Path Planning for Within-Hand Manipulation over Learned Representations of Safe States

Berk Calli, Andrew Kimmel, Kaiyu Hang, Kostas Bekris, and Aaron Dollar

Abstract This work proposes a framework for tracking a desired path of an object held by an adaptive hand via within-hand manipulation. Such underactuated hands are able to passively achieve stable contacts with objects. Combined with vision-based control and data-driven state estimation process, they can solve tasks without accurate hand-object models or multimodal sensory feedback. In particular, a data-driven regression process is used here to estimate the probability of dropping the object for given manipulation states. Then, an optimization-based planner aims to track the desired path while avoiding states that are above a threshold probability of dropping the object. The optimized cost function, based on the principle of Dynamic-Time Warping (DTW), seeks to minimize the area between the desired and the followed path. By adapting the threshold for the probability of dropping the object, the framework can handle objects of different weights without retraining. Experiments involving writing letters with a marker, as well as tracing randomized paths, were conducted on the Yale Model T-42 hand. Results indicate that the framework successfully avoids undesirable states, while minimizing the proposed cost function, thereby producing object paths for within-hand manipulation that closely match the target ones.

1 Introduction

Within-hand dexterity enables robots to achieve efficient, human-like manipulation skills by reducing unnecessary large whole-arm motions and other compensatory actions, such as re-grasping. This often requires, however, accurate models of the hand-object system and multi-modal sensory feedback for planning and control [3, 15, 8]. Underactuated hands, such as the one shown in Fig. 1, passively achieve stable contact thanks to their adaptability.

Berk Calli is with the Computer Science Department and Robotic Engineering Program of Worcester Polytechnic Institute, MA, USA. e-mail: bcalli@wpi.edu;
Andrew Kimmel and Kostas Bekris are with the Computer Science Department of Rutgers University, NJ, USA. e-mail: {andrew.kimmel,kostas.bekris}@cs.rutgers.edu;
Kaiyu Hang and Aaron Dollar are with Mechanical Engineering and Materials Science Department of Yale University, CT, USA. e-mail: {kaiyu.hang,aaron.dollar}@yale.edu.

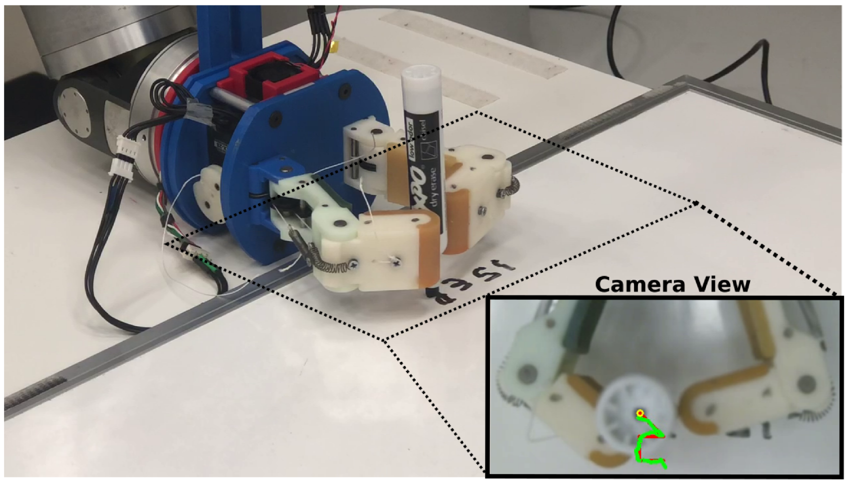

Fig. 1: The Yale T-42 hand writing "ISER".

They allow pure vision feedback-based manipulation even without accurate hand-object models [4]. Using a data-driven approach, vision feedback can further enable the prediction of manipulation states, such as rolling, sliding, singularities and object dropping [6].

In tasks, such as handwriting or painting, the object must be moved along a desired path. Directly tracing such a path without considering the undesired states exposes the execution to risks, such as dropping the object. This work focuses on path planning for within-hand manipulation to trace a reference path, while avoiding to drop the object. Accordingly, given a reference path π^+, the objective is to plan a path so as to find path π^*:

$$\pi^* = \arg\min_{\pi} \operatorname{diff}(\pi, \pi^+), \text{ subject to} : \omega(q, \dot{q}) \geq K, \ \forall\, q \in \pi \text{ and } \dot{q} \in \dot{\pi}, \quad (1)$$

where $\operatorname{diff}(\pi, \pi^+)$ is the path difference, $\omega \in [0, 1]$ is the risk estimate given a configuration along the path and its derivative, and K is a risk threshold.

For measuring path difference, this work uses the Dynamic Time Warping (DTW) distance [14, 10, 16], which minimizes the area between two curves given their time parameterization. DTW is preferred over *Hausdorff* distance [2], which does not reason about the order of states along a path. It is also preferred over the Fréchet distance [1, 9, 17, 18], which can lead to non-robust behavior, where small input variations can significantly distort the output [7].

2 Technical Approach

Planning the joint motions of a robotic hand so as to move a grasped object to a target pose often requires accurate hand-object models, which are difficult to obtain. Instead, the proposed method: a) utilizes vision feedback for generating velocity references in the task space, b) projects the velocities to the joint space via rough approximations, and c) relies upon passive adaptability of the underactuated hands for maintaining stable object contact [4].

Accordingly, the planner reasons only about the configuration of the held object. A regression and classification process is used which evaluates the validity of moving the held object in a particular direction from a given configuration. The objective of the planner is to find a path that minimizes the distance to the reference path, while also ensuring that the paths remains valid (safe) as shown in Fig. 2.

Classifier: The classifier is based on previous work [6], which uses visual and actuator measurements to detect four manipulation states, i.e. object rolling (normal operation), sliding, near drop, and stuck in a singularity. This work adopts the same framework, and trains a classifier for detecting states that would cause dropping the object. Nevertheless, the approach does not use actuator measurements as features, since these data are not available for offline planning. Instead, the classifier is trained with the object position and its velocity along the trajectory. Therefore, the classifier output ω in Eq. (1) corresponds to the probability of the state being safe given the object's position and reference velocity. The training used state vector machines (SVM) with a radial basis function.

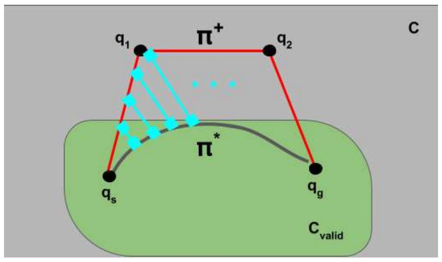

Fig. 2: A reference path π^+ and a path π^* tracking it, where π^+ consists of piecewise straight line segments connecting set points. The valid region $\mathbb{C}_{valid}$ is defined by the classifier. Path π^* must remain within $\mathbb{C}_{valid}$ and minimize the area between it and π^+. The initial and final points q_s, q_g are within $\mathbb{C}_{valid}$.

An object configuration $q \in \mathbb{C}$ is valid under the following constraint:

$$\texttt{valid}(q) = \begin{cases} 1, \, \omega(q, \dot{q}) \geq K \\ 0, \, otherwise \end{cases} \tag{2}$$

where ω is the classifier output, and K is the risk-parameter. The valid configuration space is therefore based on the classifier, $\mathbb{C}_{valid} = \{\forall q \in \mathbb{C} : \texttt{valid}(q) = 1\}$. The risk-parameter K allows for the same classifier to be used in challenging scenarios without requiring it to be retrained. For example, heavier objects can increase the risk that the object gets dropped during within-hand manipulation. In such cases, K should be increased, making the system more conservative and discouraging the object to get close to risky states. Conversely, lowering K would allow exploration of object states closer to the boundaries of $\mathbb{C}_{valid}$, which might be desirable for higher quality paths for lighter objects.

Curve Similarity through Dynamic Time Warping: Let $q \in \mathbb{C} \subset \mathbb{R}^2$ represent the 2-dimensional configuration of the held object. Given two paths $\pi_1 = \{q_1(t), t \in \{1, T_1\}\}$ and $\pi_2 = \{q_2(t), t \in \{1, T_2\}]\}$, where $q_i(t)$ maps to configurations along the corresponding path, and t represents a discrete time parameterization of the path. Assume that a correspondence map exists $\phi = [\phi_1(t), \phi_2(t)]$ between π_1 and π_2, such that a configuration $q_1(\phi_1(t)) \in \pi_1$ maps to a configuration $q_2(\phi_2(t)) \in \pi_2$ for $t \in \{1, ..., T\}$, where $T = max(T_1, T_2)$. With a slight abuse of notation, let $\mathbf{v}(\phi_t)$ be a vector between corresponding mapped points at time t along π_1, π_2. The similarity measure [14] between these paths is the following:

$$\text{diff}(\pi_1, \pi_2) = \sum_{i=2}^{T} ||\mathbf{v}(\phi_t) - \mathbf{v}(\phi_{t-1})||_2 \quad (3)$$

Traditionally, DTW involves computing the correspondence map ϕ such that Eq. 3 is minimized. Because dynamic programming principles hold for DTW, this lends itself nicely to an incremental search. This also assumes, however, that there are two complete paths available for comparison. In this application, one of these paths is discovered on the fly. To accommodate this, the proposed search process imposes a series of constraints while solving the dual of DTW so as to obtain a path that optimizes Eq. 3.

Optimizing DTW through Informed Search: Similar to the above discussion, $\mathbb{C}_{valid}$ denotes the valid configuration space of the object, i.e., the manipulator is not in an undesirable state (singularity, dropping or sliding object). This subspace is defined through the classifier. Given a sequence of target object configurations $Q^+ = \{q_1^+, q_2^+, ..., q_T^+\} \in \mathbb{C}$, the *reference path* π^+ is composed of piecewise linear segments connecting subsequent pairs of points in Q^+. The objective then is to compute a continuous curve $\pi : [0, 1] \rightarrow \mathbb{C}_{valid}$, which minimizes Eq. 3 relative to the reference path π^+, i.e. diff(π, π^+).

In order to use Eq. 3, a time parameterization over π, π^+ must be employed. This is defined by discretizing the reference path using a fixed-length δ, essentially enforcing a constant velocity over the path. The correspondence map can then be defined as $\phi = [\phi(t), \phi_+(t)]$ for N timesteps, where N stands for the number of δ segments in π^+. Furthermore, the optimization is constrained, such that $\pi(0) = q_1^+, \pi(1) = q_T^+ \in \mathbb{C}_{valid}$, and only *monotonic* paths are explored. The *monotonicity* requirement enforces that corresponding configurations can only be reached by prior time steps, which prevents sudden "jumps" along the reference path.

Computing π can then be accomplished by utilizing an A* search rooted at q_1^+ and terminating at q_T^+. Each search node keeps track of its correspondence map index t, which determines which point on π^+ the node is mapped to. The search proceeds by expanding valid configurations by a δ amount at each iteration, constrained in the 8 cardinal directions for computational efficiency. The search nodes are visited in order according to the evaluation function $f(q(t)) = g(q(t), q^+(t)) + h(q(t))$, where $q(t) \in \pi$ and $q^+(t) \in \pi^+$. The cost of a node g is then computed from Eq. 3 given the duration of the path that corresponds to the search node. Let $m = \frac{||q(t)-q^+(t)||_2}{\delta}$, then:

$$h(q(t)) = \frac{1}{2} \sum_{k=0}^{m} (||q(t) - q^+(t)||_2 - \delta * k). \quad (4)$$

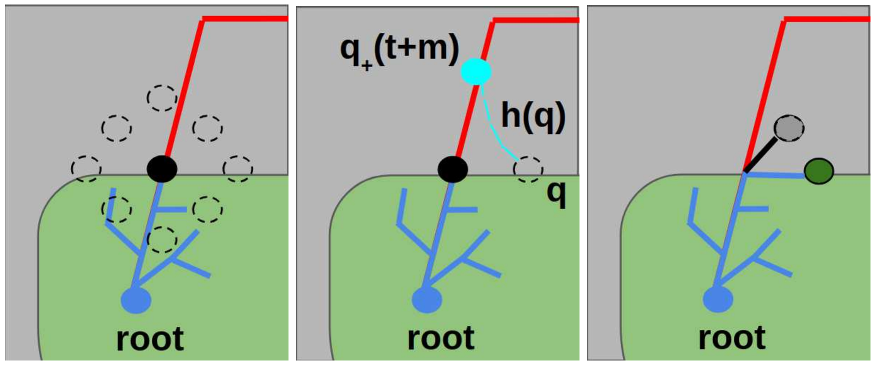

Fig. 3: (Left) iSST selects the black vertex to expand from; 8 possible directions are considered; (Middle) A heuristic is computed for each possible direction using Eq. 4, (Right) Directions with the lowest heuristic whose edges lie wholly in the valid configuration space are added to the tree.

The heuristic represents the minimum required area to the reference path for a path starting from $q(t)$ to latch on to the reference path $\pi^{+}(t)$. The best case involves the paths π^{+} and π approaching one another at a rate of $2 * \delta$ per time step. This is therefore an admissible heuristic for the search process.

Efficient DTW Optimization: Although the A* process described above manages to produce appropriate solutions, as shown in Section 3, there are some drawbacks which warrant the use of an alternative process. Primarily, as the resolution of the search increases (i.e. by decreasing the value of δ), the A* process increasingly depends on the validation of subsequent search nodes to be fast. Despite the classifier calls being relatively quick, they incur a measurable cost, and accordingly the number of calls to the classifier could be viewed as an objective to minimize. Furthermore, A* can only provide solutions up to the selected resolution. To alleviate these issues, an alternative search process is proposed, based on an informed asymptotically optimal sampling-based planner (iSST [12, 13]) that does not require access to a steering function [11].

The heuristic described in Eq. 4 can be directly applied in this method, allowing it to reduce the number of validity checks, as well as guiding the search process. Along with randomly sampling in the configuration space, the method also makes use of maneuvers (i.e. guided samples), which in this case will be similar to the A* (8 cardinal directions). An illustration of this is shown in Figure 3. The initial solution returned by the method, while not guaranteed to be optimal as in A*, can be improved upon, due to the asymptotically-optimal property of iSST. This allows an anytime performance, which is a desirable property for execution of trajectories on real robots.

3 Simulation Results

The search framework was implemented using both an A* approach, as well as iSST [12]), a heuristically-guided, asymptotically-optimal tree planner that propagates a single control during each iteration. A visual comparison of the expansion process for both is shown in Fig. 4. For each simulated evaluation, a reference path is generated by uniformly sampling the set points with the following constraints: (1) the start and goal are valid, (2) there are at least 4 total set points, and (3) at least 1 of the set points is invalid. These constraints ensure that directly tracing the reference path results in a failure state.

The two alternatives are compared in terms of average success rate, computation time, and solution cost. Both methods are evaluated with two classifiers: i) *simulated*, which partitions $\mathbb{C}_{valid}$ using a random distribution of obstacles and uses standard collision-checking to determine state validity; and ii) *trained*, which invokes the trained classifier to determine state validity. Fifty trials with different randomized reference paths were tested. A trial was counted as successful if it found a solution within 60 and 300 seconds for *simulated and trained* classifiers respectively. The solution cost was computed using Eq. 3.

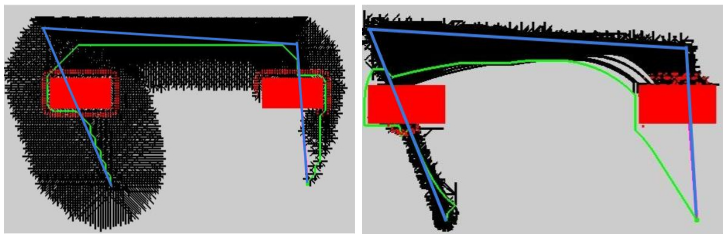

Fig. 4: A visual comparison of A* (left) and iSST (right) search trees with simulated obstacles. The blue line represents π^+, the green line is the solution π, the black lines are the search tree, and the red blocks are invalid regions.

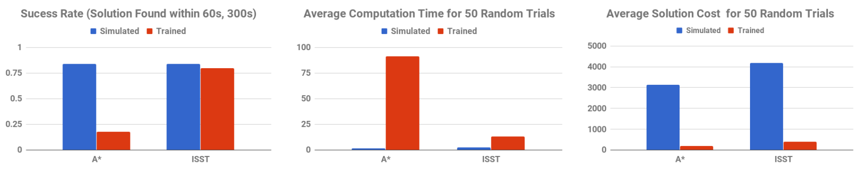

Fig. 5: Results of A* vs. iSST with *simulated and trained* classifiers over 50 randomized reference trajectories . (left) Success Rate, (middle) Average Computation Time, (right) Average Cost.

All experiments were run on a single computer with an Intel Xeon E5-1660 CPU and 32 GB RAM, with results shown in Figure 5. In the *simulated* classifier case, the A* performed the best across all metrics, although iSST remained competitive. In the *trained* classifier case, A* only produced solutions 18% of the time (within the 300s time limit), compared to iSST's 78%, which requires fewer propagations that are computationally expensive as they require multiple calls to the trained classifier.

4 Experiments

Given the randomized set points described in Section 3, several random reference paths were generated and tested given a) a naive planner that follows the reference path with a straight line, and b) the iSST planner that considers the classifier's output for avoiding risky regions. The Model T-42 hand (Fig. 6) is utilized together with a webcam observing it from the top.

Results with Randomized Trajectories: For testing the ability of the algorithm to avoid risky regions, we generated 10 random reference paths, and conducted experiments with the naive planner and the safe iSST planner using a cylindrical object with 2 cm diameter. For all 10 paths, the naive trajectory resulted in dropping the object, whereas the iSST planner successfully reaches the goal by avoiding the risky regions. Three samples from our runs are presented in Fig. 6. It can be seen that the planner diverts from the naive trajectory to stay in the safe regions according to the classifier output, but the distance to the straight path is minimized as much as possible. Despite these deviations, the executed path preserves a notion of similarity to the original reference trajectory.

Results with Different Weights: In this set of experiments, the same cylindrical object is used, but additional weight is attached to make it six times heavier. Here, we analyze the effectiveness of the risk factor parameter K for handling this scenario, which is significantly different than the data used for training the classifier. We run the experiments with risk factors varying from 0 to 1 with 0.1 increments. The trajectories with $K = 0$, $K = 0.6$ and $K = 0.8$ are presented in Fig. 7. It is seen that the system is able to execute longer segments of the trajectory without failure as the K value is increased. The object is dropped for risk factor values between $K = 0$ and $K = 0.7$, and the whole trajectory is successfully executed for values between 0.8 and 1.0. This indicates that the risk parameter is instrumental

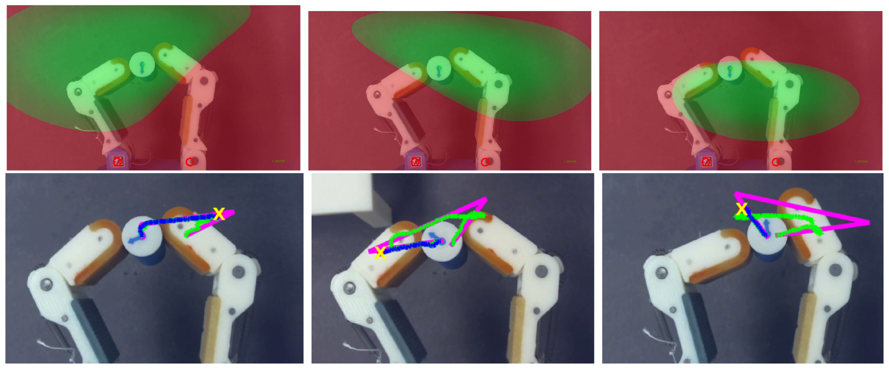

Fig. 6: (Top) The classifier outputs for the exploration direction that caused dropping the object (*green*: normal, *red*: drop), and (Bottom) the generated random paths (pink lines), executed trajectory with the naive planning (blue line), and executed trajectory with I-SST (green line). The object drops for the naive execution (the drop locations are marked with yellow crosses), while iSST completes the trajectory successfully.

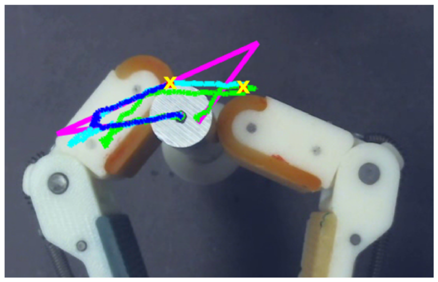

Fig. 7: Manipulation of a heavier object with varying risk parameter (K). Reference trajectory (pink), executed trajectory with $K = 0$ (blue), executed trajectory with $K = 0.6$, and executed trajectory with $K = 0.8$. The object drops for the $K = 0$ and $K = 0.6$ at the locations indicated by yellow crosses.

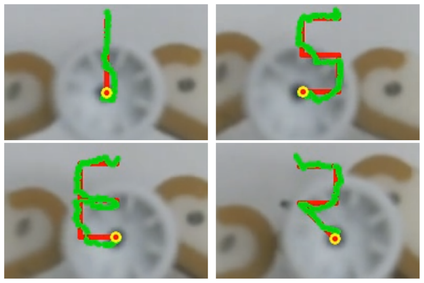

Fig. 8: Camera view while writing the letters "I", "S", "E", "R". Red lines are the generated reference trajectories, green lines are the execution of the trajectories.

to cope with challenging scenarios by generating more conservative, cautious trajectories.

Results with Different Object Geometries: In order to further analyze the robustness of the algorithm, we conducted experiments with two cylinders and two rectangles with 2cm and 4cm radius and widths respectively. Four random paths are generated as explained in Section 3, and each path is tested with all the objects using the naive approach and the safe iSST planner (16 runs each). The risk parameter K was set to zero. The naive approach was successful only for 1 out of 16 cases, whereas the planner was successful for 15/16. This indicates the effectiveness of the safe planning framework. Nevertheless, for three of the successful iSST executions, even though the object is not dropped, significant out-of-plane rotations were observed for the rectangular objects. This was largely caused by within-hand sliding. In these cases, since $K = 0$, the system operates very close to the vicinity of the risky region, and cannot compensate for the unplanned sliding motions. Since the classifier utilized in this paper does not take sliding into account, the system gets too close to the risky region border. The failed iSST case was with the large rectangle, and was also caused by such sliding motions. This can potentially be fixed by increasing the K value, and obtaining more conservative trajectories. Utilizing a classifier that can detect sliding is another option, which is discussed in Section 5. In the successful execution of the naive planner, the sliding motion caused a positive effect by keeping the object close to the palm, resulting in path completion.

Handwriting Experiments: The safe planning framework was used to generate trajectories of the "I", "S", "E" and "R" letters: the feasibility of each letter was checked with our classifier, and the letters are shrunk until safe trajectories were possible. We used a white board and a board marker grasped by the Model T42 hand. A snapshot of the experiment is given in Fig. 1, and the letter trajectories followed by the marker is presented in Fig. 8. Even though the writing operation was successful, we observed that the letters written by the system deviated from the ideal trajectories. This was partially caused by the contact between the pen and the white board, which can be addressed with a more robust image-based controller, i.e. [5]. Nevertheless, the shapes were still well-preserved, and clearly recognizable (Fig. 9). To the best of our knowledge, this is the first execution of handwriting purely implemented with within-hand motions (except for moving between letters, which was done with the robot arm motion).

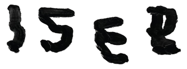
Fig. 9: "ISER" written by the T42 Hand.

5 Conclusions

The proposed method is a crucial part of a vision-based underactuated within-hand manipulation framework. The system does not rely on accurate hand-object models, or force sensors, but it is still able to generate safe and reliable trajectories. The risk factor parameter is an effective tool to make the system cope with challenging cases by generating more conservative trajectories. Nevertheless, we have also seen that unplanned sliding motions may occur during manipulation, which effects system robustness negatively. One solution would be to utilize a classifier that can recognize gripper singularities and sliding along with drop modes as in other related work [6]. Not being able to access actuator loads, velocities and finger positions, however, during offline planning, makes it much more challenging to recognize all the target modes and have a good partitioning of the workspace as in Fig. 6. This issue can be addressed by employing different regression methods and data evaluation.

Another improvement can be achieved by using a probabilistic classifier in a planning under uncertainty framework, which reasons over distributions of safe states. This can potentially provide improved tracking results by allowing the robot to momentarily enter "dangerous" regions for short periods of time. This would entail changing the objective function to operate over expected costs.

Acknowledgements: This work was partially sponsored by NSF IIS-1734492, IIS-1723869, and IIS-1317976.

References

1. Alt, H., Godau, M.: Computing the fréchet distance between two polygonal curves. International Journal of Computational Geometry & Applications **5**(01n02), 75–91 (1995)
2. Belogay, E., Cabrelli, C., Molter, U., Shonkwiler, R.: Calculating the hausdorff distance between curves. Information Processing Letters **64** (1997)
3. Bicchi, A., Melchiorri, C., Balluchi, D.: On the mobility and manipulability of general multiple limb robots. IEEE Transactions on Robotics and Automation **11**(2), 215–228 (1995). DOI 10.1109/70.370503
4. Calli, B., Dollar, A.M.: Vision-based precision manipulation with underactuated hands: Simple and effective solutions for dexterity. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1012–1018 (2016). DOI 10.1109/IROS.2016.7759173
5. Calli, B., Dollar, A.M.: Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection. IEEE Transactions on Automation Science and Engineering pp. 1–14 (2018). DOI 10.1109/TASE.2018.2819661
6. Calli, B., Srinivasan, K., Morgan, A., Dollar, A.M.: Learning modes of within-hand manipulation. In: IEEE International Conference of Robotics and Automation (Accepted) (2018)
7. Efrat, A., Fan, Q., Venkatasubramanian, J.: Curve matching, time warping, and light fields, new algorithms for computing similarity between curves. Mathematic Imaging and Vision **27**, 203–216 (2007)
8. Hang, K., Li, M., Stork, J.A., Bekiroglu, Y., Pokorny, F.T., Billard, A., Kragic, D.: Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation. IEEE Transactions on Robotics **32**(4), 960–972 (2016). DOI 10.1109/TRO.2016.2588879
9. Har-Peled, S., Raichel, B.: The fréchet distance revisited and extended. ACM Transactions on Algorithms (TALG) **10**(1), 3 (2014)
10. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowledge and information systems **7**(3), 358–386 (2005)
11. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. International Journal of Robotics Research (IJRR) **35**(5), 528–564 (2016)
12. Littlefield, Z., Bekris, K.E.: Informed asymptotically near-optimal planning for field robots with dynamics. In: Field and Service Robotics, pp. 449–463. Springer (2017)
13. Littlefield, Z., Bekris, K.E.: Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018)
14. Munich, M.E., Perona, P.: Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In: IEEE Intern. Conf. on Computer Vision, vol. 1, pp. 108–115 (1999)
15. Okamura, A.M., Smaby, N., Cutkosky, M.R.: An overview of dexterous manipulation. In: IEEE Intern. Conf. on Robotics and Automation (ICRA), vol. 1, pp. 255–262 vol.1 (2000). DOI 10.1109/ROBOT.2000.844067
16. Salvador, S., Chan, P.: Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis **11**(5), 561–580 (2007)
17. Solovey, K., Halperin, D.: Efficient sampling-based bottleneck pathfinding over cost maps. In: Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, pp. 2003–2009. IEEE (2017)
18. Wenk, C., et al.: Geodesic fréchet distance inside a simple polygon. ACM Transactions on Algorithms (TALG) **7**(1), 9 (2010)