Motions of Modular Volumetric Actuators

Joaquin Giraldo-Laguna
joaquing@mit.edu

***Abstract*—In robotics, actuation via volume change has become popular and critical to many systems. A spherical electromechanical actuator was developed in 2019 by the Distributed Robotics Laboratory in CSAIL [1]. These actuators are modular and can be linked to form structures and achieve certain dynamic behaviors. Yet, forming large structures physically is difficult and expensive. In this paper, I introduce a framework for generating structures from spherical volumetric actuators to simulate, and discuss how analysis of these systems fosters educated decisions about which structures to generate, and how to control them.**

I. Introduction

As a mechanical engineer, much of my work with robots is physical. When I started working with Lillian Chin on this project in January, I was drawn to the motions of the actuators themselves, and how they work together to create complex behaviors. The concept of cellular volumetric actuation is bioinspired, as plants and several animals change the volume of their cells to move, in a process known as *auxesis* [2]. This concept has been explored in evolutionary algorithms [3], but hardware limitations prevented physical representations from being built.

These auxetic actuators are designed using a polyhedral linkage system, which prevents bending stresses from actuation. Layers of shells of linkages rotate relative to each other to expand. Furthermore, the polyhedral pattern provides centers across three perpendicular axes, which is crucial for designing structures with cubic tiling.

For actuation, two motors force a geometry expansion of the external shell. By using auxetic materials with a negative Poisson's ratio, this shell expands uniformly and simultaneously in every direction. The current model expands to 1.83x its closed radius, or 616% of its original volume.

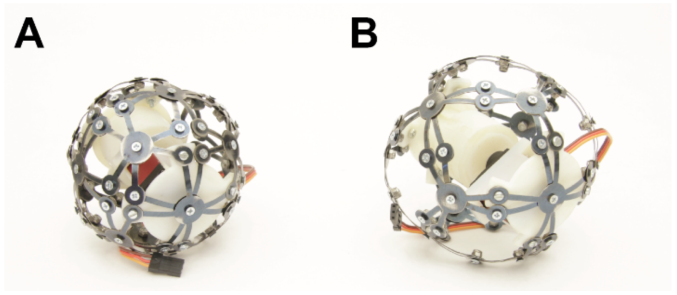


Fig. 1. Previous iteration of the electric volumetric actuator in closed state (A) and open state (B). The actuator is completely actuated in its open state. This iteration expanded to 122% of its original volume.

II. Creating the Simulation

A. Initial Explorations

To create the model, I wanted to ensure that the modularity of the system was conserved. I began by working to modify the rimless wheel example provided in chapter four of the textbook. The notion of controlling through contact is crucial for auxetic actuator structures, since the structures are not fixed to the ground and move by expanding and contracting to generate the right contact forces. I sought to explore the limit cycle of a 2x2 square system of actuators rolling over 90 degrees. The design of the rimless wheel, with radial legs extruding from a center mass for collision, was important for understanding how I could create an auxetic actuator.

After exploring this for a bit, I shifted to building an example using drake in C++. I had some prior experience in building drake from source and using the C++ API to generate deform-able structures, so I figured I would get a more consistent rate of progress. The RimlessWheel model was a bit too defined to modify for a more complex cycle. Navigating the source through bindings was also unwieldy; once I was in one language it became easier to move forward.

B. A Single Actuator

For the 3D simulation of a single auxetic actuator, a system programatically generates the key components of the actuator. Static urdf initializations were avoided to make the system more compatible when generating structures. Each actuator includes:

- A spherical body at the center of the auxetic actuator, whose mass is 4/5 of the total mass.
- Six spherical bodies To create 'spokes' or protruding shell elements for each positive and negative unit axis direction.
- a `Joint` and `JointActuator` for each spoke with the closed and open radii as joint limits.
- Collision geometries for each external spoke.

To actuate each spoke, a wrapper for a `SingleOutputVectorSource` takes a vector of forces whose size is the number of auxetic actuators in the system, and then outputs an actuation vector for the plant which is mapped to send the same force to all joint actuators in a single auxetic actuator simultaneously. The actuators have a mass of 0.4kg, a closed radius of 4cm, and an open radius of 7cm.

Establishing the joint limits for each radius proved here to be difficult, since they would sometimes be ignored unless the time step was decreased. Decreasing the time step and the force on each actuator helped, but joints would seem to settle beyond their limits with some error regardless.

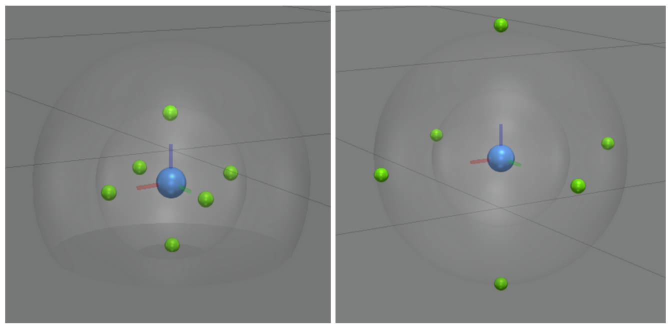


Fig. 2. A single auxetic actuator in its closed position (left) and open position (right) in Drake. The center body is shown in blue, and the spoke bodies in green. The closed and open shell radii are shown with translucent spheres.

C. Actuator Structures

To generate actuation structures, the simulation requires a vector of positions like so:

```
std::vector<Vector3d> box = {
    Vector3d(0, 0, 0),
    Vector3d(1, 0, 0),
    Vector3d(0, 1, 0),
    Vector3d(1, 1, 0),
    Vector3d(0, 0, 1),
    Vector3d(1, 0, 1),
    Vector3d(0, 1, 1),
    Vector3d(1, 1, 1)};
```

Each entry defines the position of an actuator in a cubic unit space. This system, for instance, defines a 2x2x2 box.

To weld neighboring auxetic actuators together, the simulation uses `LinearBushingRollPitchYaw` force elements. Spoke bodies with are occupying the same space as neighbors have no collision geometry. Using this framework, the simulator can create any user defined actuator structure.

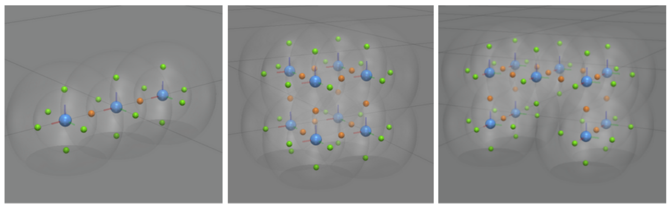


Fig. 3. Three actuator structures, from left to right: 3x1x1 worm structure, 2x2x2 box structure, complex pants structure. Spokes welded to neighboring spokes are shown in orange.

III. Behaviors

Two motions were observed so far using this framework.

First, with the simplest 3x1x1 worm structure, the simulation emulated the actuation sequence used by the physical robots for horizontal crawling.

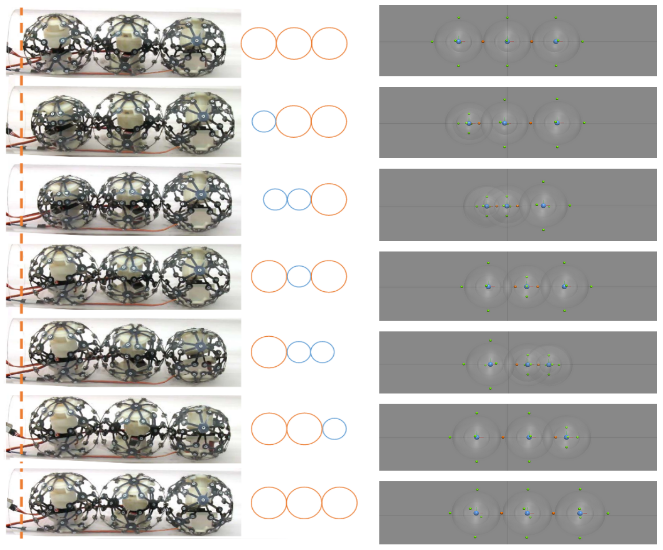


Fig. 4. The actuation sequence for horizontal crawling for the 3x1x1 actuator structure. On the left, the physical model is shown. Expanded cells in the center drawing are orange, and collapsed cells are blue. On the right, The simulated structure with the same sequence.

As expected, the structure was able to move similar to it's physical counterpart and translated 7.17cm in the x direction. This test was used as a baseline to establish a working and representative simulation.

For the rest of the paper, we will be discussing the 2x2x2 box structure and analyzing its actuated rolling behavior. This structure and behavior has intricacies associated with its limit cycle which are worthwhile to explore.

IV. Analysis of the Box

A. Back and Forth

The first demo for the 2x2x2 box structure features two steps of actuation. The first expands two of the actuators in contact with the ground to push off and tip the structure. The second expands a different pair to return to the original position.

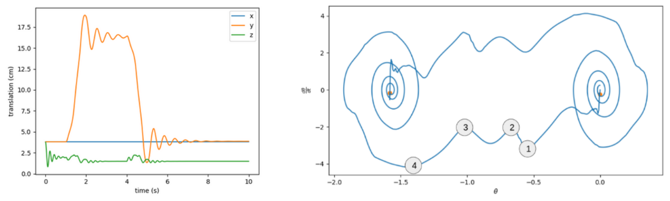
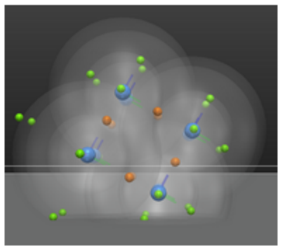
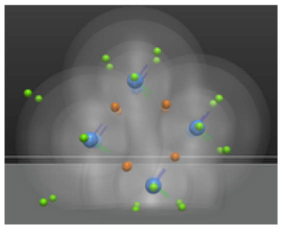
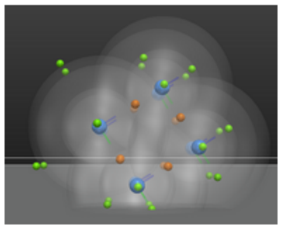
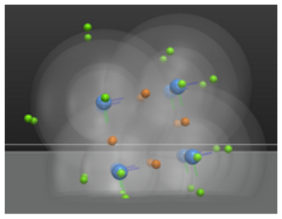


Fig. 5. On the left, the position of the center of mass of the structure during the Back and Forth demo. On the right, the phase plot of the system. Each numbered circle is an event.

The phase plot has two regions of stability, at the base rotation, and at $-\frac{\pi}{2}$. The events which cause jumps in the phase plot are four events like the contact events in models like the kneed walker.

TABLE I
Collision Events in the Roll Cycle

1		The actuated spoke pair stops contact with the ground.
2		The first spoke pair on the face which will be tipped on starts contact with the ground.
3		The center of mass of the structure passes the center of contact between the structure and the ground.
4		The next spoke pair on the tipped-on face starts contact with the ground.

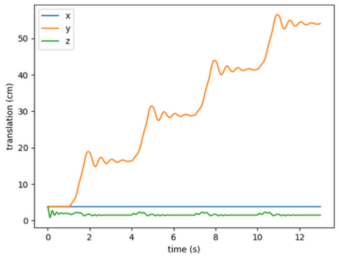
B. Full Cycle

To formulate the box tipping over four times to complete a full rotation, the simple time-based controller actuates each spoke pair sequentially.

V. Control

Unfortunately, I fell into the the hole of spending too much time on the simulation for my project, and therefore controlling the system with something other than the sequential controller is a work in progress. I'll spend this section detailing my plans for controlling this system.

My first approach is to use a finite-horizon linear quadratic regulator to roll the box system into the next configuration. Using the discrete Riccati equations to stabilize a system with an initial rotation of $-\frac{\pi}{2}$, the system can solve for the next rotation. For this implementation, The structure will output the pose of its center of mass as a RigidTransform to the solver, which will then determine the actuation to use for the system



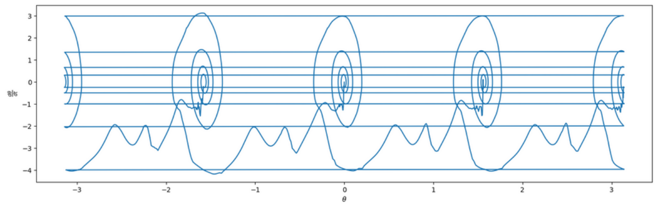


Fig. 6. On the left, the position of the center of mass of the structure during the Full Cycle demo. On the right, the phase plot of the system.

u(t), where the size of ***u(t)*** is the number of auxetic actuators in the system.

This form of model predictive control is best suited for my discrete time system setup. Moving forward with this model, I hope to establish a consistent form of locomotion for the 2x2x2 box structure of auxetic actuators.

Acknowledgment

Thanks to Lillian Chin for introducing me to DRL with this project and allowing me to use it for this class. Thanks to the underactuated staff for the course!

References

[1] J. Lipton, L. Chin, J. Miske and D. Rus, "Modular Volumetric Actuators Using Motorized Auxetics," 2019 IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019, pp. 7460-7466, doi: 10.1109/IROS40897.2019.8968187.

[2] G. K. Muday, "Auxins and tropisms," *Journal of plant growth regulation*, vol. 20, no. 3, pp. 226–243, 2012.

[3] M. Joachimczak, R. Suzuki, and T. Arita, "Artificial metamorphosis:evolutionary design of transforming, soft-bodied robots," *Artificial Life*, vol. 22, no. 3, pp. 271–298, 2016.