Energy Gradient-Based Graphs for Planning Within-Hand Caging Manipulation

Walter G. Bircher, *Student Member, IEEE,* Andrew S. Morgan, *Student Member, IEEE,*
Kaiyu Hang, *Member, IEEE,* and Aaron M. Dollar, *Senior Member, IEEE*

***Abstract*— In this work, we present a within-hand manipulation approach that leverages a simple energy model based on caging grasps made by underactuated hands. Instead of explicitly modeling the contacts and dynamics in manipulation, we can calculate a map to describe the energy states of different hand-object configurations under an actuation input. Since the system intrinsically steers towards low energy states, the object's movement is uniquely described by the gradient of the energy map if the corresponding actuation is applied. Such maps are pre-calculated for a range of actuation inputs to represent the system's energy profile. We discretize the workspace into a grid and construct an energy gradient-based graph by locally exploring the gradients of the stored energy profile. Given a goal configuration of a simple cylindrical object, a sequence of actuation inputs can be calculated to manipulate it towards the goal by exploiting the connectivity in the graph. The proposed approach is experimentally implemented on a Yale T42 hand. Our evaluation results show that parts of the graph are well connected, explaining our ability to successfully plan and execute trajectories within the gripper's workspace.**

I. Introduction

Manipulating a grasped object within the hand is an important functionality for many practical tasks, especially for instances where the grasp type must be changed without releasing the object, such as changing from a fingertip grasp to a palmar grasp. Nearly all within-hand manipulation (WIHM) tasks involve changing the contact location on the hand or object, which will typically involve some amount of sliding. Rather than directly modeling the complex frictional properties and behaviors at contact in these scenarios, we instead seek to create a scenario in which the object is passively prevented from being ejected from the grasp (i.e. it is "caged" grasp [1] [2]), and the manipulation is actively guides the object in the desired directions by shaping the potential energy of the underactuated fingers. In this way, we can ensure that the object moves towards the desired target without needing precise information about the contact forces and frictional properties.

Traditional approaches to this type of problem rely strong assumptions about the nature of contact—namely being able to precisely model the contacts between the robot and object in order to enable effective control [3][4][5][16]. By relaxing the rigid constraints in a grasp, objects can be manipulated by rolling contacts on its surface based on kinematic trajectory optimization [1]. In an object-centric formulation, a virtual-frame can be derived to enable impedance control to implicitly regulate contact forces during manipulation [6]. Using tactile feedback from the fingertips, grasp stability can be estimated online to inform the system so that force adaptation and finger gaiting can be utilized to prevent the system from dropping the object [7], [8]. Although these approaches can sometimes successfully reconfigure the object within hand, they are vulnerable to external disturbances and require great mechanical and computational complexity to maintain the grasp and often fail due to uncertainty or errors in the required sensing and control.

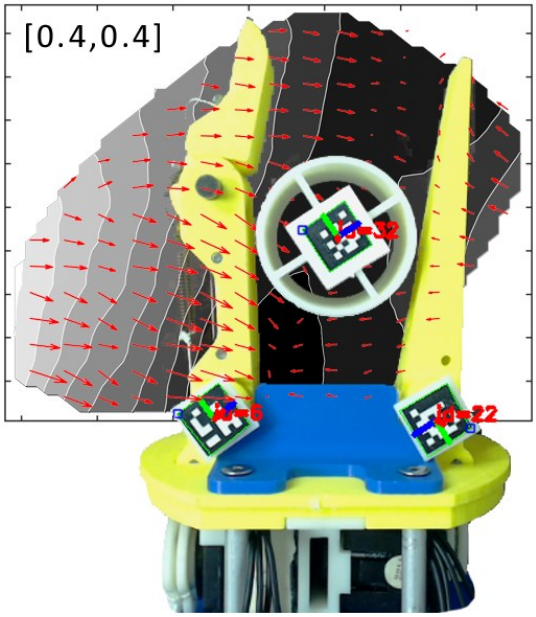

Fig. 1. A Yale OpenHand T42 gripper and associated energy map (contour plot). The grid of gradient vectors of the energy map (red arrows) show the possible motions that can be applied to an object at each location with a hand, for a specific actuation input [0.4, 0.4].

Rather than using high Degree of Freedom (DOF) hands for manipulation, extrinsic dexterity has enabled simple grippers to reconfigure an object with larger motions by exploring external contacts [9]. To understand how an object can be manipulated by external pushing, motion cones have been proposed to represent feasible actions applicable to an object [10]. Moreover, by analyzing the geometries of objects to model the feasible translations and rotations of contacts on an object surface, dexterous manipulation graphs have been proposed to plan a sequence of pushing actions in a dual-arm formulation [11]. Nevertheless, this class of approaches

Research supported by the U.S. National Science Foundation under grant IIS-1734190 and IIS-092856.

W. G. Bircher, A. S. Morgan, K. Hang, and A. M. Dollar are with the Dept. of Mech. Eng. and Mater. Science, Yale University, CT, USA, ({walter.bircher andrew.morgan; kaiyu.hang; aaron.dollar}@yale.edu).

requires complicated geometrical analysis of both the object and the environment, and generates large motions of the arm.

Instead of using a force feedback-based grasp for manipulation, caging grasps can be used as a very robust means to guard against external disturbances [12]. Using topological representations, neck or fork structures in an object can be detected to enable caging by simple grippers [13]. Moreover, loop-grasping can also be synthesized by modeling a Writhe Matrix between the loops in an object and robot links [14]. Caging has also been used for "blind" (open loop) within hand manipulation using a fully defined model of the hand [17]

In our previous work, based on caging grasps and the passive reconfigurability of underactuated hands, we have developed an energy map to implicitly represent the mapping between the hand-object configuration and the actuation inputs [2]. This enables us to understand the energy status of the hand-object system, as well as how the object can move towards a lower energy configuration under certain actuation inputs. Based on those energy maps, in this work we develop a graph representation to model the connectivity among the object positions in the hand's workspace. In brief, based on a set of energy graphs calculated from different actuation inputs, the graph is constructed by exploring the map's gradient directions, along which the object moves under different actuations. The graph is then used to plan a sequence of actuation inputs to reconfigure the object towards a given goal configuration, while attempting to maintain the object in a caging grasp.

In the following sections of this paper, we will begin by describing the modeling of energy maps in Sec. II. In Sec. III, we detail the construction of energy gradient-based graphs, as well as how to plan and execute actuation sequences for within-hand manipulation. This section also introduces the system implementation, control, and analysis. We then are able to experimentally implement and evaluate our approach in Sec. IV. In the end, we conclude and discuss future works in Sec. V.

II. Energy Model

In this paper we utilize a planar energy based caging model first presented in [2] to translate symmetric cylindrical objects. In short, this work combines a linkage based caging model with a method that computes the total energy of the hand-object system in each kinematically feasible state. Whereas a traditional caging model assumes immovable rigid obstacles, we instead acknowledge that obstacles can be moved for a cost. This assumption is valid for two reasons. First, it is valid because we apply this model to an underactuated hand with compliant elements each with a number of kinematically admissible configurations, and each storing different amounts of energy. Second, we consider actuators to be backdrivable, treating them as linear springs around their commanded setpoints. In other words, if you do work to rotate the shaft of an actuator operating in position control mode, you can change its position. With these assumptions in mind, we use an extended caging formulation, beyond the more traditional purely kinematic analysis, by also considering the energy associated with movable obstacles.

By definition an object is caged if it cannot be moved to a point at infinity without first intersecting other objects in its workspace. In general, this corresponds to a point contained within a closed, isolated volume in configuration space. In this work we narrow our scope to the case of a planar object being caged by the links of a planar gripper. We adopt notation from the caging formulation described in [15] and consider caging configurations that minimize the object's configuration space. We do this by making the strong assumption that there are no dissipative forces in our system, and that a stable grasp on an object, representing a single actuation input combination, is associated with an energy minimum configuration. In other words, we assume that for a given object position, there is some combination of actuator inputs that minimizes the system's energy, somewhere in the feasible range of joint configurations that adhere to the physical contact constraints between the links and the object. This allows us to consider a manipulable caging grasp, meaning that the hand can be reconfigured into other non-caging configurations with non-zero work done on the object. To formulate the energy minimization, we follow previous work from [2], computing energy values specifically for caged configurations of our system:

p_k: reference position for joint or element k, either p_A for linear actuators or θ_A for rotary actuators, p for joints

f_A: force from actuator A (for linear actuators)

τ_A: torque from actuator A (for rotary actuators)

The actuation energy associated with a given reference value a_k can be expressed as the following for rotary actuators:

$$E_{Ak}(\theta_k) = -\tau_{Ak}(\theta_k - \theta_{Ak}) = -f_{Ak} r_{Ak}(\theta_k - \theta_{Ak}) \quad (1)$$

Or, in the case of linear actuators:

$$E_{Ak}(p_k) = -f_{Ak}(p_k - p_{Ak}) \quad (2)$$

Then, the total energy associated with a configuration of the hand is written as the summation of the energy for all actuators in the system:

$$E_A(a_{hand}) = \sum_{k}^{N} max(E_{Ak}(a_k), 0) \quad (3)$$

where a_k is the position controlled actuation input. The *max* function is used to select only positive energy values (see [2]). The system energy, which depends only on the configuration of the hand, is computed using 3 at each caged object *xy*-position in front of the hand. As the object is virtually placed throughout the workspace, the hand's configuration adjusts to maintain contact. Thus, the hand's configuration would change if the object were forcefully placed in its path, doing energy against the actuators. This workspace of energy values forms a contour plot, similar to that shown in Fig. 1. We refer to this bounded contour plot containing system energy values as an Energy Map $M_i = f(E_A(x, y)), \in \mathbb{R}^2$. A single energy map can be computed for every combination of actuation inputs, as described in [2], and as illustrated in Fig. 2. We

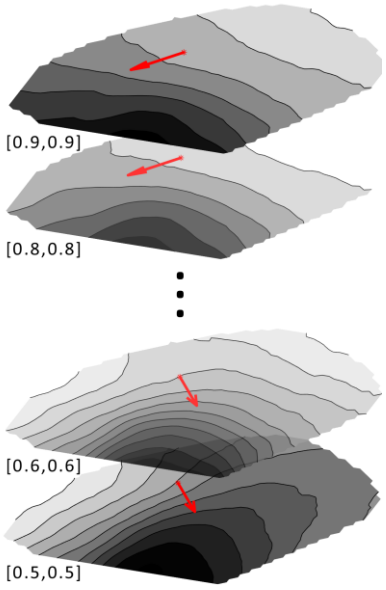

Fig. 2. Given the geometry of a hand and corresponding object, we compute offline a library of energy maps—one for each unique combination of actuation inputs, over the range of possible actuation inputs for the motors of the hand.

extend our previous work by numerically computing the gradient vector field γ_i of a simulated energy map

$$\gamma_i = -\nabla_{x,y} M_i \tag{4}$$

For a given hand-object configuration and a given actuator input set, γ_i can be visualized as a vector field overlaid on the workspace of the hand, with all vectors flowing towards the lowest system energy. An example is shown in Fig. 1.

III. Energy Gradient-Based Graph

In this section, based on the set of obtained energy maps $\mathcal{M} = \{M_1, \dots, M_N\}$, we will first introduce the construction of the energy gradient-based graph, and then use the constructed graph to plan actuation input sequences to move the object between positions in the workspace. In our case we utilized a set of $n = 100$ energy maps corresponding to 100 actuation input combinations.

A. Graph Construction

As the goal of the energy gradient-based graph, given an object's position within the hand's workspace, we wish to represent whether the object can be translated from its current position to any of the neighboring positions, and whether new translations can be derived recursively to expand over the entire reachable workspace. For this reason, in order to keep the problem tractable, we discretize the hand's workspace into a grid as illustrated in Fig. 3.

Concretely, the grid contains a set of nodes $\{n_i | i = 1, \dots, K\}, n_i \in \mathbb{R}^2$, representing a finite set of positions in the workspace. Each node is associated with 8 neighbor nodes, denoted as $\rho(n_i)$, in the 8 cardinal directions. Using the energy maps, an energy gradient, $\gamma(M_l, n_i) \in \mathbb{R}^2$ as defined in (4), can be calculated on a node n_i using any of the $M_l \in \mathcal{M}$. As such, we can obtain a set of energy gradients, denoted as $\Gamma_i = \{\gamma(M_l, n_i) | M_l \in \mathcal{M}\}$, for each node n_i on the grid. As shown in Fig. 2, the energy gradient vectors point to different directions, indicating potential translation directions resulting from the corresponding energy maps. Since each M_l is associated with a certain pair of actuation inputs, Γ_i actually represents how an object can be locally translated by applying different motor actuations. To this end, we are now interested in connecting the nodes n_i on the grid based on the energy gradient vectors Γ_i, in order to investigate how the object can be manipulated by translating among the grid points. Note that the gradient vector $\gamma(M_l, n_i)$ is calculated in a continuous space and not aligned with the grid, they need to be projected in order to match the discretization. As depicted in Fig. 3, along the 8 cardinal directions defined at each node, given a gradient vector $\gamma(M_l, n_i)$ at n_i, the object can be translated to its neighbor node n_j if $\gamma(M_l, n_i)$ points to a similar direction as $\overrightarrow{n_i n_j}$.

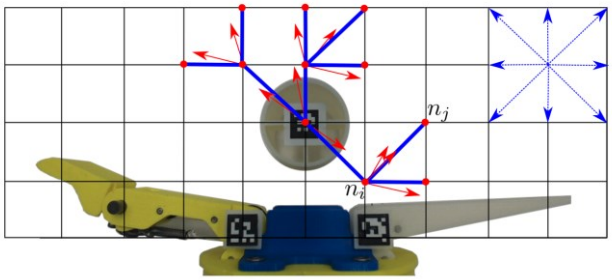

Fig. 3. Illustration of graph generation. In the figure, red points and blue lines are nodes and edges in graph G. Red arrows are the gradient vectors $\gamma(M_l, n_i) \in \mathbb{R}^2$ calculated from energy maps.

Formally, the energy gradient-based graph, denoted by $G = (E, V)$, with nodes V and edges E is constructed by:

$$E = \left\{ (n_i, n_j) \,\middle|\, \exists \gamma(M_l, n_i) \in \Gamma_i, \delta\left(\overrightarrow{n_i n_j}, \gamma(M_l, n_i)\right) \leq \epsilon \right\}$$
$$V = \{ n_i \mid \exists\, n_j \in \rho(n_i), (n_i, n_j) \in E \ \vee (n_j, n_i) \in E \} \tag{5}$$

where $\delta(\cdot)$ calculates the angle difference as:

$$\delta\left(\overrightarrow{n_i n_j}, \gamma(M_l, n_i)\right) = arccos \frac{\overrightarrow{n_i n_j} \cdot \gamma(M_l, n_i)}{|\overrightarrow{n_i n_j}||\gamma(M_l, n_i)|} \tag{6}$$

We can see that G is a directional graph, and the threshold ϵ determines how accurately an edge describes possible motion between 2 nodes. It is possible that some nodes on the grid are not in the graph if there is no edge connecting to them—for example if a node on the boundary of the workspace has a gradient vector that points outside of the caged region, rather than to other nodes. In addition, as will be evaluated in experiments, in the graph there can be multiple connected components, which means that each node can be reached from a certain subgraph, and that it is possible that there is no path connecting a pair of nodes in the graph.

B. Motor Actuation Planning and Execution

Having constructed the energy gradient-based graph G, we are now able to plan a sequence of actuation inputs in order to manipulate the object to translate along the edges in E.

Algorithm 1 Planning and Execution

Input: $G=(E,V)$, n_g, m, $maxIter$
Output: *Status*

while $maxIter \geq 0$ **do**
  $n_s \leftarrow observePosition()$
  if $\|n_s - n_g\| \leq \eta$ **then**
    return Success ▷ Reached
  end if
  $N_s \leftarrow KNN(n_s, m)$
  $N_g \leftarrow KNN(n_g, m)$ ▷ Snapping
  $P = constructPairs(N_s, N_g)$
  for all $p \in P$ **do**
    $(n_s^i, n_g^j) \leftarrow p$
    $\pi = G.FindPath(n_s^i, n_g^j)$
    if $\pi.found()$ **then**
      $L^* \leftarrow GetActuation(n_s, \pi)$
      $Execute(L^*)$ ▷ Execution
    else
      return Failure ▷ No Path
    end if
  end for
  $maxIter = maxIter - 1$
end while
return Failure ▷ Too Many Steps

Concretely, letting $n_s, n_g \in V$ be the initial and goal positions of the object, we aim to find a path $\Pi = \{n_s, \pi_1, \dots, \pi_T, n_g\}, \pi_1 \in V$, such that the goal position can be reached by executing the actuation input associated with each edge (π_t, π_{t+1}).

In reality, however, the start and goal positions n_s, n_g are in continuous space and unlikely to be exactly on the grid. Therefore, before finding the path Π we snap the continuous positions to the grid points using the K-Nearest Neighbor (KNN) algorithm. In order to increase the planning success rate, rather than finding one nearest neighbor for each, n_s and n_g are snapped to m nearest neighbors by the function $KNN(n_i, m)$ to generate a set of m start candidates $\{n_s^1, \dots, n_s^m\}$, and m goal candidates $\{n_g^1, \dots, n_g^m\}$ in the graph. The candidates are ordered by their distances to the original point. During planning, we try to find a path by iterating through each pair of start and goal candidates and returning a path as soon as the first path is found for a pair n_s^i, n_g^j.

For executing the path, in order to compensate for the inaccuracies caused by snapping and discretization, the motor actuation for the first edge (n_s^i, π_i) in Π is not directly found by its associated actuation input. Instead, we calculate the actuation input based on the original n_s and the first waypoint π_1. The index of the actuation input at the start point n_s is found by:

$$L^* = \underset{l}{\operatorname{argmin}}\ \delta\left(\overrightarrow{n_s \pi_1}, \gamma(M_l, n_s)\right) \tag{6}$$

Furthermore, to compensate for noise and execution errors, although we have an entire path planned, we in practice execute only the first actuation input L^*. Thereafter, our system will re-observe a new n_s and replan a path, from which again only the first actuation is executed. In this manner, our system will iteratively move the object towards the goal position, while being able to adjust its behavior on the way by online re-planning. The planning and execution based on the energy gradient-based graph is summarized in Algorithm 1.

For path finding in the graph, we use breadth-first search to find the shortest path. We can see that the manipulation is limited to take at maximum $maxIter$ steps. Once the difference between real-time observed object position and goal position is smaller than η, we consider it as a success. A failure can occur if a path cannot be found to connect start and goal, or the maximum number of executions has been exceeded.

C. System Implementation

This framework was physically implemented and tested by utilizing an overhead camera observing the manipulation. First, state detection processes determined the physical location of the gripper through the use of ArUco markers attached to the base frame of the hand. The origin of the workspace was then calculated, which is specified to be directly between the base frame joints of the hand. It is important to note that the object can move in the positive and negative x direction, but can only in the positive y. At this point, the object's physical location was then extracted with respect to the defined origin of the gripper. From this information, we were able to define the current location of the object which is utilized for planning a desired trajectory.

Three 3D printed cylindrical objects were manipulated and the system was evaluated. Radii corresponding to these objects were 18mm, 22mm, and 27mm, respectively. The gripper was attached to a physical support structure so that the manipulation workspace was parallel to the object's support plane. System state data was then evaluated online, observing object location, goal location, and corresponding nodes of interest in the graph. The system executed randomly selected goal locations, corresponding to random nodes in the graph. Since not all points are reachable, a validity check first determined if a path existed. If a path did not exist, a new random location in the workspace was selected. Otherwise, we executed the desired path determined by the graph.

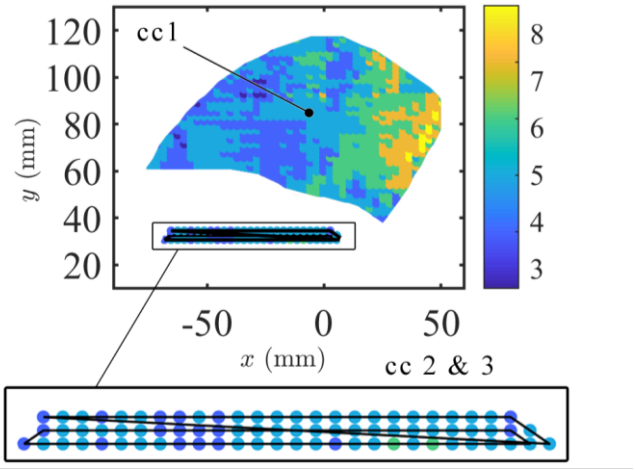

Fig. 4. The three largest strongly connected components (cc) of the graph created for the 18mm object, shaded according to the number of edges connected to each node.

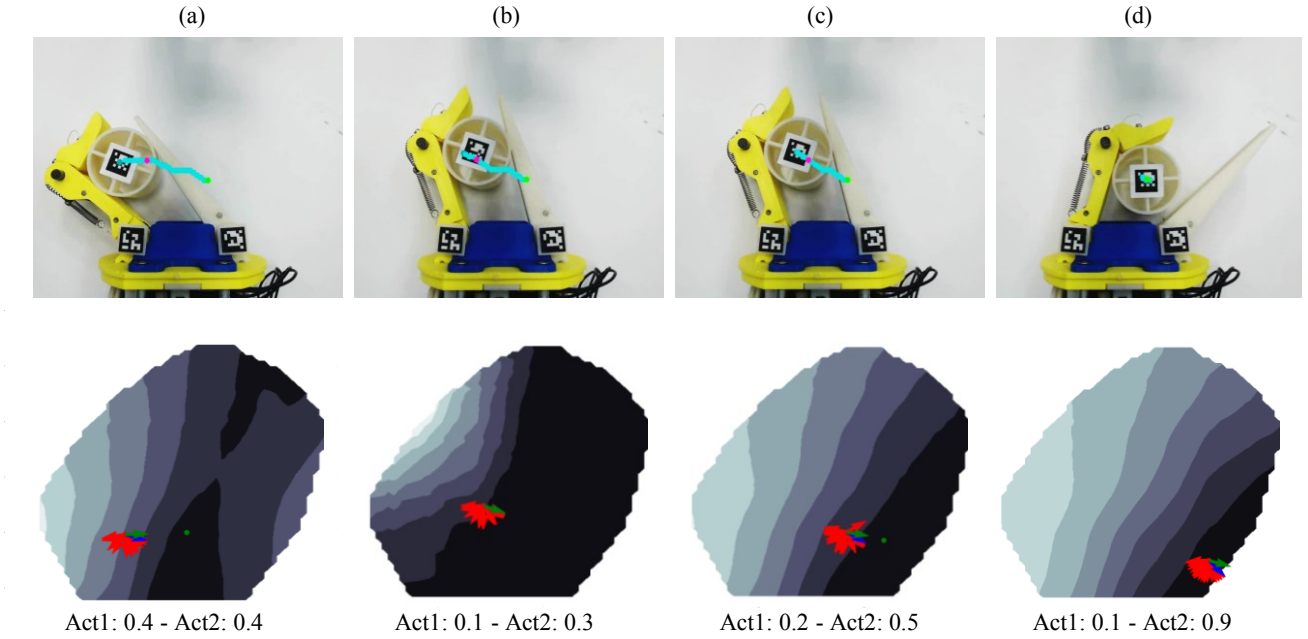

Fig. 5. Example trajectory moving the object from the left to right side of the workspace. (Top) The teal trajectory indicates the shortest path found to the goal position (green), along waypoints (pink). Throughout our progression from (a) – (d), the system determines that we have deviated off the desired path in (b), so an updated path is formulated in (c), and executed in (d) until our goal position is reached. (Bottom) Energy gradient maps are evaluated at each time step, selecting the gradient that is instantaneously closest to our desired object direction. All paths are outlined in red, whereas the green arrow indicates our desired direction and the blue arrow indicates our selected activation input.

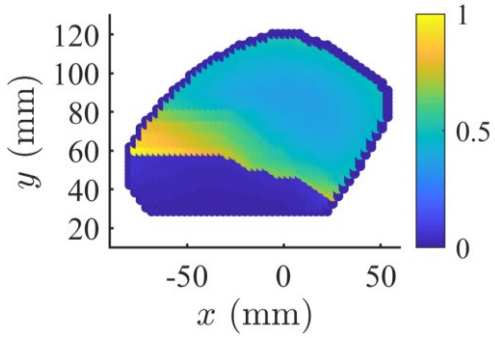

Fig. 6. The normalized summed total edge distance of all shortest paths possible from a given start node for the smallest object (18mm).

Fig. 7. Energy gradients help define desired non-linear trajectories when point-to-point trajectories from current location to goal location are unavailable.

IV. Graph Evaluation and Experiments

In this section, we evaluate properties of our constructed graph to understand connectivity between different areas of the workspace. We follow this discussion with experiments conducted on a physical robotic hand, and evaluate successful trajectories found in our implementation.

A. Graph Evaluation

To better understand the manipulation capability of specific hand-object systems, we used standard graph theory measures to evaluate the graphs constructed from the set of energy maps as described in Section II. First, we considered the connectivity of the graph, to learn where the object might be more freely manipulated within the hand. Specifically, we found the strongly connected components of our graph using a breadth-first search. The three largest resulting strongly connected components are shown in Fig. 4 for the smallest object (18mm). Each grid point in this figure represents a node in the graph, and the shading of that point indicates the number of edges connected to that node. It is interesting to note that there is an extremely large connected component further out in the workspace, where most of the manipulation occurs. The next largest connected components are orders of magnitude smaller in size, and concentrated at the very bottom of the reachable workspace. Intuitively, there is no connection between these two regions. In practice, this is demonstrated very clearly by the fact that the hand cannot push the object outwards from the palm, making it a likely region for the object to become stuck. Since this is the case, path planning often fails when the object is in that region.

We also created shortest path trees between every node in the graph and all other nodes. Then, we summed the total edge distance of all shortest paths possible from a given start node.

The results of this are shown as a color coding in Fig. 6, again for the smallest object (18mm). Essentially, the lighter colored areas in this figure are regions in the workspace that have high total edge distance sums. This means they are most connected to regions very far away, indicating a higher likelihood of finding a valid path between those nodes and randomly generated goal nodes far away. These regions occur along the lower boundary of the shaded region, along the path followed by the object when it is contacting both the proximal and distal links of the 2-link finger, as it sweeps across the workspace.

B. Experiments

In our physical experiments, as presented in Fig. 5, we are able to evaluate trajectories developed by the graph and in real-time, execute computed paths. Given a valid desired trajectory, we first find the path required to reach our desired goal. Once a path is validated, we query our graph for all energy gradients for our current node, and select an actuation input that is closest to our desired next goal direction. Some goal locations cannot be reached due to friction, limited control authority, or kinematic limitations of the hand, and in these cases, a different goal position was generated. During object manipulation, it is required that we update our planned trajectories online, which is due to uncertainties in object movement and the difficultly of precisely following the original desired path. This adaptive planning approach was shown to be successful in our physical implementation, illustrated in Fig. 5 and further in the media attachment.

The main benefit of our energy gradient approach is that it allows us to define shortest path trajectories that are non-trivial to compute otherwise. That is, as presented in Fig. 7, our computed trajectory for these two cases is not a simple point-to-point path within the workspace, but a more complex state-to-state transitions. Analyzing these paths in accordance to the geometric constraints of the gripper, we can note that sharp changes in the projected path are typically due to the fluidly changing contact scenarios throughout the manipulation.

V. Conclusion

In this work, we presented an approach to address the problem of within-hand manipulation for caging grasps. Rather than explicitly modeling the dynamics between the object and finger contacts, we adopted the concept of energy maps to represent the underlying relationship between hand-object configurations and actuation inputs when a caging grasp is formed. By pre-computing a large set of energy maps corresponding to different actuation inputs, an energy gradient-based graph was constructed to represent the connectivity among all hand-object configurations, exploiting the local transitions enabled by the energy gradient. Using the constructed graph, we showed how to plan a sequence of actuation inputs to manipulate the object to achieve a goal state, as well as how to execute the plan adaptively to handle the uncertainties during manipulation.

In experiments, we quantitatively analyzed the properties of our proposed energy gradient-based graph for a single hand-object system, and showed that it can cover a large portion of the gripper's workspace, allowing the gripper to manipulate an object between many different positions within hand. Moreover, we showed that our approach is able to generate shortest actuation sequence trajectories for manipulation, as well as to adaptively update the execution trajectory online to improve the execution robustness. In future work, we plan to develop an object-independent energy gradient-based map, in order to generalize the system to work with novel asymmetric objects and to change their orientation in a controlled way.

References

[1] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: Ingrasp manipulation using purely kinematic trajectory optimization," in Robotics: Science and Systems, 2017.

[2] R. R. Ma, W. G. Bircher, and A. M. Dollar, "Toward robust, wholehand caging manipulation with underactuated hands," in IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 1336–1342.

[3] Mason, Matthew T., and J. Kenneth Salisbury Jr. "Robot hands and the mechanics of manipulation." (1985).

[4] M. Li, K. Hang, D. Kragic, and A. Billard, "Dexterous grasping under shape uncertainty," Robotics and Autonomous Systems, vol. 75, pp. 352 – 364, 2016.

[5] J. Trinkle and R. Paul, "Planning for dexterous manipulation with sliding contacts," The International Journal of Robotics Research, vol. 9, no. 3, pp. 24–48, 1990.

[6] K. Tahara, S. Arimoto, and M. Yoshida, "Dynamic object manipulation using a virtual frame by a triple soft-fingered robotic hand," in IEEE International Conference on Robotics and Automation (ICRA), 2010.

[7] Y. Bekiroglu, J. Laaksonen, J. A. Jorgensen, V. Kyrki, and D. Kragic, "Assessing grasp stability based on learning and haptic data," IEEE Transactions on Robotics, vol. 27, no. 3, pp. 616–629, 2011.

[8] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," IEEE Transactions on Robotics, vol. 32, no. 4, pp. 960–972, 2016.

[9] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge, "Extrinsic dexterity: In-hand manipulation with external forces," in IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 1578–1585.

[10] N. C. Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," in Proceedings of Robotics: Science and Systems, June 2018.

[11] S. Cruciani, C. Smith, D. Kragic, and K. Hang, "Dexterous manipulation graphs," in IEEE International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[12] J. Mahler, F. T. Pokorny, Z. McCarthy, A. F. van der Stappen, and K. Goldberg, "Energy-bounded caging: Formal definition and 2-d energy lower bound algorithm based on weighted alpha shapes," IEEE Robotics and Automation Letters, vol. 1, no. 1, pp. 508–515, 2016.

[13] A. Varava, D. Kragic, and F. T. Pokorny, "Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1479–1497, 2016.

[14] J. A. Stork, F. Pokorny, and D. Kragic, "A topology-based object representation for clasping, latching and hooking," in IEEE International Conference on Humanoid Robots (HUMANOIDS), pp. 138-145, 2013.

[15] Makita, Satoshi, and Yusuke Maeda. "3D multifingered caging: Basic formulation and planning." Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008.

[16] Bircher, Walter G., Dollar, Aarom M., and Rojas Nicolas. "A two-fingered robot gripper with large object reorientation range." Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.

[17] Maeda, Y. and Asamura, T., 2016, July. Sensorless in-hand caging manipulation. In International Conference on Intelligent Autonomous Systems (pp. 255-267). Springer, Cham.