Towards Robust Planar Translations using Delta-manipulator Arrays

Skye Thompson[1], Pragna Mannam[2], Zeynep Temel[2], and Oliver Kroemer[2]

***Abstract*— Distributed manipulators - consisting of a set of actuators or robots working cooperatively to achieve a manipulation task - are robust and flexible tools for performing a range of planar manipulation skills. One novel example is the delta array, a distributed manipulator composed of a grid of delta robots, capable of performing dexterous manipulation tasks using strategies incorporating both dynamic and static contact. Hand-designing effective distributed control policies for such a manipulator can be complex and time consuming, given the high-dimensional action space and unfamiliar system dynamics. In this paper, we examine the principles guiding development and control of such a delta array for a planar translation task. We explore policy learning as a robust cooperative control approach, allowing for smooth manipulation of a range of objects, showing improved accuracy and efficiency over baseline human-designed policies.**

I. INTRODUCTION

Distributed approaches to manipulation, such as those seen in automated conveyors, smart surfaces, and planar manipulators, have long served as valuable tools for use in manufacturing and other domains. Such systems offer an appealing flexibility and resilience in their capabilities, achieved through the redundancy and bandwidth afforded by their many cooperating actuators. The design and control of distributed manipulators departs significantly from that of more traditional robots. Many are best-suited to non-prehensile and hardware-dependent manipulations with complex, geometry-dependent dynamics, like vibration [1] or wheeled conveying [2]. Designing these policies can be difficult even when manipulating known objects in a controlled environment, particularly where wear-and-tear or manufacturing error may be present in the manipulator. Developing robust approaches to distributed manipulation, capable of manipulating unfamiliar objects, requires innovation in both hardware design and control.

Our distributed manipulator is composed of a hexagonal grid of delta robots. Individually, a delta robot is a 3 degree of freedom (DOF) parallel mechanism with a fixed base and moving stage, originally designed for pick-and-place and other high-speed, high-precision tasks [3]. With a straightforward design and simple kinematics, the delta is an appealing candidate for mass production. Recent advancements have allowed for the production of these robots at smaller scales, making it feasible to assemble groups of small delta robots for cooperative manipulation tasks [4] [5]. By arranging these deltas in a grid, called a *delta array*, it is possible to achieve a wide range of object manipulations, combining a variety of manipulation strategies requiring both stable contact and dynamic cooperation. The 3 degrees of freedom of each individual delta afford incorporation of both non-prehensile manipulations, like conveying, and contact-based interactions, like grasping or pinning, within a single approach.

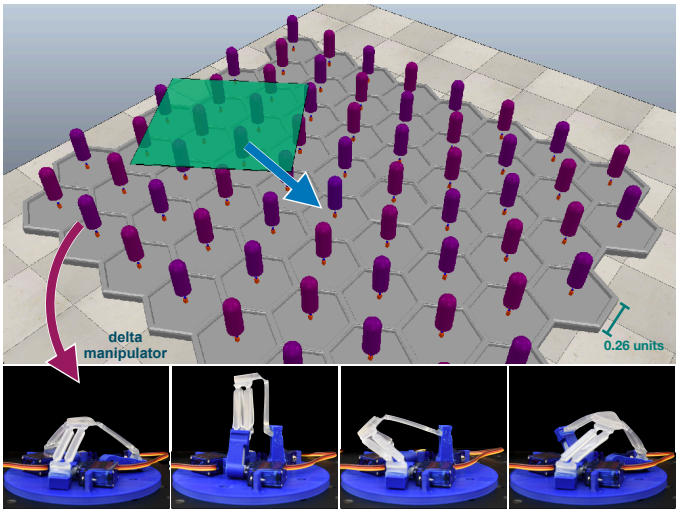

Fig. 1: The delta array was represented by 64 robots with 3 actuators each in CoppeliaSim. The hexagonal packing arrangement minimizes the gap between neighboring deltas' workspaces.

However, it is unclear how well distributed manipulation strategies that have been effective on other forms of hardware translate to use on the delta array. Unlike the continuous interactions of vibration plates and conveyors, delta arrays require the use of finger-gaiting to effectively "walk" the objects across their surface, with individual deltas repeatedly making and breaking contact along the way. The guiding principles behind control for this new type of hardware have not yet been developed. We aim to identify a set of factors that are important to consider when developing cooperative control policies for the delta array, and to establish effective strategies for achieving smooth, efficient, and accurate planar manipulations. We propose to develop these principles through examination and experimentation at each stage of design and control of the manipulator. We begin by analyzing the finger-gaiting strategy necessary to achieve planar translation on the delta array, evaluating how

This material is based upon work supported by the National Science Foundation under Grant No. 1659774 as well as an Amazon Research Award.

[1]Skye Thompson is with CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA, rsthomp@mit.edu

[2]Pragna Mannam, Zeynep Temel, and Oliver Kroemer are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, pmannam,ztemel,okroemer@andrew.cmu.edu

different gait patterns and each individual delta's trajectory impact the manipulation. We then analyze the delta array's performance on a planar translation task, with and without the presence of simulated manufacturing error, and attempt to learn a correction for the introduced performance error. Finally, we explore how the delta array's capacity to execute hybrid manipulation strategies, combining dynamic and static contacts, can further improve translation performance.

II. Related Works

Previous work on the design and control of distributed manipulators has focused primarily on industry applications, and the forms of distributed manipulators most applicable to them. *Yaemglin et. al* [2] focuses on automated conveyors or other industrial systems. Some approaches, such as that used by *Bedillion et. al*, rely on dynamics-dependent models of known objects for consistent control [6], while that used by *Luntz et. al* focuses instead on algorithmic models attempting to guarantee the manipulated object will reach a certain pose, regardless of dynamics [7].

These approaches, while informative, are not clearly analogous to those which would be effective on the delta array. Dynamics calculations that may be feasible for a known set of objects interacting with pins, or the surfaces of spinning wheels, do not translate directly to a system with finger-like points of contact with an object. Additionally, even the most robust of these control approaches, such as the elliptical and squeeze fields explored in *Lunz et. al* [7] and *Bohringer et. al* [8], fail to generalize to objects of certain shapes and sizes, for reasons not fully addressable by their system design.The delta array uses a different model of interaction than these previously explored systems, requiring closed-form contact or finger-gaiting, where not all of the actuators under an object are in contact simultaneously, requiring a different control approach. From this, we conclude the work of exploring the factors to consider in the design and control of a delta array, or similar distributed manipulator, is vital in guaranteeing it's potential as a flexible, resilient manipulation system. The unique traits of such a system - the three degrees of freedom of each delta unit, and the difference in models of contact and control between a delta and a wheel or pin actuator - require a novel investigation.

Another relevant body of work addresses smart surfaces. *Barr et. al* [9] explore the algorithmic and mechanical properties of a grid of single degree of freedom actuators, capable of rising and lowering to manipulate objects on the surface. This design resembles that of the delta array. However, unlike the smart surface, an array of delta robots can translate in the X, Y, and Z axes, granting them a wider range of manipulative capabilities, and requiring a different focus of control to execute those skills. Although the dynamics and kinematics of the delta itself are thoroughly covered in [3], we are not aware of any existing work exploring interactions between delta robots, or the traits and capabilities of such robots operating in unison. In this work, we explore the factors to consider in the design and control of a delta array, or similar distributed manipulator, that are vital in creating a flexible, resilient manipulation system.

III. Delta Array Simulation

The delta array is assembled from a grid of 3-DOF delta robots. Each delta consists of three arms, attached to revolute joints at the base, connected through parallelograms to universal joints at the end-effector. The workspace of each delta robot is hemispherical, proportional to the lengths of the legs, and the end-effector is constrained to a fixed orientation. The *delta array* is composed of deltas packed in a hexagonal arrangement with no overlapping workspaces of adjacent deltas and minimal gaps in the workspace between robots.

For our experiments, we generated a simulated model of an 8x8 delta array in CoppeliaSim, as seen in Figure 1. Each delta manipulator was represented by a hexagonal base and a cylindrical end-effector, controlled by prismatic x-, y-, and z-joints to replicate the delta's 3 translational degrees of freedom. A sphere mounted on the end of the end-effector represented each delta's "fingertip", with which it could interact with the objects. Each finger was constrained to move within the volume above its hexagonal base, which falls within the delta's allowable workspace, preventing it from colliding with its neighbors. This simulation provides a sufficient model of the interactions between the delta array and a manipulated object for our analysis, as well as allowing us to easily simulate additional conditions on the array that may be difficult to replicate across experiments in hardware, such as simulated manufacturing defects impacting the workspace or range of motion of a delta.

IV. Planar Manipulation Policies

Planar manipulation - using a set of actuators to translate or rotate objects in the XY plane - is the intended purpose of common distributed manipulators, such as automatic conveyors or smart surfaces. For our purposes, planar manipulations can both contribute to useful three-dimensional skills, such as re-grasping, as well as provide insight on how the delta array's design can impact its general manipulative capacity. Using the delta array, we also aim to be able to move objects to specific poses on a plane, ideally in a manner that is smooth, quick, and consistent across executions.

A common representation of planar manipulation policies for distributed manipulators is the velocity field [7] [10] [11]. Each actuator in the manipulator that is in contact with the object can be treated as contributing some force to the object to be manipulated, driving the object to a point of equilibrium in the field. This model serves as a good approximation of object behavior for existing distributed manipulation systems, but it is fundamentally dependent on two primary assumptions: 1) the interaction between the objects can be sufficiently approximated as a force originating at the location of each manipulator, and 2) the object is sufficiently large and appropriately shaped that the forces in contact with it aggregate to the desired net force or state of equilibrium. These assumptions hold relatively well for manipulation

systems consisting of wheels, electrical fields, or vibrating plates. However, translation on the delta array is achieved through the use of *finger-gaiting*, a technique in which only a subset of the deltas are in contact with, and therefore applying force to, the object at a given time. This necessitates a closer analysis of the dynamics and control of the delta array, to determine under which conditions this model serves as a suitable approximation for successful manipulation of an object.

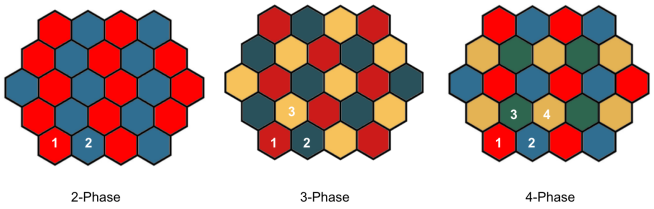

Fig. 2: Three possible phase patterns of finger-gaiting on the hexagonal grid. Each color represents a group of deltas that move together.

We first address the impact of the finger-gaiting approach. Multiphase finger-gaiting is a manipulation approach taking advantage of the degrees of freedom of each manipulator in the delta array, where separate groups of deltas moving together in a staggered pattern, transferring the object between the deltas assigned to a given phase as they oscillate. For a hexagonal packing, we explored three potential gait patterns for a given manipulation - dividing the deltas into two, three, or four groups as seen in Figure 2, where deltas marked by the same color move together, i.e., they are in the same phase of their cyclic movements. Our goal was to determine how the use of different gait patterns impacts the path and behavior of a translated object, and to establish what restrictions the use of each pattern imposes on the set of objects that can be successfully manipulated.

A. Gait Pattern Evaluation

To analyze the different gait patterns' impact on object behavior, we controlled each delta to move in a simple up-down elliptical pattern with constant speed, in the desired direction of translation, with a phase delay corresponding to the gait pattern of the maneuver. The 3D path of each delta is given as

$$x(t) = d_1 \cos(\theta)\cos(\phi + t\pi/f)$$
$$y(t) = d_1 \sin(\theta)\cos(\phi + t\pi/f)$$
$$z(t) = d_2 \sin(\phi + t\pi/f)$$

where d_1 and d_2 are the major and minor axis lengths of the ellipse, angle θ is the direction of translation in the plane, and ϕ and f are the phase delay and frequency of the oscillation respectively. This motion corresponds with a velocity field pointing in a single direction with a uniform magnitude across the surface of the delta array.

First, we compared 2-, 3-, and 4-phase gaits for translating a square object of fixed side length across the surface of the array, as illustrated in Figure 3. We found that gaits with more phases tended to produce smoother trajectories, with less vertical deviation, and a lower average acceleration, likely due to the smaller phase delay between each phase resulting in a more constant velocity, as the handoff between deltas in a two-phase pattern involves stopping the object at the end of each beat. However, the increased spacing between deltas in the same phase caused a different form of instability in the trajectory. We observed that with the higher-phase gaits, the object was more likely to tip due to having fewer points of contact with the array at any given time. We found particularly pronounced instability for smaller objects with fewer overall contacts with the array at any given time.

To explore this further, the translation test was repeated with flat square objects of varying side lengths. The vertical deviation, acceleration, and maximum roll, pitch, and yaw deviations of the object across the trajectory were measured, as shown in Figure 4. Performance on all gaits stabilized when the surface area of the object to be translated was approximately 8 times the area of the delta's simulated hexagonal workspace. This suggests a lower bound on the surface area of manipulable objects, and indicates that use of a higher-phase gait may limit the range of objects that can be reliably manipulated.

V. Individual Delta Trajectories

Having examined how the gait pattern of a delta array may impact its manipulation capacity, we subsequently examined control of the individual deltas within the array. We previously assumed that each delta would perform a simple oscillating elliptical trajectory - but the optimal trajectory may differ significantly from this assumption. We therefore employ reinforcement learning to acquire delta trajectories that result in smooth, stable object motions.

To improve the efficiency of manipulation using the delta array, we aim to learn an improved trajectory that will translate an object quickly and smoothly. We learn this improved trajectory through episodic Relative Entropy Policy Search, or REPS [12], a policy search approach that imposes a constraint on the information loss of each updated policy distribution to its predecessor. We focus on learning a lower-level control policy for the vertical dimension $z_\omega(t)$, parameterized by ω, and keep the x and y trajectories the same as before. Our control policy is represented as a series of N weighted Gaussians centered at equally spaced timesteps

$$z_\omega(t) = \frac{\sum_i^N \omega_i \mathcal{N}(t - \mu_i, \sigma_i)}{\sum_i^N \mathcal{N}(t - \mu_i, \sigma_i)}$$

with time t looping at the end of each cycle. This allows us to parameterize a smooth trajectory with N control points for efficient learning.

Our higher-level policy, $\pi(\omega)$, consists of an N-dimensional Gaussian representing the weights used in the lower-level policy z_t. A set of weights, $\omega_{1..N}$, are sampled from this Gaussian at the beginning of the episode and used to define the cyclical trajectory to be executed. Each

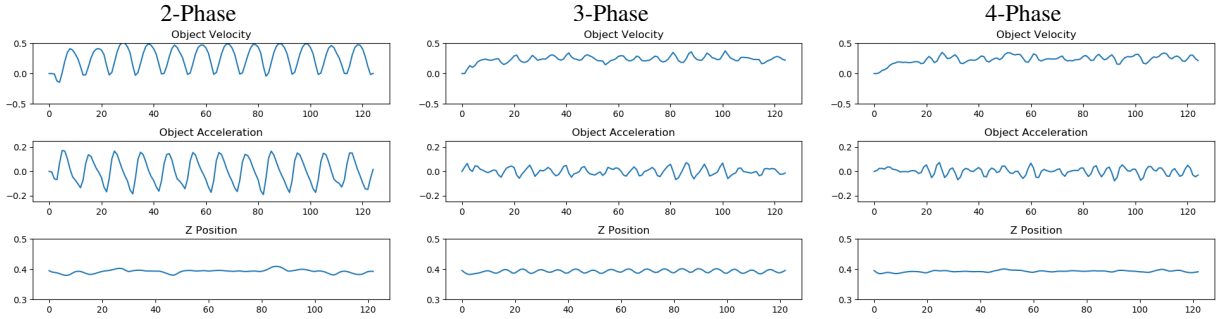

Fig. 3: Analysis of the object trajectory for a planar translation skill using each gait pattern. Higher-phase gaits showed smoother travel, with lower average acceleration and Z-axis deviation of the translated object, regardless of the direction of the translation.

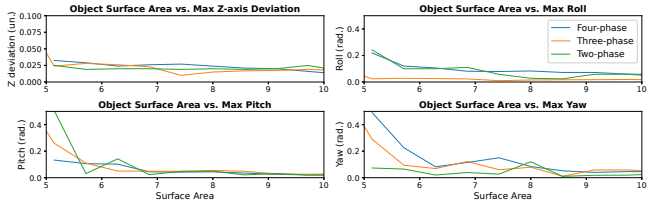

Fig. 4: A comparison of the performance of each phased gait on objects of different surface area relative to the area of the workspace of an individual delta. Performance on all gaits stabilized when the surface area of the object to be translated was approximately 8 times the area of the delta's hexagonal workspace in simulation.

episode consists of a rollout of this policy across a fixed number of timesteps. The return of each rollout $R(\omega) = c_1 v - c_2 h - c_3 a$ considers the average speed of the object in the desired direction v, its maximum vertical deviation h, and its maximum acceleration a, where c_{1-3} are reward factors.

The higher level policy is updated every epoch, and the new distribution $pi(\omega)$ is approximated by performing a maximum-likelihood estimate on the parameters $\omega^{[i]}$, weighted by weights $d_i = \exp(R(\omega^{[i]}/\eta)$. The parameter η is found by minimizing the dual function

$$g(\eta) = \eta\epsilon + \eta \log(\sum_i N^{-1} \left(R(\omega^{[i]})/\eta \right))$$

where ϵ is the manually defined upper bound on the KL-divergence between the previous distributions and the new distribution.

After 30-50 epochs of 5 episodes each, We found the learned policy for all gait patterns converges to a trajectory with a widened, flattened upper half compared to the prior elliptical trajectory, as seen in Figure 5. This flat period is shorter for higher-phase gaits, where the delta spends less time in contact with the object. The period of contact for each learned gait is indicated by the highlighted area. This pattern achieves an object translation approximately 1.5 times faster than the elliptical trajectory.

VI. Cooperative Array Control

We further examined the general principles of control for the delta array by evaluating their ability and effectiveness in performing coordinated manipulation skills. We examined a planar translation task in the XY plane. The goal was to move a flat cuboid object to a specified global position, as seen in Figure 1. The object has a surface area approximately 9 times that of the delta's planar workspace area. The standard velocity field approach for such translation is the construction of a radial attractor field. Given a desired equilibrium point, (x_e, y_e), the velocity value of the field for each delta is defined by

$$V_{xi} = -k_x(x_i - x_e)$$
$$V_{yi} = -k_y(y_i - y_e)$$

where k_x and k_y are positive constants. Each delta can be set in its gaited pattern to move at the direction and magnitude consistent with its position in the field. This approach has been shown to be effective at moving objects to a specified location regardless of starting position on the array [7]. However, there are limitations to this approach on the delta array, given the lack of constant contact between the individual deltas and the object to be manipulated, as well as the the complex and varying dynamics that arise when different numbers of deltas contact the object. To evaluate the delta array's translation performance, we generated a set of translation tasks by randomly sampling a starting location (x_s, y_s) for the object on the surface of the array and a goal location (x_g, y_g) to serve as the center of the attractor. A radial attractor policy was generated with these parameters. For each test, the same task is executed on the array using a two phase gait pattern, as this approach was indicated to be more appropriate for a range of object sizes, and minimized trajectory instability in the manipulation. The delta array does not fully embody the assumptions underlying the velocity field model informing the attractor policy used to complete translation tasks. Thus, we saw a consistent, non-negligible error in the object's final location across testing. The complex dynamics of the interactions between the sets of deltas and the manipulated object, as well as the high dimensionality of the policy itself, make it difficult to explicitly define what changes to this policy could correct for this error.

Additional sources of error may arise in the construction of the array itself - manufacturing error, wear-and-tear, and fatigue across individual manipulators may cause defects

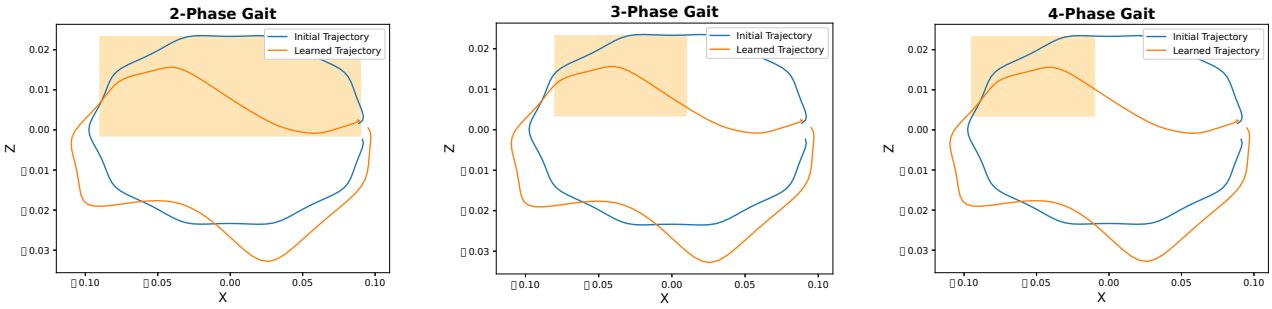

Fig. 5: The learned gait trajectory for each of the possible gait patterns. While the bottom half of the trajectory, where the delta is no longer in contact with the object, is highly variable, the period of contact (highlighted) is consistently smooth and slightly sloped across all the trajectories.

that further violate the assumption of level contact, resulting in further error. To model such hardware variations, we evaluated the idealized attractor policy performance on 15 tasks with and without the presence of a randomized error in the z-position of the base of each delta, representing a manufacturing error impacting the z-dimension of the delta's workspace by some constant offset. The individual errors were uniformly sampled between $(-d_2/4, d_2/4)$, where d_2 is the minor diameter of the baseline elliptical trajectory of the delta. Incorporating these errors resulted in a significant drop in the trajectory smoothness, as shown in Figure 6.

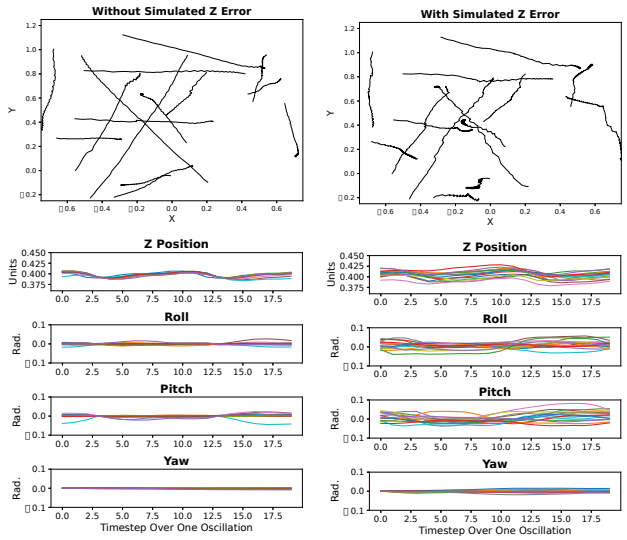

Fig. 6: The trajectories achieved by the radial attractor policy on the delta array, with (right) and without (left) the presence of simulated manufacturing error, represented by a randomized constant offset in the z-position of each delta, are depicted here. The introduction of manufacturing error results in a substantial increase in the instability of the trajectory.

A. *Improving Accuracy through Learning*

To correct for these hardware variations, we proceeded to determine whether an adjusted attractor policy could be learned to achieve more accurate manipulations. We learned an inverse model, predicting the necessary policy to translate the object towards a proposed goal, in a self-supervised manner. We considered both a densely connected MLP neural net, and a transpose convolutional neural net, in order to determine whether the latter would be able to take advantage of the grid structure of the array for more accurate or efficient learning. Both networks take the goal location, (x_i, y_i), normalized to zero mean and unit variance, as input, and outputs the policy parameters for each delta as two 8x8 arrays representing the X- and Y- velocity components, (V_x, V_y). The transpose convolutional network consists of an input layer, a densely-connected layer, two transpose convolutional layers with a (2×2) kernel and stride 2, while the densely connected network contains two fully connected hidden layers with 10 units each. Both networks have an output layer with linear activation, and were trained with a standard mean squared error loss. To generate our training set, we executed 1000 additional attractor tasks, randomly sampling an object starting location (x_s, y_s), and a location for the center of the attractor (x_c, y_c). Note that we only sample the attractor center here, not the goal itself. This is to account for error in the hand-designed policy - for an executed policy, we assign the 'true' goal location to instead be the location of the object after 400 timesteps, regardless of the position of the attractor's center for that execution. This data is assembled into a set of training inputs x_g, y_g and outputs, V_x, V_y represented as 8x8 matrices of the x- and y-magnitudes of each delta's component of the attractor field.

	Mean Error	Mean Error with Z-defect
Ideal Attractor	0.143 ± 0.093	0.197 ± 0.076
Learned MLP Policy	0.129 ± 0.011	$0.124 \pm .012$
Learned TC Policy	0.114 ± 0.011	0.128 ± 0.012

We evaluated the learned model by sampling a goal location (x_{gi}, y_{gi}) and predicting the appropriate policy to translate an object to that location as the model's output. We perform two tests comparing the performance of the hand-designed and learned policy on the task with and without the presence of simulated manufacturing error. We calculate the error for each rollout as the Cartesian distance between the goal and the object's location after 400 timesteps.

We find that both of the learned policies were successful at reducing the error resulting from the simulated z-defect, indicating that with some adjustments, the radial attractor model is an appropriate choice for a translational policy on

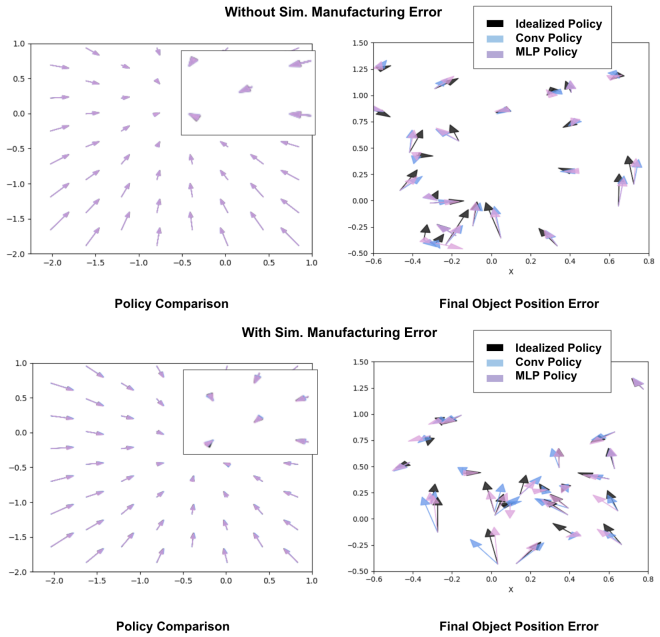

Fig. 7: A comparison between the learned and hand-designed attractor policies is depicted on the left. As shown by the close-up image, only very minor differences exist between the learned and hand-designed policies. The comparison between the final error in object position between the two policies is shown on the right, as an arrow pointing from the intended final object position to the actual final position achieved. The learned policy generated by both the simple and transpose convolutional networks is able to correct for some of the error present in the idealized policy in both cases.

the delta array. The transpose convolutional approach fails to correct as well as the policy learned on the densely connected network. We attribute this to the nature of the error, which differs independently across individual deltas. The transpose convolutional structure imposes some additional structure that nearby deltas behave similarly in the final policy, due to the influence of shared parameters in the previous layer. This structure seems to fail to correct sufficiently for this independent variation.

B. Hybrid Control Approaches

Due to each deltas' three degrees of freedom, it is possible to combine both static contact and non-prehensile manipulations, like finger-gaiting, in the same approach. For a translational task, this can be achieved by raising (in the Z-axis) a subset of the deltas to serve as a "wall" against which an object can be pushed to constrain its final position and orientation. We constructed a general approach for designing such policies, and compare its performance to that of the radial attractor on the delta array.

We used the same task definition as in the attractor, providing a randomized start position (x_s, y_s) and goal position (x_g, y_g) for the object. For this control approach, the position of the attractor is on the opposite side of the wall, such that it draws the object into contact with the wall. To generate suitable attractors and walls we used a rejection sampling approach: We randomly sampled contiguous lines in the delta arrays, separating the start position and the attractor center, to serve as a wall. Any line that was too close to the start position was rejected to avoid the object starting on the wall.

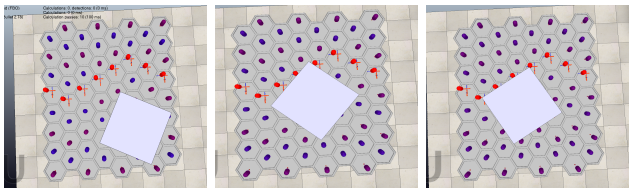

Fig. 8: These images provide an example of a manipulation policy incorporating hybrid control. The object is drawn towards the wall, which serves as a constraint on the object's position and orientation.

We use the above approach to generate 2000 training samples for an MLP policy. The network outputs an additional 8x8 array with binary values indicating the wall deltas. The approach was evaluated on 20 additional tasks and found to have an error of 0.418 ± 0.053 units. This performance is significantly worse than that of the attractors without walls, even when incorporating knowledge of the object's size to position walls continuously rather than discretely in the space, allowing walls to shift within the comprising deltas' workspaces to accomodate the object (resulting in only a 0.120 units reduction of error). The decrease in performance is potentially the result of the more complex interactions between the object and the wall and the presence of multiple solutions to each task resulting in an ill-posed problem. In the future, we plan to explore distal teacher approaches to improve the learning approaches in order to achieve more accurate object control.

VII. Conclusion

In our experiments, we identified several principles and tradeoffs to guide development of the delta array manipulator, including optimal gaiting approach, planar manipulation policies, and learned control. We determined that with consideration of the gaiting pattern and the learned adjustments to the individual delta trajectories and the overall policy, the velocity field model is effective for achieving planar translation tasks. We proposed a hybrid control approach unique to the delta array, demonstrated the construction of such a policy, and identified sources of possible error in its use. We present these results to inform further use and study of such systems. The preliminary approaches to learning examined here suggest that it may be possible to learn hierarchical policies for achieving better performance with the manipulator. Future work should also consider the dynamics of the delta robot in the array and the effects of their compliance on control.

ACKNOWLEDGMENT

Special thanks to the RISS program, the Robotics Institute at CMU, and the Intelligent Autonomous Machines Lab for their support.

REFERENCES

[1] K. . Bohringer, V. Bhatt, and K. Y. Goldberg, "Sensorless manipulation using transverse vibrations of a plate," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 2, 1995, pp. 1989–1996 vol.2.

[2] T. Yaemglin and S. Charoenseang, "Distributive behavior-based control for a flexible conveying system," *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT '02*, pp. 24–29 vol.1, 2002.

[3] M. Lopez, E. Castillo, G. Garcia, and A. Bashir, "Delta robot: inverse, direct, and intermediate jacobians," *Journal of Mech. Eng. Science*, pp. 220:103–109, 2006.

[4] H. McClintock, F. Z. Temel, N. Doshi, J. S. Koh, and R. J. Wood, "The milliDelta: A high-bandwidth, high-precision, millimeter-scale Delta robot," *Science Robotics*, vol. 3, no. 14, 2018.

[5] J. E. Correa, J. Toombs, N. Toombs, and P. M. Ferreira, "Laminated micro-machine: Design and fabrication of a flexure-based Delta robot," *Journal of Manufacturing Processes*, vol. 24, pp. 370–375, 2016.

[6] M. Bedillion, R. Hoover, and J. McGough, "A distributed manipulation concept using selective braking," *2014 American Control Conference*, pp. 3322–3328, 2014.

[7] J. E. Luntz, W. C. Messner, and H. Choset, "Distributed manipulation using discrete actuator arrays." *The International Journal of Robotics Research*, pp. 20:553–583, 2001.

[8] K. Bohringer, B. Donald, R. Mihailovich, and N. C. MacDonald, "A theory of manipulation and control for microfabricated actuator arrays," *Proceedings IEEE Micro Electro Mechanical Systems An Investigation of Micro Structures, Sensors, Actuators, Machines and Robotic Systems*, pp. 102–107, 1994.

[9] D. R. W. Barr, D. Walsh, and P. Dudek, "A smart surface simulation environment," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4456–4461, 2013.

[10] J. E. Luntz, W. Messner, and H. Choset, "Velocity Field Design on the Modular Distributed Manipulator System," *Robotics: The Algorithmic Perspective*, 1998. [Online]. Available: http://www.ri.cmu.edu/pubs/pub_2868.html%5Cnpapers://dc655b36-91f1-4c2b-88b1-0972dfc51324/Paper/p271

[11] K. Varsos and J. Luntz, "Superposition methods for distributed manipulation using quadratic potential force fields," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1202–1215, 2006.

[12] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, pp. 57–58:130–136, 2013.

[13] M. Bedillion and W. Messner, "Trajectory tracking control for actuator arrays," *IEEE Transactions on Control Systems Technology*, pp. 21(6):2341–2349, 2013.

[14] K. F. Bohringer, H. Choset, and H. Choset, *Distributed Manipulation*. Springer Science Business Media, 2000.

[15] M. Sinclair and I. A. Raptis, "Distributed manipulation using cyber-physical systems," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3097–3102, 2014.