WE}^{B} is a deformation twist in the body frame associated with a small change in joint variables $\Delta\theta$, and $\tilde{V}_{WE}^{B}$ represents a desired deformation twist computed using the matrix logarithm [2]. The matrix norm P_χ is used to weight units of position and orientation ($\|x\|_{2,P} = \sqrt{x^T P x}$) according to the user's objectives. The second term $\|\Delta\theta\|_{2,P\theta}^{2}$ is included as a damping term for algorithm stability purposes.

[1] The 6x1 deformation twist vector V represents screw motion along a constant twist ξ.

The result of solving a QP of this form is an optimal, local search direction in joint space that drives the manipulator towards its assigned target pose. By iteratively solving, stepping and then updating the QP problem, an arm is pushed to a desired pose.

Our approach to path planning follows this formulation, and extends the problem to include k copies of the same manipulator, each representing the manipulator's configuration θ_i at a specific instance in time. We also introduce a shared base M with degrees of freedom θ_M. This base represents a stationary mount or parking position from which the manipulator will carry out motion. We define $J^M_{B_i} \; \forall \; i = 0, 1, 2, \ldots, k$ as the body Jacobian matrices mapping movement at the base (as part of the progression of the optimization only) to corresponding twists at the *ith* configuration's end effector. Each end effector is assigned a separate target along the parameterized path given as $\chi(\frac{i}{k})$ from which a desired deformation twist $\tilde{V}^{B_i}_{WE}$ is computed using the matrix logarithm as before. The optimization formulation then becomes,

$$\begin{aligned} &\underset{\Delta\theta}{\text{argmin}} \qquad \left\|\bar{V} - \tilde{V}\right\|^2_{2,P_{\chi_G}} + \|\Delta\theta\|^2_{2,P_{\theta_G}} \\ &\text{noting that} \quad \bar{V} \approx J^G_B(\theta)\Delta\theta \end{aligned} \tag{4.2}$$

where,

$$J^G_B = \begin{bmatrix} J_{B_0} & 0 & \ldots & 0 & J^M_{B_0} \\ 0 & J_{B_1} & \ldots & 0 & J^M_{B_1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & J_{B_k} & J^M_{B_k} \end{bmatrix} \tag{4.3}$$

$$\Delta\theta = \begin{bmatrix} \Delta\theta_0 \\ \Delta\theta_1 \\ \vdots \\ \Delta\theta_k \\ \Delta\theta_M \end{bmatrix} \bar{V} = \begin{bmatrix} V^{B_0}_{WE} \\ V^{B_1}_{WE} \\ \vdots \\ V^{B_k}_{WE} \end{bmatrix} \tilde{V} = \begin{bmatrix} \tilde{V}^{B_0}_{WE} \\ \tilde{V}^{B_1}_{WE} \\ \vdots \\ \tilde{V}^{B_k}_{WE} \end{bmatrix}. \tag{4.4}$$

P_{χ_G} is defined as,

$$P_{\chi_G} = \begin{bmatrix} P_{\chi_0} & 0 & \dots & 0 \\ 0 & P_{\chi_1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_{\chi_k} \end{bmatrix} \tag{4.5}$$

and can be strategically adjusted to achieve objectives required by the application. If orientation is not important during a particular task, for example, weights in P_{χ_G} corresponding to orientation are set to zero. If the particular route a manipulator takes between two end-effector targets is unimportant, $P_{\chi_1} = P_{\chi_2} = P_{\chi_{k-1}} = 0$.

Equation 4.2 provides a framework whereby path planning can be carried out using inverse kinematics. By rearranging this expression into a more compact form and adding linear constraints representing joint stops we can achieve, [2]

$$\begin{aligned} &\underset{\Delta\theta}{\text{argmin}} && \Delta\theta^T P \Delta\theta + L^T \Delta\theta \\ &\text{subject to} && A\Delta\theta \leq b \end{aligned} \tag{4.6}$$

where,

$$P = P_\chi + P_{\theta_G} \tag{4.7}$$

$$P_\chi = J_B^{G^T} P_{\chi_G} J_B^G \tag{4.8}$$

$$L = L_\chi = -2 J_B^{G^T} P_{\chi_G} \tilde{V} \tag{4.9}$$

4.2.3 Line Search

The result of solving (4.6) is a search direction in θ that optimally pushes the manipulators towards their respective targets. We step in this direction a length of α_{step} which we determine by performing a line search in θ, comparing forward simulated poses with their respective targets. The α_{step} corresponding to the set of simulated configurations closest to their respective targets

[2] For details on how this expression can be obtained, see section 3.1 in [11]. Together, A and b represent constraints as convex polytopes. Here, we iteratively redefine b to clamp $\Delta\theta_{ij}$ to keep movement within its joint stops, as well as within a maximum allowable deviation $abs(\Delta\theta_{ij}) \leq \Delta\theta_{max}$.

is accepted as the optimal step size and the manipulators' configurations are updated accordingly. This process is repeated as the configurations branch towards their desired poses along the path.

4.2.4 Secondary Objectives

Smoothness

In its current form, Eq. 4.6 does not penalize adjacent configurations for being far from each other in joint space, a condition necessary for globally smooth motion. This can be achieved by adding the penalty term,

$$\kappa \sum_{i=0}^{k-1} \sum_{j=0}^{n} \left\| (\theta_{ij} - \theta_{(i+1)j}) \right\|_2^2 \tag{4.10}$$

to the minimization in Eq. 4.6 where θ_{ij} refers to the jth joint on the ith virtual manipulator. By approximating a joint parameter at iteration p as $\theta_{ij_p} \approx \theta_{ij_{p-1.}} + \Delta\theta_{ij}$, expanding the norm and eliminating terms that do not affect the minimization problem, we can include this term in Eq. 4.6 in a convenient form as,

$$\Delta\theta^T P_\kappa(\kappa)\Delta\theta + L_\kappa^T(\theta_{p-1})\Delta\theta \tag{4.11}$$

where,

$$\Delta\theta^T P_\kappa(\kappa)\Delta\theta = \kappa \sum_{i=0}^{k-1} \sum_{j=0}^{n} (\Delta\theta_{ij} - \Delta\theta_{(i+1)j})^2$$

$$L_\kappa^T(\theta_{p-1})\Delta\theta = \kappa \sum_{i=0}^{k-1} \sum_{j=0}^{n} 2(\theta_{ij_{p-1}} - \theta_{(i+1)j_{p-1}})\Delta\theta_{ij} + 2(\theta_{(i+1)j_{p-1}} - \theta_{ij_{p-1}})\Delta\theta_{(i+1)j}$$

Manipulability

Traveling in a manipulator's dexterous workspace is also desirable as a secondary objective in order to avoid singular configurations. Yoshikawa presented a measure of manipulability in [36]

that is proportional to the volume of the manipulability ellipsoid given as,

$$E = \sqrt{det(J_B J_B^T)} \tag{4.12}$$

We penalize joint motions that decrease this term by including a linear term weighted by scalar γ,

$$L_\gamma^T \Delta\theta = -\frac{\gamma}{k} \sum_{i=0}^{k} \sum_{j=0}^{n} \frac{d(\sqrt{det(J^{B_i} J^{B_i T})})}{d\theta_{ij}} \Delta\theta_{ij} \tag{4.13}$$

The derivative in this term is taken numerically. We also include a second, less-important term weighted by β. This term encourages joints to remain near their center value (and thus away from joint stops),

$$\beta \sum_{i=0}^{k} \sum_{j=0}^{n} \left\| (\theta_{ij} - \theta_{ij.ctr.}) \right\|_2^2 \tag{4.14}$$

As before, by ignoring terms independent of $\Delta\theta$, using the previously introduced approximation of θ_{ij} and expanding, this term can be included in Eq. 4.6 as,

$$\Delta\theta^T P_\beta(\beta) \Delta\theta + L_\beta^T(\theta_{p-1}) \Delta\theta \tag{4.15}$$

where,

$$\Delta\theta^T P_\beta(\beta) \Delta\theta = \beta \sum_{i=0}^{k} \sum_{j=0}^{n} {\Delta\theta_{ij}}^2$$

$$L_\beta^T(\theta_{p-1}) \Delta\theta = \beta \sum_{i=0}^{k} \sum_{j=0}^{n} 2(\theta_{ij_{p-1}} - \theta_{ij_{ctr.}}) \Delta\theta_{ij}$$

Collision Avoidance

An extra term can be added to virtually push configurations away from potential collisions in G. This implementation only considered simple geometries (points, lines, and planes) but any collision detection package which gives collision distances and normals could be incorporated (for example Bullet Physics [58], which uses the Gilbert-Johnson-Keerthi [59], algorithm).

For each link l that comes within a distance ε of a constraint, a secondary target is defined for that link normal to the constraint that pushes the link away from the constraint. A desired deformation twist $\tilde{V}_{WL_l}^{B_i}$ is then taken between the link frame and this target, and incorporated into the objective function as,

$$c\sum_{l=0}^{m}\left(\frac{\varepsilon-d}{\varepsilon}\right)^2\left\|V_{WL_l}^{B_i}-\tilde{V}_{WL_l}^{B_i}\right\|_{2,P_{\chi_c}} \tag{4.16}$$

where d is the distance between the link and the constraint and c is a scalar weight. Entries in P_{χ_c} weighting the orientation components of the desired deformation twist are set to zero as they are unimportant in this case. The link twist $V_{WL_l}^{B_i}$ can be expressed as a product of the respective link's body Jacobian expressed at its center of mass, and joint velocities up to the respective link as,

$$V_{WL_l}^{B_i}=J_{B_i}^{l}\Delta\theta_i \tag{4.17}$$

Columns in $J_{B_i}^{l}$ corresponding to joints distal to link l are set to zero. As with Eq. 4.2, we can express these collision terms in a quadratic form, scaled by a weight c. With m links near collisions we have,

$$\sum_{l=0}^{m}\Delta\theta^T P_{c_l}(c)\Delta\theta+L(c)_{c_l}^{T}\Delta\theta \tag{4.18}$$

where P_{cl} and $L_{c_l}^{T}$ are analogous to the expressions given in Eqs. 4.8 and 4.9.

With objectives now defined in a coherent form, we can express the optimal path planning QP formulation as given in Eq. 4.6 with,

$$P=P_\chi+P_\kappa+P_\beta+P_\theta+\sum_{l=0}^{m}P_{c_l} \tag{4.19}$$

$$L=L_\chi+L_\kappa+L_\beta+L_\gamma+L_\theta+\sum_{l=0}^{m}L_{c_l} \tag{4.20}$$

4.2.5 Weighting strategies

The weighting of terms κ, β, and γ are critical to algorithm performance and must gradually decay from their initial values to near zero (letting P_χ dominate in Eq. 4.6) if exactly following a defined end-effector target path is desired. To do this we measure the sum of errors computed

between the arm configurations and their respective targets at each time step p given as,

$$e_p = \tilde{V}^T P_{\chi_G} \tilde{V} \tag{4.21}$$

We choose to scale weights by a factor $0 \leq \lambda \leq 1$ when the improvement rate $e_{p-1} - e_p$ falls below a user-specified value Ω. At this point, proportionally more weight is shifted to the configuration target objectives in the optimization routine, and the arms subsequently reach closer towards the desired path.[3]

The collision weight c is held constant throughout the algorithm, consistently pushing the configurations away from defined boundaries. To allow the algorithm to reach near these constraint boundaries, we slowly reduce the critical tolerance ε that triggers these terms to a fixed ε_{min} during the course of the path planning algorithm.

4.2.6 Termination Criterion

The algorithm terminates when improvement in the objective function falls below a critical tolerance Φ and when secondary weights all fall below critical thresholds. We have observed in cases where the manipulator is not able to reach every target pose on the path, that it is generally a good idea to keep secondary weights slightly positive near the end of the algorithm to maintain desirable path properties including smoothness.

4.2.7 Refinement Strategies

Running this path planning algorithm results in a set of k configurations that reach out to the chosen path. The QP problem has computational complexity of order $O(k^2)$. However, the planning problem is quite sparse, and by specifying this sparsity in the QP solver we have seen experimental run times that are only slightly superlinear, approximately $O(k^{1.1})$.

When a finely discretized path is desired, we recommend running this planner recursively between identified configurations. This can be done by first solving with moderate k (e.g., k=10 or

[3]This strategy of weight scaling is only applicable when it is essential that end-effector poses end up at their respective targets. In cases where end effectors are allowed to deviate from the path (including trajectory optimization), we can choose a different weighting strategy (including P_{χ_i}) to meet other performance objectives.

Algorithm 1 Planner

procedure PLANNER
  $initialize(KinematicModel, M, G, \chi(t))$
  $\theta_f \leftarrow PlanPath(\theta_{init}, Refine=False)$
  for all $\theta_i, \theta_{i+1} \in \theta_f$ **do**
    $RefineBetween(\theta_i, \theta_{i+1})$
  end for
  $RecordConfig(\theta_k)$
end procedure

Algorithm 2 PlanPath

procedure PLANPATH($\theta_{init.}, Refine$)
  $InitializeArms(\theta_{init.})$
  $AssignTargets(X(t))$
  if $Refine$ **then**
    $FixBoundaryConfigs()$
    $FixBase()$
  end if
  while $True$ **do**
    $ComputeQPTerms()$
    $\Delta\theta \leftarrow SolveQP()$
    $\alpha_{step} \leftarrow LineSearch(\Delta\theta)$
    $UpdateArms(\alpha_{step}, \Delta\theta)$
    $e_p = e$
    $e \leftarrow ComputePoseError()$
    if $e_p - e < \Omega$ **and not** $WeightScalingComplete()$ **then**
      $RescaleWeights()$
    end if
    if $e < \Phi$ **and** $WeightScalingComplete()$ **then**
      $break$
    end if
  end while
  return θ_f
end procedure

Algorithm 3 RefineBetween

procedure REFINEBETWEEN(θ_1, θ_2)
  if $abs(t_{\theta_1} - t_{\theta_2}) > \zeta$ **then**
    $\theta_{init.} \leftarrow GenerateBranchBetween(\theta_1, \theta_2)$
    $\theta_f \leftarrow PlanPath(\theta_{init.}, Refine=True)$
    for all $\theta_i, \theta_{i+1} \in \theta_f$ **do**
      $RefineBetween(\theta_i, \theta_{i+1})$
    end for
  else
    $RecordConfig(\theta_1)$
  end if
end procedure

Figure 4.1: These algorithms summarize the path planning procedure.

20) and then running a sequence of subproblems, each in which a new branch of arms is defined between previously identified configurations. In doing so we fix the boundary configurations of subproblems to be the poses between which we are refining. Refinement continues recursively until the parameterized distance between every adjacent target $t_{\theta_{i+1}-t_{\theta_i}}$ along the path falls below a tolerance ζ. This method of planning and refining are summarized in three algorithms presented in Fig. 4.1.

4.3 Results

Our planner has been tested on several different case studies and has successfully provided smooth paths in joint space that meet performance objectives. In situations where the manipulator is unable to navigate the path due to kinematic constraints, our planner provides motion close to the desired path. Given kinematic redundancy within the problem however, our planner performed very well for a large set of input parameters and was tuned with minimal effort despite being a multi-objective problem. We present here several examples that demonstrate the planner's performance.

4.3.1 4-DOF Planar Robot

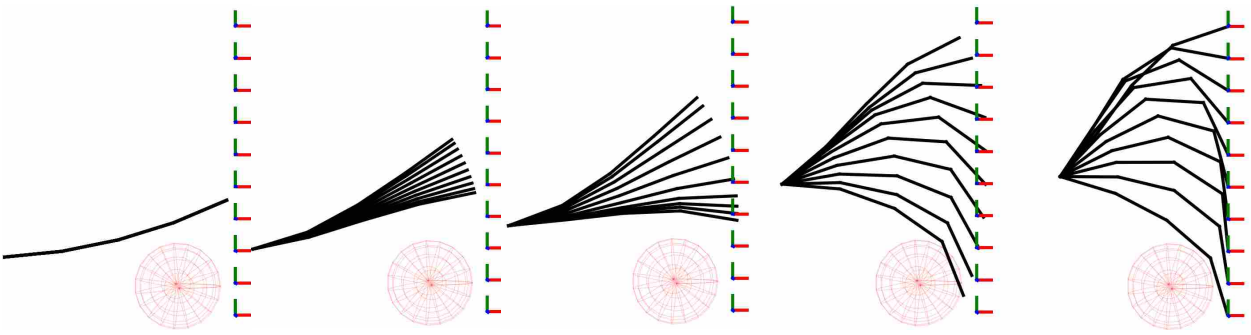
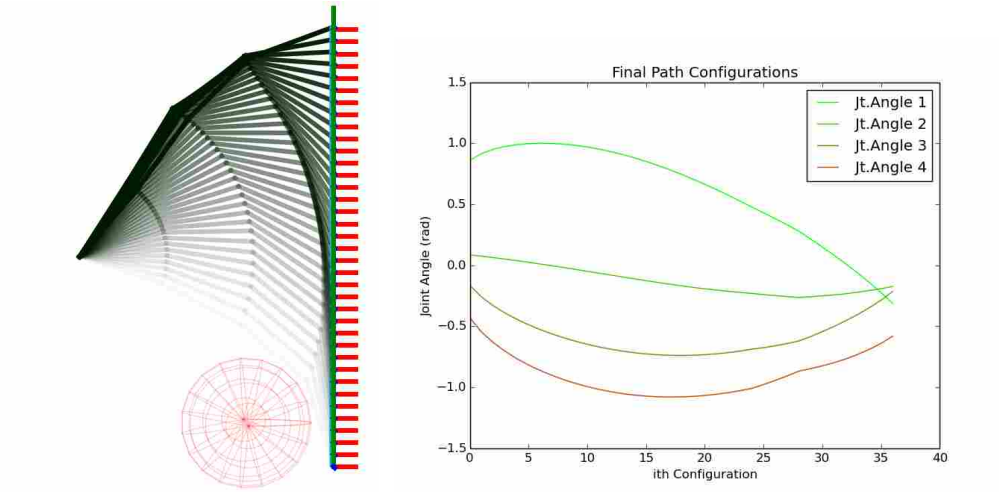


Figure 4.2: Optimal inverse kinematic branching motion is shown in which it is desired that a manipulator navigate continuously along a wall while avoiding a circular collision boundary. Iterations 0, 20, 60, 150 and 240 are shown.

We have used the planner on an example similar to that shown in Fig. 1.2. The 2D manipulator has four links connected with revolute joints, and the goal is for the end effector to track

a linear position (with no orientation goal) while avoiding a circular collision obstacle (shown in red). Hard constraints are included to model joint stops. Figure 4.2 shows how the planner begins by branching a set of 10 virtual arms towards assigned targets on the path while considering both the spherical collision boundary, as well as a planar constraint representing the wall. Once the algorithm converges on a final set of configurations, the path is refined in order to plan overall smoother global motion as is shown in Fig. 4.3a. Figure 4.3b shows how the refinement routine smoothly connects configurations in joint space which is important for implementation on a real robot.



(a) Refined Path ($\zeta = .05$) (b) Smooth, Continuous Paths in Joint Space

Figure 4.3: Configurations found along the final path. The 1 meter vertical path was planned for a four 20cm-link robot.

Figure 4.4 shows the final, scaled costs contributed by the different objective terms in the quadratic objective function as a function of the number of steps taken in the QP algorithm. This figure shows how secondary objectives are continuously scaled down so as to prioritize the objective pushing arms to their respective targets.

Our implementation was done in Python, with SWIG bindings that link to a custom package in C++ handling robot kinematics. Running on a single 3.4 GHz Intel Core i7-6700, and using the Python QP solver CVXOPT [60], we've solved the four-link planar problem in under 3 seconds. We expect significant speedups with more efficient implementations.

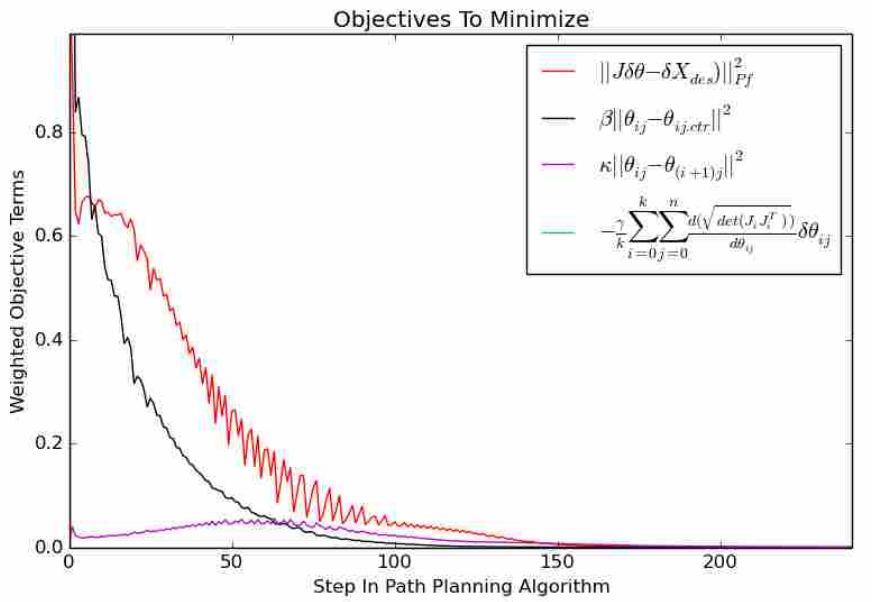


Figure 4.4: Secondary objective weights are continuously scaled down during the algorithm's progression in order to prioritize the cost associated with arms being far from their targets.

4.3.2 8-DOF Manipulator Painting

We now show a practical example in which the 8-DOF manipulator shown in Fig. 2.2 performs a painting task. In this example, it is desired that the manipulator's end effector remains normal to the surface (twisting motion around the end effector's z-axis is acceptable) as it paints. We use a kinematic model that matches the behavior of the pneumatically actuated robot developed by Pneubotics. This model has two degrees of freedom at every joint (the orange sections) that allow the manipulator to bend in a circular arc in any desired direction. Hard constraints are implemented in the QP to reflect the manipulator's joint limits. Defining our path function $\chi(t)$ to be a square wave that sweeps across a $1m^2$ surface, the algorithm assigns targets and spreads arms to reach those targets as is shown in Fig. 4.5. The shared base is allowed to move freely during branching motion to accommodate motion objectives. Once a rough set of configurations has been structured around the desired path, we refine it to obtain the smooth joint motions shown in Fig. 4.6a and 4.6b. In this case, our implementation of this algorithm took under six seconds to perform the highest level of planning (k=20) and a little less than a second to perform refinement. We emphasize again that a more optimized implementation should be able to perform planning much quicker.

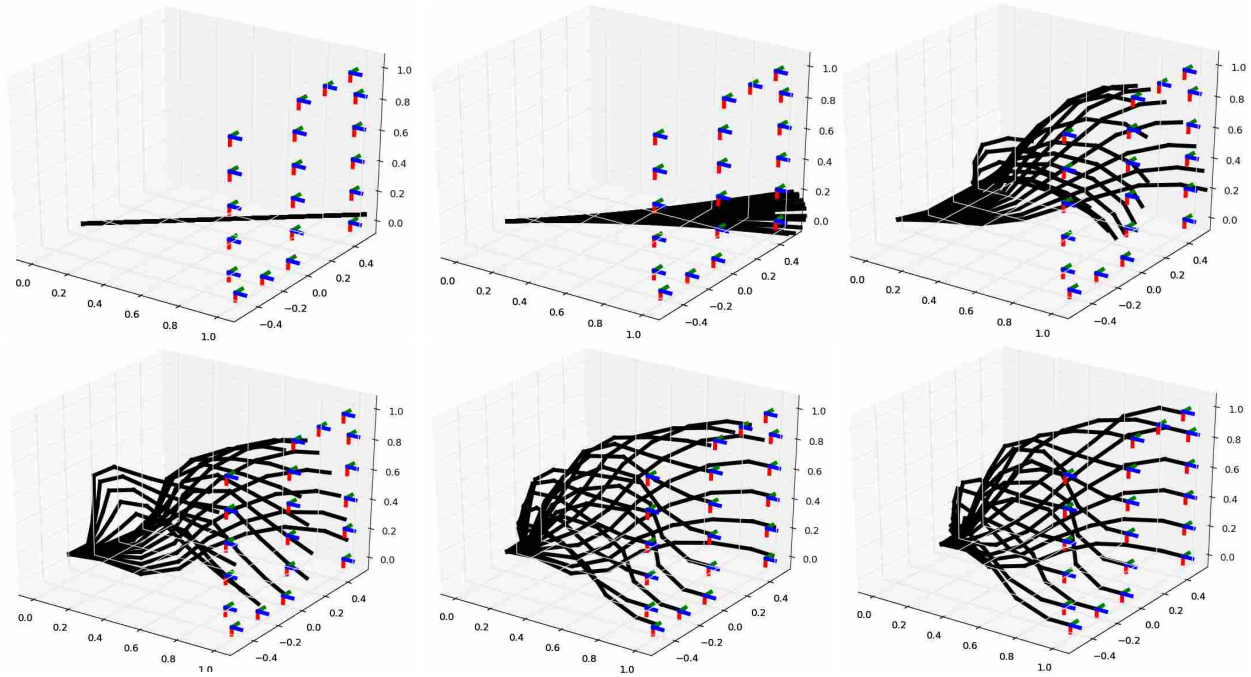
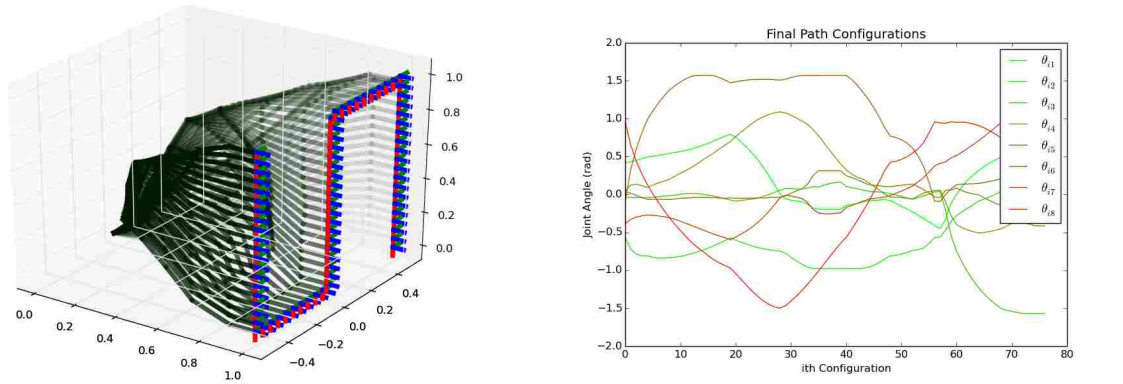


Figure 4.5: Branching to reach a square-wave path function χ that imitates painting motion over a flat surface. Iterations 0, 10, 40, 120, 250, and 450 are shown.



(a) Refined Path ($\zeta = .02$)

(b) Continuous Paths in Joint Space

Figure 4.6: Configurations found along the final path. The path was planned for a pneumatically actuated robot built by Pneubotics.

Similar motions as those shown in this example were executed on real hardware at the Pneubotics facility in San Francisco as shown in Fig. 4.7. In this case, the planner enabled a 6DOF pneumatically actuated, soft robot to sand a $1m^2$ flat surface in a little under a minute. A video of the robot carrying out this task can be found at `https://youtu.be/wrmfIZMHVkY`.

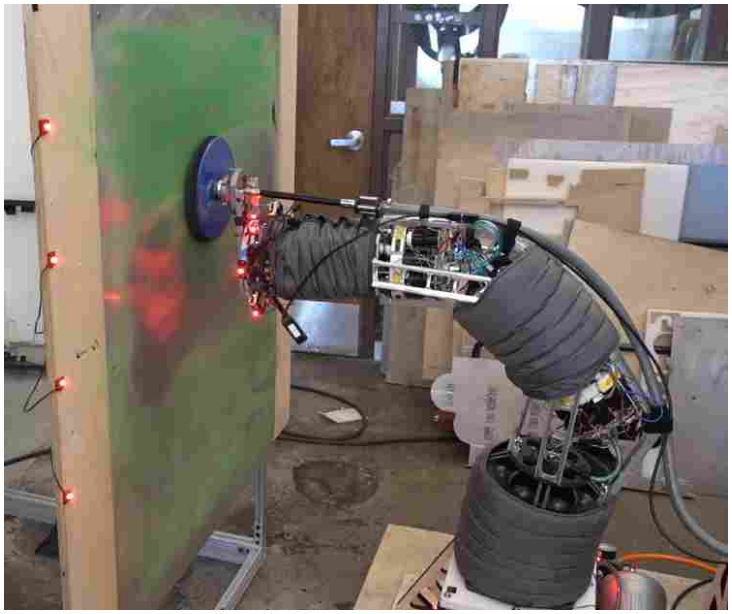
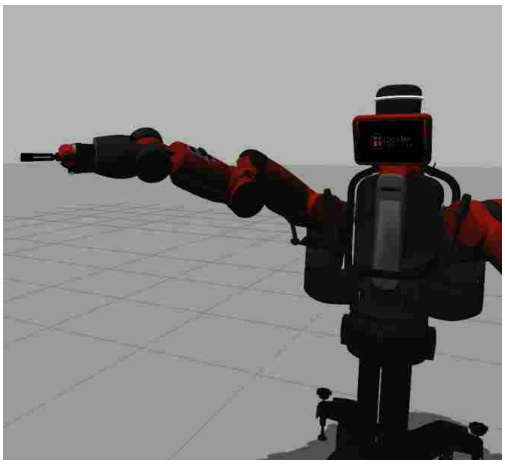


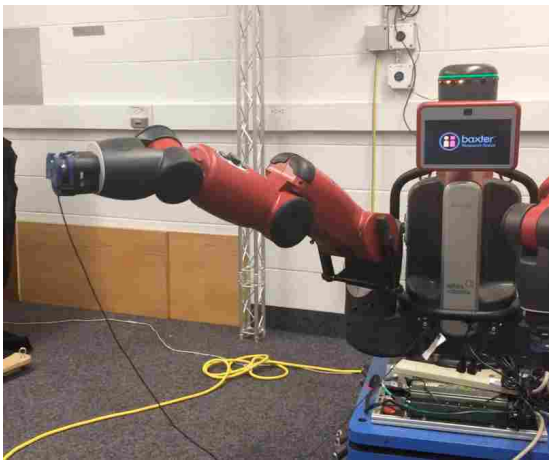
Figure 4.7: A 6DOF pneumatically actuated, soft robot sands a surface using our QP planner. See `https://youtu.be/wrmfIZMHVkY` for video.

4.3.3 Baxter Robot Performing Painting Motions

We implemented the planner for Baxter, a collaborative robot made by RethinkRobotics (see Fig. 4.9). Baxter's arms each have seven degrees of freedom: four revolute joints and three pin joints. Painting motions were planned in which Baxter was assigned to paint a 60cm x 40cm



(a) Motion Carried out on Simulation



(b) Motion Carried out on Hardware

Figure 4.8: Painting motions performed in simulation and on hardware can be found at `https://youtu.be/LoUyA76mmpQ` and `https://youtu.be/onwH3KWfVOE` respectively.

surface in 15 seconds. Motion was carried out with a fixed end effector orientation. Videos of motion in simulation and as applied on hardware can be found at https://youtu.be/LoUyA76mmpQ and https://youtu.be/onwH3KWfV0E respectively.

4.3.4 6-DOF Arm Mounted on A Planetary Rover: Pick And Place Motion

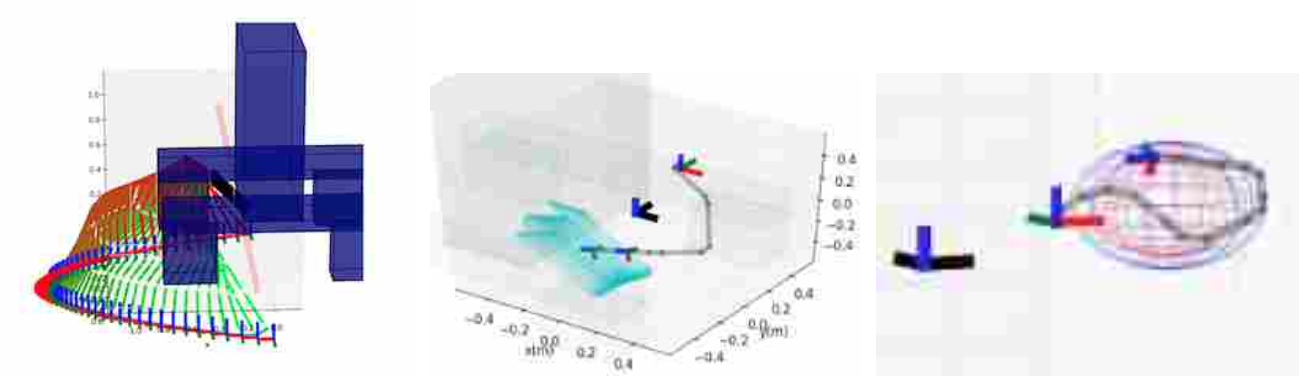
We include videos of more complicated 3D paths generated by our planner. In this case, a six degree of freedom robot with a kinematic joint model similar to the one described in [6] is mounted to a model of a planetary rover from NASA Ames (K-REX). The ground is included as an obstacle as are spherical obstacles which represents the chassis of the rover. The goal trajectory takes the end effector from a location on the ground on the side of the rover to a location under and in front of the rover with a specific orientation with respect to the ground. This may represent a soil sample retrieval and storage task. A frontal view of the different iteration steps of the optimization is at https://youtu.be/i815T2GUGo4 while a frontal and side view of the planned path are at https://youtu.be/HHf0B43luIs and https://youtu.be/KPhWVgrnhBk. A snapshot of this motion is shown in Fig. 4.9a.

4.3.5 6-DOF Arm Mounted on A Planetary Rover: Wiping Motion

This same arm mounted to K-REX is also shown using its link to perform wiping motions on a flat surface at https://youtu.be/whDpZFsFwXI and https://youtu.be/A0xbdewknNs. This motion could represent wiping off a solar panel in dusty terrain. In this case, extra terms were added to the QP that encouraged the last links of the arm to remain flat against the panel during motion. These terms were constructed by assigning appropriate targets to link bodies instead of end effectors, and weighting orientation about the plane of interest appropriately. A snapshot of this motion is given in Fig. 4.9b.

4.3.6 10-DOF Arm: Wrapping Motions

This form of planning has the potential of being useful for very high dimensional systems including those with tentacle and finger like mechanisms. This is especially the case as the QP problem scales quite nicely to higher dimensions as discussed earlier. We show an example of



(a) Pick and Place Motion (b) Wiping Motion With Link (c) Wrapping Motion

Figure 4.9: The planner has enabled robots to wipe surfaces and wrap around objects.

planning motion to allow a 10-dimensional robot (with continuous-curvature joints as discussed) to grasp a sphere at `https://youtu.be/a2LpDKGXRZg`. A snapshot of this motion is shown in Fig. 4.9c.

4.3.7 Sample-Based Path Planning Approach

For comparison with our planner we implemented a second form of path planning that used fast forward kinematic sampling (see Chapter 3) in combination with inverse kinematic techniques to find many configurations along the path. A simple graph search algorithm (Dijkstra's algorithm)

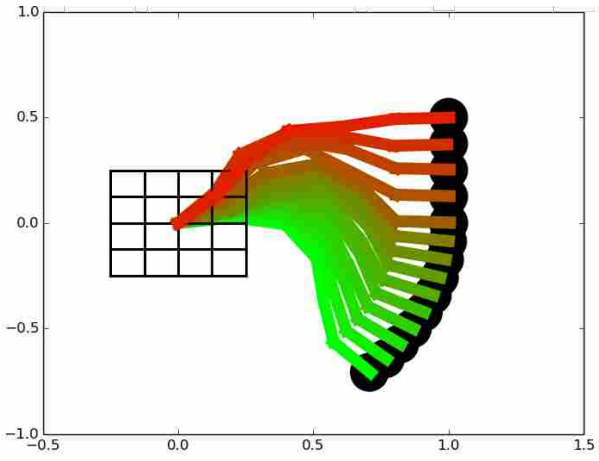


Figure 4.10: Forward kinematics is used to identify many configurations near the path. These configurations are then driven to the path using QP inverse kinematics as discussed. A graph search algorithm is then employed to link similar configurations together into a sequence representing motion.

was then used to connect similar configurations along the path into an optimal sequence representing desired motion. This approach provided smooth motion but generally took much longer to solve (30-60s for simple paths) and could not easily include secondary objectives as desired. Additionally it could not optimize the base mounting position from which motion was to be carried out. For these reasons we abandoned this form of planner in favor of the presented QP method.

4.4 Discussion

We have presented a robust, scalable framework of path planning for high-dimensional manipulators using quadratic programming. We have demonstrated its utility in simultaneously planning for the optimal location of the mobile base, as well as joint motions needed to traverse a desired end-effector path. Further we have shown how secondary objectives can be incorporated and appropriately tuned in this planning routine using additional terms in the quadratic objective function. Finally, we have presented a few practical examples that motivate the use of this algorithm for common manipulation tasks.

We anticipate that this method will provide a platform by which optimal motion planning can be efficiently and effectively carried out for a variety of useful tasks that require end-effector trajectories including surfacing operations (e.g., painting, washing, sanding), manufacturing processes (e.g., drilling holes, assembling parts), and practical every-day tasks (e.g., opening a door, turning a wheel, vacuuming). With an appropriate weighting strategy, this framework can be generalized to plan motion for operations without required end-effector trajectories as well (e.g., moving objects from one place to another). In either case it provides task-optimized base positions which makes it especially suited to mobile-manipulators which can choose base locations relative to the task at hand.

CHAPTER 5. CONCLUSION

The theory and methods introduced in this work are offered as a platform on which others can build. We expect that the methods will be especially useful for those designing mobile robots with many degrees of freedom and with inherent compliance similar to the soft robot platforms we have introduced. Naturally our methods have limitations and logical extensions. We list several here.

5.1 Limitations and Possible Extensions

5.1.1 Optimization

To improve the accuracy of our optimization methodology, more advanced modeling strategies that may be used. As we have noted, we have assumed rigid links with induced deflections occurring only at the joints. While this assumption has yielded a convenient approximation, modeling link deflections to more adequately describe the behavior of our platform would improve accuracy. A more accurate joint stiffness model would also help in this regard. This could be developed using a nonlinear stiffness model or by factoring in the effect of joint configuration and bladder pressure within the currently formulated linear fit.

We also recommend that future work reexamine the problem of discretizing SO(3) space. Finding a discretization strategy that eliminates the dependence of the number of orientations found at a given point and the reference frame from which orientations are measured at that point is an important step in this regard. Other representations of orientations (i.e., quaternions, Euler angles, etc.) may be more suited to discretizing than the axis-angle representation we have used.

Another logical and important extension of our optimization method is to develop and incorporate more metrics within the optimization routine that accurately describe other desirable qualities of soft robotic manipulators. The proposed optimization structure readily allows for the

definition of additional metrics and can produce high dimensional Pareto fronts for use in the early design stages of these robots. Potentially useful optimization metrics may describe a manipulator's capacity to generate high end effector velocities for "reckless" behavior that is more suitable to soft robots (e.g., using a hammer to hit a nail) or perform pre-determined tasks at specific regions in the workspace of the manipulator. They may also describe how design parameters affect the dynamics and control of these systems as these relationships become better understood.

In addition to new metrics, additional design variables may also be added to our optimization. Link and joint sizes may be included as design variables and the positioning of valves on link bodies could be optimized. In some cases, introducing fixed bend angles on certain links has been shown to increase design dexterity in specific areas of the workspace for specific tasks. Choosing appropriate design variables and design metrics is a critical design task that is largely platform-dependent. In this sense, our application is offered as an example to those seeking to apply similar optimization strategies.

5.1.2 Path Planning

Despite considering the entire path the manipulator takes, the planning method we have introduced is still a local optimization method and will struggle with more complex paths. For such cases, this algorithm could be used to find a common base position as well as collision-free start and end configurations that would then be passed into a sampling-based path planner [42–45]. Our algorithm could then be initialized with the resulting path of this sampling-based planner and be used to shorten and smooth this path.

In formulating our planner we have assumed rigid body kinematics. Thus our planner does not allow for links or joints to bend or deflect in uncontrolled ways. Using a model that more accurately describes the kinematics of soft robots is an important extension of this work. This could be done using the stiffness modeling techniques introduced in the optimization section.

Finally, though meant primarily as a means to optimize base position and redundant motion, our algorithm can also perform planning without a specified end effector path by simply fixing the start and end configurations and setting weights of intermediate end-effector targets to zero. Comparing this capability against other standard trajectory optimization algorithms ([50–52, 61]) is an important and logical extension of this work that we wish to pursue in future studies.

5.2 Observations and Closing Remarks

We have observed a fundamental relationship between the processes of design optimization and path planning presented in this work. This relationship arises from the fact that the kinematics of a robot significantly affects how well it is able to track trajectories with desired end-effector positions and orientations. We observed in this regard that optimized robot designs produced smoother paths with less error than unoptimized designs. This provides valuable validation of our optimization routine and of the dexterity metric presented. We are confident this same system of optimization can be applied to other platforms with similar results.

The quadratic programming planner presented continues to be (as of the publication of this work) the state-of-the-art motion planner for robots designed by Pneubotics. It has enabled their robots to perform surfacing operations including spraying, sanding, and painting and shows promise for many other behaviors as demonstrated in this work. It has been shown to be robust, fast and accurate in providing path configurations along a desired end effector trajectory and provides a natural way to allow the user to tune paths based on desired objectives. We are confident that this method will generalize to other systems and provide a useful, intuitive solution to path planning problems.

REFERENCES

[1] Alberty, M., 2014. Rover searches california desert for water to simulate future lunar missions. viii, 5

[2] Murray, R. M., Li, Z., Sastry, S. S., and Sastry, S. S., 1994. *A mathematical introduction to robotic manipulation*. CRC press. xiii, 9, 13, 14, 15, 16, 17, 19, 27, 45

[3] Voisembert, S., Riwan, A., Mechbal, N., and Barraco, A., 2011. "A novel inflatable robot with constant and continuous volume." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 5843–5848. 1, 7

[4] Sanan, S., Moidel, J. B., and Atkeson, C. G., 2009. "Robots with inflatable links." In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, IEEE, pp. 4331–4336. 1, 7

[5] Sanan, S., Lynn, P. S., and Griffith, S. T., 2014. "Pneumatic torsional actuators for inflatable robots." *Journal of Mechanisms and Robotics,* **6**(3), p. 031003. 1

[6] Best, C. M., Wilson, J. P., and Killpack, M. D., 2015. "Control of a pneumatically actuated, fully inflatable, fabric-based, humanoid robot." In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, IEEE, pp. 1133–1140. 1, 8, 23, 58

[7] Rus, D., and Tolley, M. T., 2015. "Design, fabrication and control of soft robots." *Nature,* **521**(7553), pp. 467–475. 2

[8] Marchese, A. D., Katzschmann, R. K., and Rus, D., 2014. "Whole arm planning for a soft and highly compliant 2d robotic manipulator." In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, pp. 554–560. 2

[9] Voisembert, S., Mechbal, N., Riwan, A., and Aoussat, A., 2013. "Design of a novel long-range inflatable robotic arm: Manufacturing and numerical evaluation of the joints and actuation." *Journal of Mechanisms and Robotics,* **5**(4), p. 045001. 2

[10] Buss, S. R., 2004. "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods." *IEEE Journal of Robotics and Automation,* **17**(1-19), p. 16. 4, 43

[11] Shankar, K., Burdick, J. W., and Hudson, N. H., 2014. "A Quadratic Programming Approach to Quasi-Static Whole-Body Manipulation." *Algorithmic Foundation of Robotics*. 4, 29, 43, 44, 45, 47

[12] Bodily, D. M., Allen, T. F., and Killpack, M. D., 2017. "Multi-objective optimization of a soft, pneumatic robot." In *The International Conference on Robotics and Automation*, IEEE. 6

[13] Bodily, D. M., Allen, T. F., and Killpack, M. D., 2017. "Motion planning for mobile robots using inverse kinematics branching." In *The International Conference on Robotics and Automation*, IEEE. 6

[14] Albu-Schäffer, A., Eiberger, O., Grebenstein, M., Haddadin, S., Ott, C., Wimböck, T., Wolf, S., and Hirzinger, G., 2008. "Soft robotics." *Robotics & Automation Magazine, IEEE,* **15**(3), pp. 20–30. 7

[15] Diftler, M. A., Mehling, J., Abdallah, M. E., Radford, N. A., Bridgwater, L. B., Sanders, A. M., Askew, R. S., Linn, D. M., Yamokoski, J. D., Permenter, F., et al., 2011. "Robonaut 2-the first humanoid robot in space." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 2178–2183. 7

[16] Best, C. M., Hyatt, P. E., Killpack, M. D., Rupert, L., and Sherrod, V., 2015. "Model predictive control for pneumatically actuated soft robots." In *Robotics and Automation Magazine*, IEEE. 8

[17] Allen, T. F., Hein, G., and Albert, K., 2016 (In Submission). "Constant-curvature continuum manipulator kinematics." In *IEEE Transactions on Robotics*, IEEE. 9, 12, 16

[18] Mombaur, K., Kheddar, A., Harada, K., Buschmann, T., and Atkeson, C., 2014. "Model-based optimization for robotics." *IEEE ROBOTICS & AUTOMATION MAGAZINE,* **21**(3), pp. 24–161. 21

[19] Oral, S., and Ider, S. K., 1997. "Optimum design of high-speed flexible robotic arms with dynamic behavior constraints." *Computers & structures,* **65**(2), pp. 255–259. 21

[20] Paredis, C. J., and Khosla, P., 1995. "Design of modular fault tolerant manipulators.". 21

[21] Rao, R. V., and Waghmare, G., 2015. "Design optimization of robot grippers using teaching-learning-based optimization algorithm." *Advanced Robotics,* **29**(6), pp. 431–447. 21

[22] Ramezan Shirazi, A., Seyyed Fakhrabadi, M. M., and Ghanbari, A., 2014. "Analysis and optimization of the 5-rpur parallel manipulator." *Advanced Robotics,* **28**(15), pp. 1021–1031. 21

[23] Hiller, J., and Lipson, H., 2012. "Automatic design and manufacture of soft robots." *Robotics, IEEE Transactions on,* **28**(2), pp. 457–466. 22

[24] Park, J.-Y., Chang, P.-H., and Yang, J.-Y., 2003. "Task-oriented design of robot kinematics using the grid method." *Advanced robotics,* **17**(9), pp. 879–907. 22

[25] Chapelle, F., and Bidaud, P., 2006. "Evaluation functions synthesis for optimal design of hyper-redundant robotic systems." *Mechanism and machine theory,* **41**(10), pp. 1196–1212. 22

[26] Yang, G., and Chen, I.-M., 2000. "Task-based optimization of modular robot configurations: minimized degree-of-freedom approach." *Mechanism and Machine Theory,* **35**(4), pp. 517 – 540. 22

[27] Tesch, M., Schneider, J., and Choset, H., 2013. "Expensive multiobjective optimization for robotics." In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 973–980. 22

[28] Coello, C. A. C., Christiansen, A. D., and Aguirre, A. H., 1998. "Using a new ga-based multiobjective optimization technique for the design of robot arms." *Robotica,* **16**(04), pp. 401–414. 22

[29] Knowles, J., 2006. "Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems." *IEEE Transactions on Evolutionary Computation,* **10**(1), pp. 50–66. 22

[30] Balling, R., and Wilson, S., 2001. "The maxi-min fitness function for multi-objective evolutionary computation: Application to city planning." In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2001)*, pp. 1079–1084. 22, 38

[31] Gosselin, C. M., 1992. "The optimum design of robotic manipulators using dexterity indices." *Robotics and Autonomous systems,* **9**(4), pp. 213–226. 23

[32] Klein, C. A., and Blaho, B. E., 1987. "Dexterity measures for the design and control of kinematically redundant manipulators." *The International Journal of Robotics Research,* **6**(2), pp. 72–83. 23

[33] Pusey, J., Fattah, A., Agrawal, S., and Messina, E., 2004. "Design and workspace analysis of a 6–6 cable-suspended parallel robot." *Mechanism and machine theory,* **39**(7), pp. 761–778. 23

[34] Kucuk, S., and Bingul, Z., 2006. "Comparative study of performance indices for fundamental robot manipulators." *Robotics and Autonomous Systems,* **54**(7), pp. 567–573. 23

[35] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G., 2009. "Mobile robots." *Robotics: Modelling, Planning and Control*, pp. 469–521. 23

[36] Yoshikawa, T., 1990. "Foundation of robotics." *Analysis and Control. The MIT Press*. 23, 32, 48

[37] Kim, K. J., and Tadokoro, S., 2007. "Electroactive polymers for robotic applications." *Artificial Muscles and Sensors (291 p.), Springer: London, United Kingdom*. 23

[38] Kim, B., Lee, M. G., Lee, Y. P., Kim, Y., and Lee, G., 2006. "An earthworm-like micro robot using shape memory alloy actuator." *Sensors and Actuators A: Physical,* **125**(2), pp. 429–437. 23

[39] Mayorga, R. V., Carrera, J., and Oritz, M. M., 2005. "A kinematics performance index based on the rate of change of a standard isotropy condition for robot design optimization." *Robotics and Autonomous Systems,* **53**(3), pp. 153–163. 23

[40] Kumar, V., Sen, S., Roy, S. S., Har, C., and Shome, S., 2014. "Design optimization of serial link redundant manipulator: an approach using global performance metric." *Procedia Technology,* **14**, pp. 43–50. 23

[41] Buss, S. R., and Kim, J.-S., 2005. "Selectively damped least squares for inverse kinematics." *journal of graphics, gpu, and game tools,* **10**(3), pp. 37–49. 29

[42] Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H., 1996. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation,* **12**(4), pp. 566–580. 43, 62

[43] LaValle, S. M., Yakey, J. H., and Kavraki, L. E., 1999. "A probabilistic roadmap approach for systems with closed kinematic chains." In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, Vol. 3, IEEE, pp. 1671–1676. 43, 62

[44] Kuffner, J. J., and LaValle, S. M., 2000. "Rrt-connect: An efficient approach to single-query path planning." In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, Vol. 2, IEEE, pp. 995–1001. 43, 62

[45] Karaman, S., and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning." *The International Journal of Robotics Research,* **30**(7), pp. 846–894. 43, 62

[46] McMahon, T., Thomas, S., and Amato, N. M., 2014. "Sampling-based motion planning with reachable volumes: theoretical foundations." In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 6514–6521. 43

[47] Kim, B., Um, T. T., Suh, C., and Park, F., 2016. "Tangent bundle rrt: A randomized algorithm for constrained motion planning." *Robotica,* **34**(01), pp. 202–225. 43

[48] Jaillet, L. G., and Porta Pleite, J. M., 2011. "Path planning with loop closure constraints using an atlas-based rrt." In *Proceedings of the 15th International Symposium on Robotics Research*, Springer, pp. 1–16. 43

[49] Berenson, D., Kuffner, J., and Choset, H., 2008. "An optimization approach to planning for mobile manipulation." In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, IEEE, pp. 1187–1192. 43

[50] Ratliff, N., Zucker, M., Bagnell, J. A., and Srinivasa, S., 2009. "Chomp: Gradient optimization techniques for efficient motion planning." In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, pp. 489–494. 43, 62

[51] Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S., 2011. "Stomp: Stochastic trajectory optimization for motion planning." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 4569–4574. 43, 62

[52] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P., 2014. "Motion planning with sequential convex optimization and convex collision checking." *The International Journal of Robotics Research,* **33**(9), pp. 1251–1270. 43, 62

[53] Chitta, S., Cohen, B., and Likhachev, M., 2010. "Planning for autonomous door opening with a mobile manipulator." In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, pp. 1799–1806. 44

[54] Gochev, K., Safonova, A., and Likhachev, M., 2012. "Planning with adaptive dimensionality for mobile manipulation." In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 2944–2951. 44

[55] Hornung, A., Phillips, M., Jones, E. G., Bennewitz, M., Likhachev, M., and Chitta, S., 2012. "Navigation in three-dimensional cluttered environments for mobile manipulation." In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 423–429. 44

[56] Escande, A., Mansard, N., and Wieber, P.-B., 2014. "Hierarchical quadratic programming: Fast online humanoid-robot motion generation." *The International Journal of Robotics Research,* **33**(7), pp. 1006–1028. 44

[57] Siciliano, B., and Khatib, O., 2008. *Springer handbook of robotics*. Springer Science & Business Media. 44

[58] Coumans, E., et al., 2013. "Bullet physics library." *Open source: bulletphysics. org,* **15**. 49

[59] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S., 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space." *IEEE Journal on Robotics and Automation,* **4**(2), pp. 193–203. 49

[60] Andersen, M. S., Dahl, J., and Vandenberghe, L., 2013. "Cvxopt: A python package for convex optimization, version 1.1. 6." *Available at cvxopt. org*. 54

[61] Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S., 2013. "Chomp: Covariant hamiltonian optimization for motion planning." *The International Journal of Robotics Research,* **32**(9-10), pp. 1164–1193. 62