2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Linear Actuator Robots: Differential Kinematics, Controllability, and Algorithms for Locomotion and Shape Morphing

Nathan Usevitch, Zachary Hammond, Sean Follmer, Mac Schwager

***Abstract*— We consider a class of robotic systems composed of high elongation linear actuators connected at universal joints. We derive the differential kinematics of such robots, and formalize concepts of controllability based on graph rigidity. Control methods are then developed for two separate applications: locomotion and shape morphing. The control algorithm in both cases solves a series of linearly constrained quadratic programs at each time step to minimize an objective function while ensuring physical feasibility. We present simulation results for locomotion along a prescribed path, and morphing to a target shape.**

I. Introduction

In this paper we present a control methodology for robots made up of high-elongation linear actuators connected at universal joints into a network, which we call Linear Actuator Robots (LARs). Such robots can change their shape dramatically through the coordinated actuation of their linear members. This shape change ability can be used for a multitude of tasks, including locomotion, manipulation, and matching of 3D target shapes (shape morphing), examples of which are shown in Fig. 1. We present a differential kinematic analysis of LARs, relating the velocities of the vertices in the structure to the rate of change of the actuator lengths. This allows us to link concepts from graph rigidity to the controllability of the robot structure. We use this kinematic analysis to derive control algorithms for locomotion and shape morphing, both based on the same underlying sequential convex programming algorithm tailored to the kinematics and constraints of LARs.

LARs have the potential to change their shape to interact with the environment, such as growing legs to traverse obstacles, turning into a ball to roll down hills, or morphing into the shape needed to manipulate an object with complicated geometry. Such a robot would be valuable in the unstructured environments of search and rescue missions. LARs can also serve as a type of high-speed 3D printer, changing shape to represent 3D objects and responding to a human designer's digital manipulations in real time.

Our work builds upon significant advancements in the modeling and control of TETROBOTs [1]–[3], as well as other tetrahedron-based modular robot systems. TETROBOTs are a type of LAR that have a particular repeating graphical motif which facilitates kinematics computations. The focus of existing work in TETROBOTs is in the physical design of the robot, algorithms for propagating kinematic chains of tetrahedrons or octahedrons [1], and centralized and decentralized algorithms for dynamic locomotion [2,3]. Other work has focused on designing gaits for similar systems [4]. Our approach differs from the TETROBOT work in that we propose models and control algorithms for robots of arbitrary graphical structure, not limited to tetrahedrons or octahedrons. We treat differential kinematics (rather than solving kinematics algebraically), which allows for general solutions regardless of robot graph topology. Also, unlike existing work in TETROBOTS, we focus on controlling the 3D shape of our robots. Compact linear actuators have recently been developed that can extend up to 10 times their nominal length [5,6]. This greatly expands the types of shapes that a LAR can reach, and motivates this work. In future work we plan to implement the proposed algorithms on a system composed of pneumatic reel actuators developed by the authors in [6].

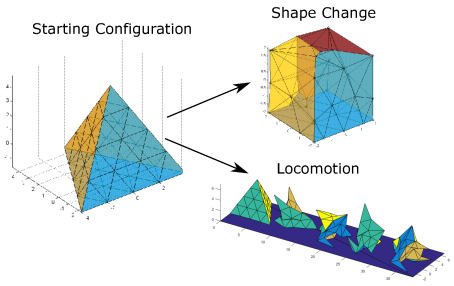

Fig. 1: We present algorithms for Linear Actuator Robots (LARs) to change shape and to locomote. A LAR composed of 108 linear actuators and 34 vertices morphs from a pyramid to a cube (top right), and locomotes (bottom right).

Other work has considered 3D shape morphing robots in different contexts. In [7], linear actuators and joints are connected to form structures capable of some local shape change, including a self leveling bridge. Mazzone et al. present an active-surface type device that uses prismatic joints to deform a surface into arbitrary shapes while respecting some constraints [8]. Tetrahedral robots have been considered as a candidate for planetary exploration due to their ability to locomote over varied terrain [9]. Tensegrity robots are similar to tetrahedral robots in that their form can be changed by varying the lengths of some members, but with the additional constraint that some elements must remain in tension. Several controllers have been proposed for locomotion of tensegrity robots, many of them based on evolutionary or machine learning approaches [10]. Other work has also focused on mechanism design for robots with linear actuators [11,12]. The majority of these studies have been limited to using actuators that can undergo only moderate extension. Our algorithm for shape morphing also builds upon existing work in computer graphics for morphing

This work was supported in part by National Science Foundation Award 1637446, ONR grant N00014-16-1-2787, and US Army Medical Research and Materiel Command grant W81XWH-15-C-0091.

one computer rendered shape or mesh into another. This problem is often divided into several sub tasks: finding a compatible mesh of the two target shapes [13], and performing a morph that ensures a smooth and natural deformation between shapes [14,15]. Our problem differs significantly from this work in that we have a fixed mesh topology, and our robot has physical and kinematic constraints that are not present in the computer graphics context.

II. Model Formalization and Problem Statement

Formally, we model a Linear Actuator Robot (LAR) as a framework consisting of a graph and vertex positions. The graph is denoted as $G = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1...N\}$ are the vertices of the graph, and $\mathcal{E} = \{...\{i,j\}...\}$ are the undirected edges of the graph. The position of each vertex is assigned $p_i \in \mathbb{R}^3$. We will perform only kinematic analysis of the network, and leave dynamic analysis of the network for future work. The kinematic state of the robot is fully represented by the concatenation of all vertex positions $x = [p_{1x}, ..., p_{nx}, p_{1y}, ..., p_{ny}, p_{1z}, ..., p_{nz}]^T$. We define a length vector L, which is a concatenated vector of the lengths of all edges in the graph $L_k = \|p_i - p_j\|$, where an actuator between nodes i and j is represented by an edge $e_k \in \mathcal{E}$. The vector L is of length n_L, or the number of edges of the graph, and can be directly computed from the pair (G, x) (note that this is the "inverse kinematics" for LAR robots, which is trivial, as noted by [1]). In contrast, we aim to present algorithms to control the positions of the vertices by controlling the lengths of the actuators (the "forward kinematics") in a coordinated fashion, to locomote or to change from an initial shape to a target shape.

We now introduce a notion of feasibility describe the configuration of the LAR that satisfy physical constraints:

Definition 1: A framework (G, x) is feasible if it meets two types of physical constraints: (i) the lengths of all bars fall within a fixed maximum and minimum length range, and (ii) the actuators do not physically intersect (except at their endpoints, where they are joined).

The squared length of an actuator e that connects nodes $\{i, j\}$, is quadratic in x and can be computed using a partial graph Laplacian $\mathcal{L}_k$ where $\mathcal{L}_{ii} = \mathcal{L}_{jj} = 1$,$\mathcal{L}_{ij} = \mathcal{L}_{ji} = -1$, and all other entries in $\mathcal{L}$ are 0. The constraint can then be written as follows:

$$L_{min}^2 \leq x^T \left[\mathcal{L}_k \otimes I_d\right] x \leq L_{max}^2. \tag{1}$$

We note that constraints of the quadratic form $x^T Q x > 0$ are convex if and only if Q is positive semi-definite. The Laplacian matrix is always positive semi-definite, meaning that the maximum length constraint is convex in the node positions, while the minimum length constraint is not. Our algorithms will handle non-convex and nonlinear constraints by considering only differential constraints, which are linear.

To determine if two actuators cross, the minimum distance between them must be greater than d_{min}, a positive diameter of the actuator assuming that the actuator can be represented as a cylinder. The minimum distance between actuators connecting vertices i, j and k, l is denoted as d_{ij}^{kl}, and can be expressed as follows:

$$d_{ij}^{kl} = min\|(p_i + \alpha(p_j - p_i)) - (p_k + \gamma(p_l - p_k))\| \ \alpha, \gamma \in (0,1) \tag{2}$$

These links are not in collision if $d_{ij}^{kl} > d_{min}$. Efficient algorithms for this computation have been explored previously [16]. Computing if a configuration of n_L links contains any collisions requires $\frac{(n_L^2 - n_L)}{2}$ collision checks of the type detailed in equation 2.

Future work will consider a constraint that keeps the minimum angles between two actuators connected at a vertex above a minimum value.

A. Problem Statement

In this paper, two primary applications of LAR robots will be explored: how to change the robot between a wide variety of shapes, and how to drive such a robot to locomote.

Problem 1: (Locomotion): Move the center of mass of the robot in a prescribed direction v_{cm}, or along a prescribed trajectory $x_{cm}(\tau)$.

Problem 2: (Shape Matching): Given a target shape Q and an initial network configuration (G, x_0), find a configuration (G, x^*) such that x^* is as close as possible to Q, while (G, x^*) remains feasible. Also determine a path $x(t), t \in [0, 1]$ that leads from x_0 to x^* such that $x(t)$ is feasible for all $t \in [0, 1]$.

III. Kinematics

This section discusses the kinematics that relate the changing actuator lengths to the changing vertex positions. Central to understanding the relationship between node positions and edge lengths is the concept of the rigidity of a framework.

In a network of linear actuators, each link length imposes one constraint on the system. These constraints can be written as follows

$$L_k = \|p_i - p_j\| \forall (\{i,j\} \in \mathcal{E}). \tag{3}$$

One method of finding the vertex position from the link lengths is to find vertex positions that satisfy all constraint equations in the network, up to translation and rotation of the entire network. Several classes of solutions exist, based on the rigidity of the underlying graph. If the system of equations has infinite solutions, the framework is not rigid, as it is possible to move the system relative to itself without violating length constraints. A framework is rigid if there are a discrete number of solutions to the constraint equations, and all deflections of the system relative to itself violate the length constraints. A more restrictive, but easier to test category is infinitesimal rigidity, which means that all infinitesimal deflections of the system relative to itself violate the length constraints.

Of particular interest in the design of LARs are minimally rigid graphs. A minimally rigid graph is a rigid graph where the removal of any link causes the graph to lose rigidity. These minimally rigid graphs provide a lower bound on the number of links necessary to constrain a certain number of nodes. For a graph in 3 dimensions, at least $3n - 6$ edges are necessary for minimal rigidity, which can be understood intuitively based on a degree of freedom argument. Each node in $\mathbb{R}^3$ has three degrees of freedom, and each edge removes at most one degree of freedom by imposing a constraint. The final structure has 6 degrees of freedom in its rigid body motion. An infinitesimally rigid graph in $\mathbb{R}^3$ with $3n - 6$ links is minimally rigid, although $3n - 6$ nodes does

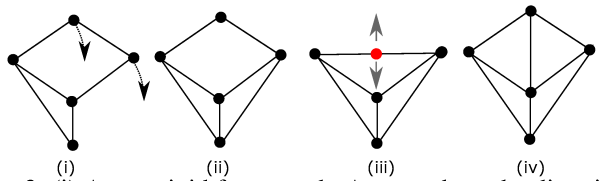

Fig. 2: (i) A non-rigid framework. Arrows show the direction nodes can be moved with no change to lengths. (ii) A minimally infinitesimally rigid network. (iii) The network has the same topology as (ii), and is rigid but not infinitesimally rigid, and hence not controllable. No controlled motion is possible in the direction of the arrows. (iv) An additional edge is added to (ii), meaning the structure is no longer minimally rigid. Motions of the actuators must be coordinated, and can not always be made independently.

not necessarily imply rigidity. Examples of these various classes of rigidity are detailed in Fig. 2.

In the TETROBOT systems the robots were designed such that the system could be decomposed into tetrahedron and octahedral modules that could be analyzed [2]. However, in a more general linear actuator network, there may not be a clear method to decompose the problem. Instead of directly dealing with solving large systems of interdependent constraint equations, we will take a different approach, and repeatedly linearize the system about the current operational point. Such an idea is closely tied to the concept of infinitesimal rigidity. We will show that the node positions of a graph are fully controllable if the framework is infinitesimally rigid.

A. Differential Kinematics and Rigidity

To obtain useful expression of how changing link lengths changes the position of the vertices, the system is linearized about a given configuration by taking the derivative of equation 3, which gives

$$\frac{dL_k^2}{dt} = 2L_k\dot{L}_k = 2(p_i - p_j)^T\dot{p}_i + 2(p_j - p_i)^T\dot{p}_j. \quad (4)$$

This can be rewritten in matrix form as

$$\dot{L} = R(x)\dot{x} \quad (5)$$

In this equation, $R(x)$ is a scaled version of the well known rigidity matrix, an important idea in the study of rigidity [17,18]. Each row of $R(x)$ represents a link L_k. For example, let row m represent the link between nodes i and j. The only non-zero values of row m will be $R(x)_{m,i} = R(x)_{m,j} = \frac{(x_i - x_j)}{\|L_{i,j}\|}$. For a graph in $\mathbb{R}^3$ with n vertices and m edges $R \in \mathbb{R}^{m,n3}$. The maximum rank of R is $3n-6$. A framework is infinitesimally rigid if the matrix $R(x)$ is of maximum rank. Note that infinitesimal rigidity is dependent on the configuration x and is not an inherent characteristic of the graph G. Infinitesimally rigid frameworks are a subset of rigid frameworks, meaning a framework can be rigid but not infinitesimally rigid, but all infinitesimally rigid frameworks are also rigid.

The current analysis is unchanged by any rigid body transformations. We will now consider constraints between the system and the environment. Sufficient constraints must be used to assure that the location of the structure is fully defined (6 relationships when the structure is in $\mathbb{R}^3$). We can encode these relationships in terms of the equation $\dot{F} = C\dot{x}$, where the C matrix relates the motion of the nodes ($\dot{x}$) with the changing environment ($\dot{F}$). The exact form of C can be determined based on how contact between the structure and the environment is modeled. For our purposes, we let each row of C have one nonzero entry that is equal to 1, such that each row of C makes one node of the structure be stationary in one coordinate of the environment.

$$\begin{bmatrix}\dot{L}\\ \dot{F}\end{bmatrix} = \begin{bmatrix}R\\ C\end{bmatrix}[\dot{x}] = H\dot{x}. \quad (6)$$

If the system is infinitesimally minimally rigid, and a minimal set of constraints is applied that is linearly independent of the link constraints, the combined matrix $[R^T C^T]^T$ is full rank and square, and hence invertible,

$$\dot{x} = H^{-1}\begin{bmatrix}\dot{L}\\ \dot{F}\end{bmatrix}. \quad (7)$$

Note that this is exactly the form of a driftless dynamical system. The fact that this matrix is invertible means that the input space is all possible length velocities.

Proposition 1: Given an infinitesimally minimally rigid framework with the minimum number of constraints to the environment, the length of each edge can independently change.

This means that it is not necessary to coordinate movements between lengths as long as the infinitesimally minimally rigid properties are maintained.

B. Controlling Over-constrained Networks

If the system is overconstrained, then the H matrix is skinny, with more rows than columns. Taking the singular value decomposition of the combined H matrix,

$$\begin{bmatrix}U_1^T\\ U_2^T\end{bmatrix}\begin{bmatrix}\dot{L}\\ \dot{F}\end{bmatrix} = \begin{bmatrix}\Sigma\\ 0\end{bmatrix}V^T\dot{x}. \quad (8)$$

The bottom rows of this expression, $U_2^T[\dot{L}^T\dot{F}^T]^T = 0$ is a constraint that encodes how certain lengths must move in a coordinated fashion.

By utilizing this constraint, redundant rows of the H matrix and their corresponding elements in the $[\dot{L}^T\dot{F}^T]^T$ can be removed until it is square and full rank, and hence invertible. Using master/slave terminology, each removed row of H and element of L correspond to that row becomes part of a slave group, while the remaining elements are part of a master group. The reduced H matrix and L vector are represented as H_m and L_m respectively. We denote the removed rows of the matrix as H_s, and the removed link inputs as $\dot{L}_s$, which allows us to express the system as follows:

$$\dot{x} = \left[H_m(x)\right]^{-1}\begin{bmatrix}\dot{L}_m\\ \dot{F}_m\end{bmatrix} \quad (9)$$

$$s.t. \quad H_s\begin{bmatrix}\dot{L}\\ \dot{F}\end{bmatrix} = 0. \quad (10)$$

The system remains controllable, but now the input space is constricted such that only combinations of link velocities that satisfy the constraints can be physically realized. $\dot{L}_m$ can be picked arbitrarily, but $\dot{L}_s$ must be chosen to satisfy the constraint equation. We note that which links chosen as the master and slave may be partially up to the users discretion, and could change based on configuration.

This system can be expressed in the standard form of a linear dynamical system, $\dot{x} = Ax + Bu$ where $A = 0$, $u = [\dot{L}^T \dot{F}^T]^T$, and $B = H^{-1}$. We now make a proposition regarding the controllability of the system:

Proposition 2: A framework that is infinitesimally rigid is controllable.

The controllability matrix is given by $\mathcal{C} = [B, AB, A^2B...A^{n-1}B]$. Because the matrix B is full row rank, the entire system is controllable. This means that for an infinitesimally rigid system, any instantaneous velocity of the nodes can be achieved given control of the rate of change of the link lengths and the contact points. If the contact points are not controlled, (meaning that we cannot control $\dot{F}$), then all unconstrained motions of the system are controllable.

This has the key advantage of allowing us to plan our motion in terms of node positions, and then use the $[R^T C^T]^T$ matrix to determine what input to apply to the actuators.

IV. Locomotion

We define locomotion as controllably moving the center of mass of the system through the environment. For our purposes, we will neglect any inertial forces. While restrictive, if the linear actuator network is moving slowly, the inertial forces will be quite small. We will consider quasistatic locomotion, meaning that at each instant, the center of mass of the system remains inside the support polygon defined by vertices that are on the ground. If, after applying some control, the center of mass leaves the support polygon, then the structure rolls about the edge of the support polygon closest to the new center of mass until the next point is in contact. This process is repeated until the center of mass is inside the support polygon.

Several other studies that developed punctuated rolling type locomotion strategies have focused on developing a gait, or a repeated periodic input to the actuators that, when resolved with environmental forces, causes the robot to move. Our strategy is different in that our controller solves a series of quadratic programs at each time step, based on the robots current configuration, to determine what inputs to apply. This has the advantage of being able to more readily adapt to variability in terrain, whereas a gait could become stuck. A higher level planner, or even a human operator could also provide high level instructions of how to move the center of mass, while the proposed control algorithm could determine how to move the large number of individual actuators to achieve the desired motion. If a method of detecting failed actuators was available, this method could also adapt to failed actuators as it was moving. While these advantages are important, our strategy requires continuous, centralized computation, while following a simple gait strategies could potentially be executed by just having each link follow a preset length profile.

A. Controlling the Velocity of the Center of Mass

The position of the center of mass is defined in terms of the mass matrix of the system, $M \in \mathbb{R}^{3\times 3n}$. if all mass is concentrated at the nodes of the system.

$$\dot{x}_{com} = M\dot{x} = MH^{-1}\dot{L} \tag{11}$$

We can now pick any $\dot{L}$ that achieves a desired motion of the center of mass. The maximum rank of M is 3, so for a system with many vertices MH^{-1} will have more columns than rows, and there is freedom in which $\dot{x}$ is selected to move the center of mass.

B. Constraint Satisfaction

The previous method does not consider the inherent physical constraints in the system. The constraints as previously formulated were based on both the graph, G, and the position of the vertices, x, and can be written in the form of $f(G, x) > 0$. By taking the derivative of the constraints with respect to time, the following expression is obtained, which is linear in $\dot{x}$.

$$\frac{df(G,x)}{dt} = \frac{\partial f(x)}{\partial x}\dot{x} > 0 \tag{12}$$

This means that even though the constraints are nonlinear functions of x, they can be enforced as linear constraints in $\dot{x}$ when they are close to being violated. This insight is used in the following algorithm in order to determine the velocities for the controllers to apply.

Algorithm 1 Constraint-Violation Free Motion

function NextConfig($G, x, \dot{x}_{cm}$)
  D=[]
  Compute $R(x)$
  $Feasible = False$
  while $Feasible == False$ **do**
    $\dot{x}_{temp} = DesiredMotion(D, \dot{x}_{cm})$
    $x_{new} = x + \dot{x}_{temp}dt$
    Active=0
    for $i = 1$ to $N_{constraints}$ **do**
      if $Constraints_i(x_{new}) < 0$ **then**
        D=[D; $\frac{\partial f_i(x)}{\partial x}$]
        $Active = Active + 1$
      end if
    end for
    if $Active == 0$ **then**
      $Feasible = True$
    end if
  end while
  $x(t + dt) = x_{new}$
  $\dot{L} = R(x)\dot{x}_{temp}$
end function

We note that this methodology works for any constraint that can be expressed as $f(G, x)$. In the case of locomotion, we also enforce the constraint that vertices do not penetrate the ground plane. The matrix D is composed of the linearized version of any active constraints. This algorithm ensures that $x(t + dt * k)$ for $k = 1, 2...$, is feasible, because only steps that satisfy constraints are ever executed. However, there

is no guarantee that the transition from $x(t)$ to $x(t+dt)$ also does not violate constraints. With sufficiently small time steps and small buffers this does not seem to pose a problem. In this implementation, the constraints that are checked are the minimum distance between actuators, the maximum and minimum actuator length, and the constraint that points do not penetrate the ground plane.

For fast computation, $\frac{\partial f_i(x)}{\partial x}$ is computed analytically before operation. The most difficult part of this algorithm in terms of computational resources is repeatedly checking the nonlinear constraints. However, this process could be easily parallelized in a future implementation.

C. Objective Function

The proposed algorithm relies on solving the DesiredMotion subproblem on line 6 of Algorithm 1. This problem must return the desired feasible velocity based on some set of linear constraints. By defining this problem as an optimization problem, the system will take the action that instantaneously optimizes some objective, $J(\dot{x})$.

$$\begin{aligned} &\min_{\dot{x}} \ \|J(\dot{x})\|^2 \\ &\text{subject to} \\ &C\dot{x} = 0, \quad M\dot{x} = \dot{x}_{cm}, \quad D\dot{x} \leq 0 \end{aligned} \tag{13}$$

where $C\dot{x} = 0$ represents the contact model, $M\dot{x} = \dot{x}_{cm}$ constrains the center of mass motion, and $D\dot{x} \leq 0$ enforces any active constraints.

One intuitive choice for the cost function is

$$J(\dot{x}) = \|\dot{L}\|_2 = \|R\dot{x}\|. \tag{14}$$

This penalizes large velocities. In the absence of inequality constraints, this solution can be obtained through the psuedo inverse, while a quadratic program solver can be used in the case of inequality constraints.

For mechanical simplicity, it may be desirable to try and move as few links as possible. In this case, a sparse solution could be obtained by minimizing the $L1$ norm of $R\dot{x}$.

Another potential consideration is to try and keep the network as close as possible to a fixed operating point, such as attempting to keep all actuators close to a nominal length l_N. However, as the control inputs are in terms of velocities and not positions, a method is needed to express the objective of equal length actuators, a function of x, in terms of the velocities $\dot{x}$. The positions and velocities can be related by a formation controller presented in [17]. This controller is given by the following control law:

$$\dot{x}_i = \sum_{j \in N_i} (\|x_j - x_i\| - l_n) \frac{(x_j - x_i)}{\|x_j - x_i\|} \tag{15}$$

This can be rewritten as $\dot{x} = -Lx + d$, where L indicates the graph Laplacian of the graph, and the vector d encodes the formation, and is computed at each time step as follows:

$$d_i = \sum_{j \in N_i} \frac{l_n (x_j - x_i)}{\|x_j - x_i\|}. \tag{16}$$

The cost function could then be expressed

$$J(\dot{x}) = \|\dot{x} + Lx - d\|. \tag{17}$$

D. Results

Results obtained from applying this method with the objective function presented in (14) to a network of 108 Linear Actuator and 34 nodes are shown in Fig. 3. In order to demonstrate the ability of the system to follow a trajectory, the network was controlled to move towards waypoints that make up the corners of a predefined trajectory, with the resulting trajectories from using both (14) and (17) as the cost functions shown in Fig. 4. The variance from the exact prescribed trajectory result because of the rolling motion caused when the greedy algorithm caused the center of mass to leave the support polygon. In order to illustrate the effectiveness of this method in preventing constraints from being violated, Fig. 5 shows the lengths of the largest and shortest actuator at each time step, as well as the minimum distance between actuators that do not share a connection for the case of the trajectories shown in Fig. 4. Note that while the constraints are often active, they are not violated. Videos of these test are included in the supplementary material.

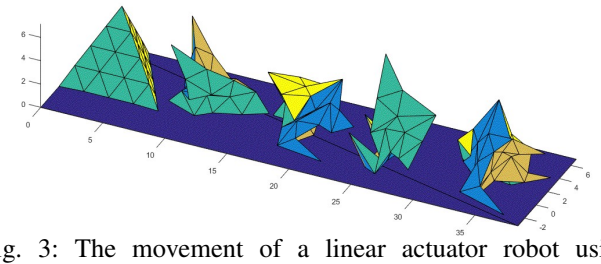

Fig. 3: The movement of a linear actuator robot using Algorithm 1 and the objective function given in equation 14. The system is an overconstrained 3D structure with 108 actuators and 34 vertices

This extended test also gives a sense of the robustness of the network. A downside to the approach of repeatedly solving the quadratic program is that persistent feasibility is not guaranteed. In the case where the objective function was (14), a configuration was reached where the device could not continue to match the desired center of mass velocity without violating constraints. With objective (17) completed the trajectory. It is possible that over the course of the motion, the network reaches a configuration where it cannot continue without violating some constraint. In this case, a feasible solution to the DesiredMotion problem would not exist. Based on simulation results, the persistent feasibility seems to strongly depend on the actuator limits, contact modeling, and other factors. Future work will examine how to analyze the problem of persistent feasibility.

Note that at each time step these methods instantaneously minimize an objective while a desired velocity of the center of mass is obtained. The algorithm can be thought of as being greedy in trying to move the center of mass. However, motion is not necessarily optimal for the entirety of the trajectory. The algorithm does not take into account making and breaking of contact with the surface, which would be required to discuss the optimality of an entire trajectory.

V. Shape Morphing

Here we describe an algorithm to move a Linear Actuator Robot from a given starting configuration to a final configuration that best approximates a target shape. The target

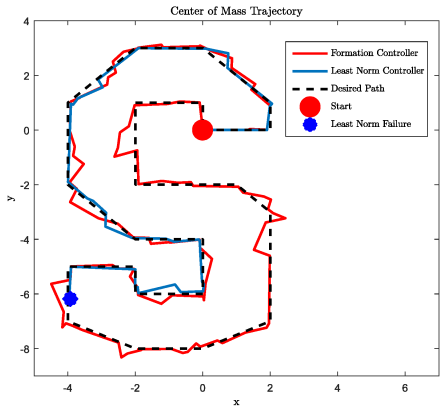

Fig. 4: The path of the center of mass of a LAR as it travels to each waypoint of an "S". When using the minimum norm controller, the LAR fails to complete the trajectory

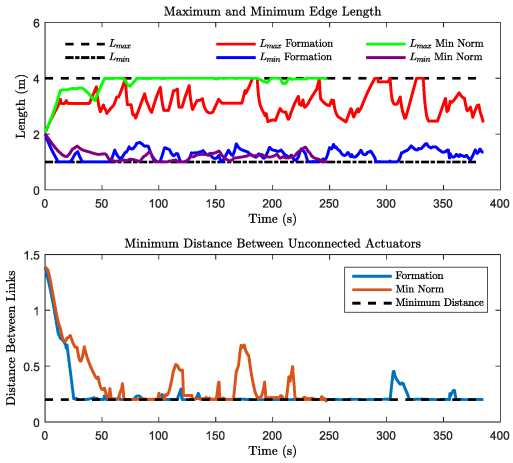

Fig. 5: Plots showing how the lengths of the longest and shortest actuators throughout the simulation shown in Fig. 4, as well as the minimum distance between any two links that do not share a joint. While constraints are active, they are never violated.

shape is defined as a 3D point cloud $q = \{q_1, \ldots, q_m\}$. These points can be obtained directly from a laser scanner, or can be generated by the uniform sampling of a mesh. Our algorithm is inspired by a mesh optimization algorithm first introduced by Hughes Hoppe [19] for mesh optimization.

A. Shape Definition

We must first define the shape of the actuator network. For our cases every vertex on the convex hull of a given initial configuration of the robot is denoted and outer vertex, with their positions given by x_o. The convex hull will be a polyhedron with planar faces. We then take a triangulation of the planar faces to obtain a triangle mesh, $M_o = (K_o, x_o)$, where K_o defines the simplices of the triangulation.

The simplicial complex K_o consists of vertices $\{1...n_o\}$, and subsets of vertices. The 0-simplices are the vertices, the 1-simplices have two elements and are the edges, and the 2-simplices contain three vertices and are the faces. We define the shape of the mesh as the union of the convex hulls of the simplices, and denote this set of points as $\phi_K(x)$. Any point x_m in $\phi_K(x)$ can be defined in terms of a barycentric coordinate vector b such that $b^T x = x_m$, and $b_i \in [0, 1] \forall i$, and $\sum_{i=1}^{n_o} b_i = 1$. This represents every point on the surface as a convex combination of the three points at the vertices of the face in which it lies. Each barycentric coordinate vector b has at most 3 nonzero coordinates, with only 2 nonzero elements if the point lies on an edge of the mesh, and 1 nonzero element if the point is coincident with a vertex of the graph.

While we defined the initial mesh in terms of the convex hull of some nominal configuration, the shape need not remain convex as it undergoes changes. The set of outer faces remains constant while the shape morphs. Physically, defining a fixed set of outer faces could mean that some sort of extensible covering is placed over the outer faces of the robot.

B. Energy Function Minimization

Our algorithm is based on iteratively minimizing an energy function of a similar form to that presented by Hoppe [19], which can be written as

$$E(G, G_o, x, x_o) = E_{dist}(G_o, x_o) + E_{spring}(G, x). \quad (18)$$

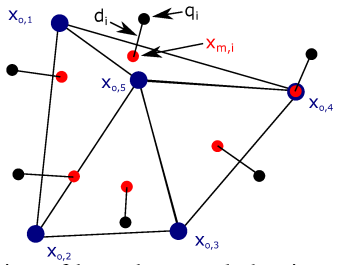

Fig. 6: Illustration of how the sampled points q are projected to the mesh defined by the outer nodes x_o.

We define the E_{dist} term as the sum of all the squared distances between each sample point and the set of all points on the surface of the outer mesh $\phi(x)$:

$$E_{dist} = \sum_{i=1}^{m} d^2(q_i, \phi_K(x)). \quad (19)$$

The spring-energy term is defined as

$$E_{spring}(G, x) = \sum_{\{j,k\} \in G} \kappa \|p_j - p_i\| = \|Sx\|, \quad (20)$$

where S is a matrix that for each row takes the appropriate difference of matrix elements. Note that E_{spring} depends on all nodes of the graph, not just the external nodes.

We then proceed with the morphing algorithm as follows.

1) With fixed vertex positions x, for each sample point q_i find the closest point on the mesh $\phi_K(x)$, and denote that point $x_{m,i}$, as shown in figure 6. Then find barycentric coordinate vectors b_i such that $b_i x = x_{m,i}$.

$$b_i = argmin_{b \in K}(\|q_i - b^T x\|) \quad (21)$$

These coordinates are found by naively projecting each sample point onto all faces, and then finding the barycentric coordinates of the closest point $x_{m,i}$. Each

of these vectors are concatenated to form a matrix B such that $Bx = x_m$

2) For fixed barycentric coordinate vectors B, find the unconstrained gradient descent direction to minimize the function

$$E = \left\| \begin{bmatrix} B \\ S \end{bmatrix} x - \begin{bmatrix} q \\ 0 \end{bmatrix} \right\|_2, \tag{22}$$

and move the vertices of the robot in this gradient direction, $\dot{x}^* = -\nabla E$.

We then use this descent direction together with Algorithm 1 to ensure constraint satisfaction. We use the following optimization for DesiredMotion in line 6 of the algorithm.

$$\begin{aligned} &\min_{\dot{x}} \|\dot{x^*} - \dot{x}\| \\ &\text{subject to} \\ &D\dot{x} \leq 0 \end{aligned} \tag{23}$$

Results obtained from matching a network of 108 actuators and 34 vertices to a cube and an "L" shaped structure are shown in figure 7. While providing good results if initialized well, this method is susceptible to local minima. Future work will focus on how to initialize the network and other methods that may better avoid local minima.

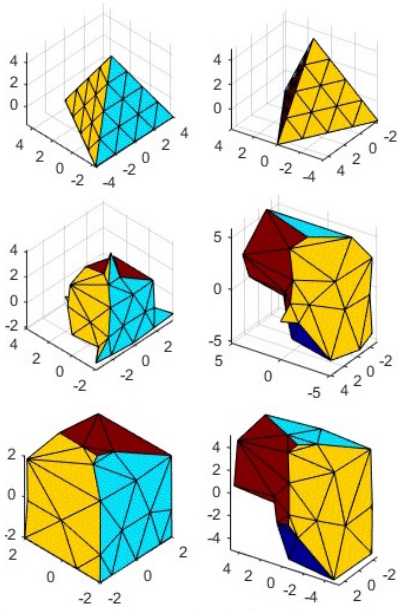

Fig. 7: 3D shape morphing. The left column shows the initial pyramid morphing to a cube, and the right column shows the pyramid morphing to an "L" shape.

VI. Conclusion

We have presented a mathematical definition of a linear actuator robot, together with the description of constraints that ensure that a configuration is physically feasible. We describe the differential kinematics for the system, and draw connections between graph rigidity and controllability. A sequential convex optimization algorithm is presented that allows for the network to move in a direction to decrease while always remaining feasible. This general approach is applied to the task of locomotion, as well as to the task of shape morphing to match a target shape. For this later application, we see the potential to develop a type of robotic graphics, in which large numbers of actuators and principles from computer graphics could enable 3D physical displays. In the future, we will also explore distributed algorithms shape morphing and locomotion.

References

[1] G. J. Hamlin and A. C. Sanderson, "Tetrobot: A modular approach to parallel robotics," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 42–50, 1997.

[2] W. H. Lee and A. C. Sanderson, "Dynamics and distributed control of tetrobot modular robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 4. IEEE, 1999, pp. 2704–2710.

[3] W. H. Lee and A. Sanderson, "Dynamic rolling locomotion and control of modular robots," *IEEE Transactions on robotics and automation*, vol. 18, no. 1, pp. 32–41, 2002.

[4] M. Abrahantes, A. Silver, and L. Wendt, "Gait design and modeling of a 12-tetrahedron walker robot," in *2007 Thirty-Ninth Southeastern Symposium on System Theory*. IEEE, 2007, pp. 21–25.

[5] F. Collins and M. Yim, "Design of a spherical robot arm with the spiral zipper prismatic joint," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2137–2143.

[6] Z. Hammond, N. Usevitch, E. Hawkes, and S. Follmer, "Pneumatic reel actuator: Design, modeling, and implementation," in *International Conference on Robotics and Automation, 2017. ICRA 2017.* Ieee, 2017, pp. 883–888.

[7] C.-H. Yu, K. Haller, D. Ingber, and R. Nagpal, "Morpho: A self-deformable modular robot inspired by cellular structure," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3571–3578.

[8] A. Mazzone and A. Kunz, "Sketching the future of the smartmesh wide area haptic feedback device by introducing the controlling concept for such a deformable multi-loop mechanism," *Links*, vol. 3, p. 248, 2005.

[9] S. Curtis, M. Brandt, G. Bowers, G. Brown, C. Cheung, C. Cooperider, M. Desch, N. Desch, J. Dorband, K. Gregory *et al.*, "Tetrahedral robotics for space exploration," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 6, pp. 22–30, 2007.

[10] C. Paul, F. J. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 944–957, 2006.

[11] A. Sofla, D. Elzey, and H. Wadley, "Shape morphing hinged truss structures," *Smart Materials and Structures*, vol. 18, no. 6, p. 065012, 2009.

[12] A. Lyder, R. F. M. Garcia, and K. Stoy, "Mechanical design of odin, an extendable heterogeneous deformable modular robot," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. Ieee, 2008, pp. 883–888.

[13] W. V. Baxter III, P. Barla, and K.-i. Anjyo, "Compatible embedding for 2d shape animation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 5, pp. 867–879, 2009.

[14] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," in *Proceedings of SIGGRAPH 2000*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 157–164.

[15] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," in *ACM transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 1134–1141.

[16] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[17] L. Krick, M. E. Broucke, and B. A. Francis, "Stabilisation of infinitesimally rigid formations of multi-robot networks," *International Journal of Control*, vol. 82, no. 3, pp. 423–439, 2009.

[18] L. Asimow and B. Roth, "The rigidity of graphs," *Transactions of the American Mathematical Society*, vol. 245, pp. 279–289, 1978.

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in *Proceedings of SIGGRAPH 1992*. ACM, 1993, pp. 19–26.