Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research

Ryan Hoque*,1, Kaushik Shivakumar*,1, Shrey Aeron[1], Gabriel Deza[1], Aditya Ganapathi[1], Adrian Wong[2], Johnny Lee[2], Andy Zeng[2], Vincent Vanhoucke[2], Ken Goldberg[1]

***Abstract*— Autonomous fabric manipulation is a longstanding challenge in robotics, but evaluating progress is difficult due to the cost and diversity of robot hardware. Using Reach, a cloud robotics platform that enables low-latency remote execution of control policies on physical robots, we present the first systematic benchmarking of fabric manipulation algorithms on physical hardware. We develop 4 novel learning-based algorithms that model expert actions, keypoints, reward functions, and dynamic motions, and we compare these against 4 learning-free and inverse dynamics algorithms on the task of folding a crumpled T-shirt with a single robot arm. The entire lifecycle of data collection, model training, and policy evaluation was performed remotely without physical access to the robot workcell. Results suggest a new algorithm combining imitation learning with analytic methods achieves human-level performance on the flattening task and 93% of human-level performance on the folding task. See `https://sites.google.com/berkeley.edu/cloudfolding` for all data, code, models, and supplemental material.**

I. Introduction

Reproducibility is the cornerstone of scientific progress. It allows researchers to verify results, assess the state of the art, and build on prior work. Recent advances in computer vision (CV), for instance, were facilitated by the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [1], a standard benchmark in CV literature.

In robotics, there is no equivalent benchmark. Simulation benchmarks [2], [3], [4] are useful but cannot replace physical experiments as the "reality gap" remains prohibitively large [5]. Physical robots are expensive and vary greatly in their capabilities and morphologies. Each research lab has a unique hardware setup, making it difficult to reliably compare results. Cost also poses a significant obstacle to individuals or institutions who wish to perform robotics research, but lack the resources to do so.

One option is to provide shared access to a remote hardware testbed via the Cloud. In this paper, we describe algorithms and experiments performed entirely remotely using *Reach*, a prototype hardware testbed from Robotics at Google [6]. Reach includes several physical robot workcells and open source software for remote execution of control policies in real time. Each workcell is configured for a benchmark task: one such task is folding a T-shirt with a UR5 robot arm and 3-jaw piSOFTGRIP gripper [7] (Figure 1).

* Equal contribution
[1] AUTOLAB at University of California, Berkeley
[2] Robotics at Google
Correspondence to `ryanhoque@berkeley.edu`

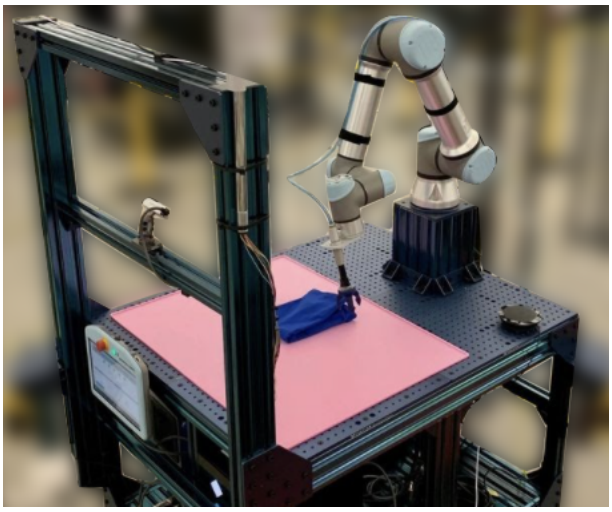

Fig. 1. Reach cloud robotics workcell developed by Robotics at Google.

While folding T-shirts and other garments is a ubiquitous daily task for humans, manipulating fabric remains challenging for robots. Fabric is difficult to model due to its infinite-dimensional state space, complex dynamics, and high degree of self-occlusion. Furthermore, accurately simulating gripper contact mechanics and fabric self-collision remains elusive for existing fabric simulators due to challenges in modeling deformation, friction, and electrostatic forces [4], [8], highlighting the need for physical benchmarking.

This paper makes the following contributions: (1) four novel learning-based algorithms for the unimanual folding task, (2) the first physical benchmarking of fabric manipulation algorithms, and (3) a case study of robotics research performed exclusively using a remotely managed robot workcell. This paper does *not* contribute the design of the Reach platform, which is being developed by a larger team at Google [6].

II. Related Work

Remote Testbeds: Remote robotics testbeds include Robotarium [9] for swarm robotics and Duckietown [10] for autonomous driving. The most similar remote testbed is from Bauer et al. [11], who hosted the online "Real Robot Challenge" for manipulation in 2020 and 2021 at Neural Information Processing Systems (NeurIPS). Six robotics groups from around the world were able to access their tri-finger robot [12] remotely via the Internet and evaluate their algorithms on the shared infrastructure. Our study differs from this project in the following ways: (1) they

consider dexterous manipulation of rigid objects while we consider deformable object manipulation; (2) they use a custom tri-finger robotic system while we use a UR5 robot arm, standard in industrial settings; and (3) the Real Robot Challenge submissions are either learning-free [13], [14] or learned only in simulation [15], [16], while we consider learning algorithms trained on real data.

Reproducibility in Robotics: Several other approaches have been proposed for facilitating reproducibility in robotics research. One direction is benchmarking in simulation, where evaluation is inexpensive and reproducible. Simulation environments have been developed for robot locomotion [3], household tasks [2], and deformable object manipulation [4]. While researchers have made significant progress on these benchmarks, especially using reinforcement learning [17], such advances do not readily transfer to physical robots [5]. Another initiative for improving reproducibility is development of a low-cost open source platform that can be assembled independently by different labs [18], [12]. A third approach considers benchmarking performance on large offline datasets such as robot grasps on 3D object models, e.g., EGAD [19] and Dex-Net [20]; RGBD scans and meshes of real-world common household objects, e.g., the YCB Object and Model set [21]; and video frames of robot experience, e.g., RoboNet [22]. These datasets have been used to explore and compare algorithms [23], but they limit evaluation to states within the dataset.

Autonomous Fabric Folding: Autonomous fabric manipulation is an active challenge in robotics. Maitin-Shepard et al. [24] and Doumanoglou et al. [25] present early approaches to reliably fold towels and garments, respectively, from crumpled initial configurations. Weng et al. [26] and Ha et al. [27] develop learning-based algorithms for fabric manipulation using optical flow and dynamic flinging motions respectively. However, these approaches were evaluated on dual-armed robots, which require coordination and are more costly. There has been recent interest in learning algorithms for unimanual (single-arm) fabric manipulation [28], [8]. These achieve strong results on fabric smoothing and folding tasks, but robust and precise unimanual T-shirt folding remains an open problem. Lin et al. [4] propose an environment for fabric manipulation and benchmark several learning algorithms, but limit results to simulation. Garcia-Camacho et al. [29] propose benchmarking tasks for bimanual fabric manipulation but allow robot hardware to vary and do not evaluate learning algorithms.

III. The Google Reach Testbed

In this section, we review the details of the Google Cloud Robotics testbed [6] that are most salient for this case study.

A. Hardware

See Figure 1 for an image of the workcell. The robot is a single Universal Robot UR5e arm equipped with a Piab piSOFTGRIP vacuum-driven soft 3-jaw gripper [7]. The workcell is equipped with 4 Intel Realsense D415 cameras which each capture 640 $\times$ 360 RGB images at 20 FPS and 640 $\times$ 360 depth images at 1 FPS. The worksurface is a bright pink silicone mat and the garment is a blue crew-neck short sleeve T-shirt. The workcell is maintained by lab technicians who are onsite 8 hours a day to reset the robot and troubleshoot system-level errors.

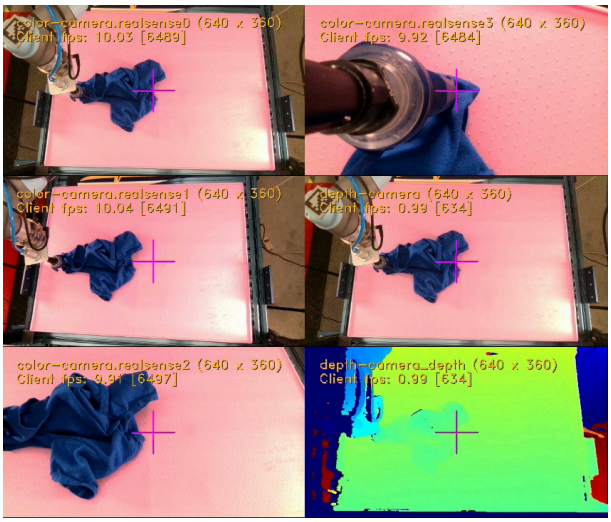

Fig. 2. The client PyReach viewer, which updates the RGB images from the workcell cameras at 10 Hz and depth images at 1 Hz. Our algorithms use the overhead RGB images (top left panel). Note that the lower two panels on the right are from the same camera as the top left panel.

B. Software

Reach includes PyReach, an open source Python library developed by Robotics at Google for interfacing with the Reach system. The software includes infrastructure for authenticated users to establish a network connection with the robot server over the Internet, a viewer tool for locally displaying the 4 workcell camera feeds in real time (Figure 2), a simulated workcell that mimics the real workcell for safely testing motions prior to deployment on the real system, and utility functions such as a pixel-to-world transform using the depth camera and conversions between different pose representations.

PyReach also includes PyReach Gym, an application programming interface (API) modeled after OpenAI Gym [3]. Remote agents receive observations of the environment and request actions through this interface. In particular, at each time step with frequency up to 10 Hz, a remote agent can receive the joint angles and Cartesian pose of the arm, the binary state of the gripper (closed or open), and camera observations. The agent specifies an action to execute as a desired pose of the arm in either joint or Cartesian space and a desired binary state of the gripper.

C. Garment Folding Case Study: Problem Definition

We assume that the target folded configuration is known beforehand, that training and evaluation are performed in real (not simulation), that the hardware setup is as specified in Section III-A, and that the garment stays the same during training and evaluation. The task is to iteratively execute two procedures in a loop: (1) crumple the T-shirt and (2)

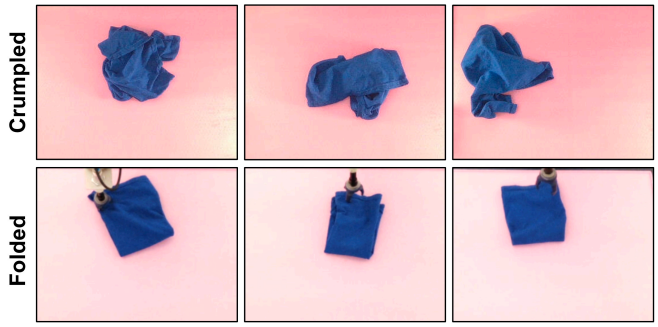

Fig. 3. Examples of crumpled states (Row 1) and folded states (Row 2).

$\mathbf{o}_t$	Cropped RGB observation of the workcell state from the overhead Realsense camera at timestep t (Figure 2).
$\mathbf{a}_t$	The action at time t, expressed as a pick-and-place action (p_0, p_1) in pixel coordinates except in Section IV-A.4.
$\mathbf{o}_t^m$	Color-thresholded mask of the T-shirt analytically computed from $\mathbf{o}_t$.
$\text{com}(\mathbf{o}_t^m)$	A function that returns the *visual* center of the T-shirt.
$\text{c}(\mathbf{o}_t^m)$	A function that computes the 2D fabric coverage.
$\mathcal{T}$	A template image of a fully flattened shirt in the workspace.

TABLE I
NOTATION FOR SECTION IV.

fold the T-shirt. Crumpling is performed via a series of 6 random drops of the T-shirt resulting in an average of 37.5% coverage (Section V-B), where coverage is the fraction of the maximum 2D area the T-shirt is able to attain. The folding task is to manipulate the T-shirt toward the target configuration in Figure 3. We decompose the folding task into two subtasks: (1) *flattening*, i.e., spreading out from an initially crumpled configuration until the garment is smooth, followed by (2) *folding*, i.e., folding the t-shirt from initially flattened until sufficiently close to the target configuration. We measure folding accuracy with a combination of Intersection over Union (IOU) and wrinkle detection (Section V-C).

IV. Garment Folding Algorithms

Due to the unique challenges of the flattening and folding subtasks, we benchmark each subtask with its own set of algorithms. Hyperparameter and implementation details for all algorithms are available in the appendix, and notation for this section is defined in Table I. With the exception of Section IV-A.4, all actions are quasistatic pick-and-place actions from pick point p_0 to place point p_1, where p_0 and p_1 are specified as (x, y) coordinates in pixel space; see Appendix VIII-D for implementation details.

A. Flattening: 4 New Algorithms

1) Learned Pick-Analytic Place (LP_0AP_1): Inspired by prior work in imitation learning for fabric manipulation [8], [30], we develop an algorithm to learn pick points from human demonstrations. Since we empirically observe that human-selected pick points combined with analytic placing performs well on flattening, we propose only learning the pick points p_0 and analytically computing place points with the strategy in Section IV-B.3 to improve sample efficiency. While other work has considered learning a pick-conditioned place policy for fabric manipulation [31], we define analytic placing actions that make the pick-conditioned policy unnecessary. To handle the inherent multimodality in the distribution of human-specified pick points, we train a fully convolutional network (FCN) [32] to output a heatmap corresponding to probability density instead of regressing to an individual action (Figure 4). The FCN can be interpreted as an implicit energy-based model [33], [34] where the state and action pairs are the receptive fields of the network. As in DAgger [35], we reduce distribution shift by iteratively adding on-policy action labels to the dataset.

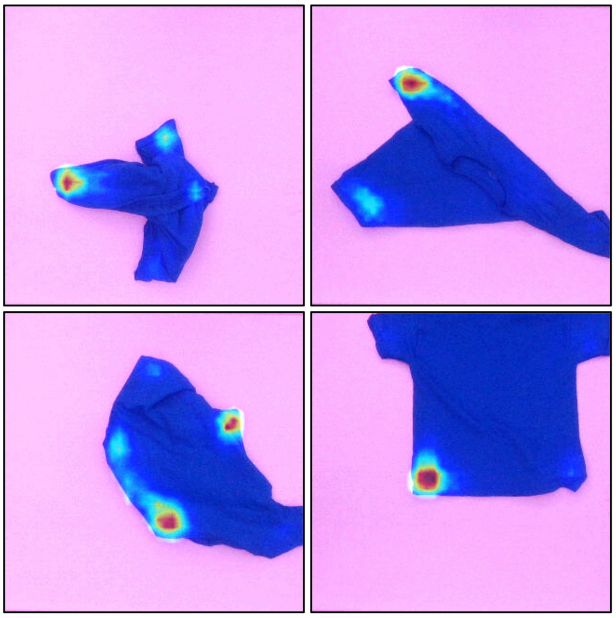

Fig. 4. LP_0AP_1 pick point predictions on the test set. Bright red and yellow regions correspond to high probability pick points. The output heatmap is able to capture the multimodality in human actions.

2) Learning Keypoints (KP): This approach separates perception from planning and proposes to only learn the perception component. Specifically, we collect a hand-labeled dataset of images with up to 5 visible keypoints on the fabric corresponding to the collar, 2 sleeves, and 2 base corners (Figure 5). While the dataset generation policy is open-ended for this approach, we choose to first train an initial KP policy on random data (Section IV-B.1) and then augment the dataset with states encountered under the policy to mitigate distribution mismatch similar to DAgger [35]. We train a FCN with 3 output heatmaps to predict each of the 3 classes of keypoints separately. Using keypoint predictions, we propose an analytic corner-pulling policy inspired by [8] that iteratively moves the keypoints from their current positions to their respective locations on a template flattened shirt $\mathcal{T}$. To reduce ambiguity, we compute the rotation and translation of $\mathcal{T}$ that best matches the current state and first move the keypoint farthest from its target location to its destination. To our knowledge, the combination of the FCN for multi-class keypoint prediction, T-shirt template matching, and corner pulling is a novel flattening policy.

3) Coverage Reward Learning (CRL): This approach seeks to learn a reward function corresponding to fabric

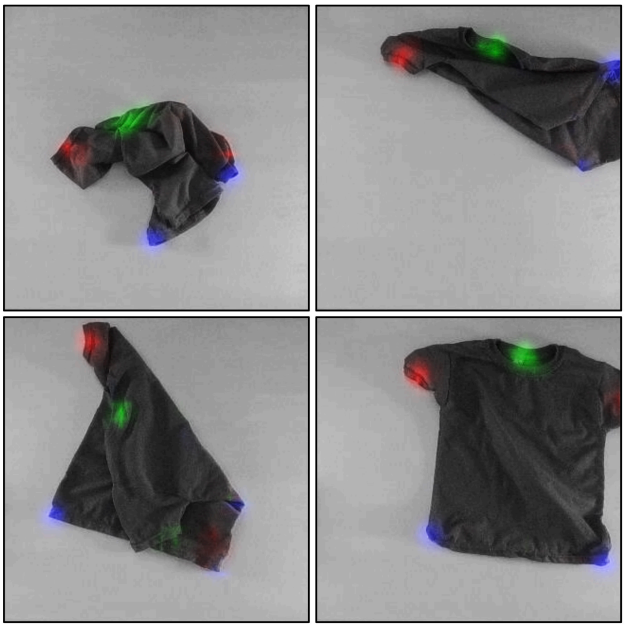

Fig. 5. KP predictions on the test set. The predicted collar is colored green, the two sleeves are red, and the two base points are blue. Shirt images are shown in grayscale for viewing convenience.

coverage $\mathrm{c}(\cdot)$ from data and execute a policy using this reward function. We learn this reward with self-supervised learning and execute a greedy policy that seeks to maximize the 1-step reward at each time step. Specifically, we fit a Convolutional Neural Network (CNN) $R_\theta(\mathbf{o}_t, \mathbf{a}_t)$ to the scalar change in coverage (i.e., $\mathrm{c}(\mathbf{o}^m_{t+1}) - \mathrm{c}(\mathbf{o}^m_t)$) that results from executing action $\mathbf{a}_t$ on $\mathbf{o}_t$. At execution time we randomly sample thousands of pick points on the fabric mask $\mathbf{o}^m_t$ and place points in the workspace and select the action with the highest predicted change in coverage. To our knowledge, greedy planning over a learned model of coverage dynamics for fabric flattening is novel. Once again, dataset generation is a design choice; here, we opt for a random action policy (Section IV-B.1) to enable large-scale self-supervised data collection and increase data diversity.

4) Drop (DROP): Inspired by Ha et al. [27], we investigate whether dynamic motions can leverage aerodynamic effects to accelerate the flattening of the shirt when combined with Approach IV-A.1. We propose a simple vertical drop primitive that grabs the visual center of mass $\mathrm{com}(\mathbf{o}^m_t)$, lifts the shirt into the air, and releases. We profile the coverage dynamics of the drop and the $\mathrm{LP}_0\mathrm{AP}_1$ pick-and-place and run Q-value iteration to determine which primitive to execute (i.e., drop or pick-and-place) given a discretized version of the current coverage $\mathrm{c}(\mathbf{o}^m_t)$. Q-value iteration on the following reward function produces a policy that minimizes the total number of actions required to flatten the shirt:

$$r(s = \mathrm{c}(\cdot), a) = \begin{cases} -1 & \mathrm{c}(\cdot) < C \\ 0 & \mathrm{c}(\cdot) \geq C \end{cases}$$

where C is a coverage threshold defined in Section V-B and $\mathrm{c}(\cdot)$ is the discretized current coverage.

B. Flattening: 4 Baselines

1) Random (RAND): As a simple baseline, we implement a random pick-and-place policy that selects p_0 uniformly at random from $\mathbf{o}^m_t$ and p_1 uniformly at random in the workspace within a maximum distance from p_0.

2) Human Teleoperation (HUMAN): As an upper bound on performance and action efficiency, a human selects pick and place points through a point-and-click interface (see the appendix for details).

3) Analytic Edge-Pull (AEP): We implement a fully analytic policy to explore to what extent learning is required for the T-shirt flattening task. The policy seeks to flatten the shirt by picking the edges and corners and pulling outwards. Formally, we sample p_0 uniformly from the set of points in the shirt mask $\mathbf{o}^m_t$ that are within a distance k from the perimeter of $\mathbf{o}^m_t$, where k is a hyperparameter. Given p_0, we compute p_1 by pulling a fixed distance l in the direction of the average of two unit vectors: (1) away from $\mathrm{com}(\mathbf{o}^m_t)$ and (2) toward the nearest pixel outside $\mathbf{o}^m_t$.

4) Learning an Inverse Dynamics Model (IDYN): A inverse dynamics model $f(\mathbf{o}_t, \mathbf{o}_{t+1})$ produces the action $\mathbf{a}_t$ that causes the input transition from $\mathbf{o}_t$ to $\mathbf{o}_{t+1}$. Here we implement the algorithm proposed by Nair et al. [36], which learns to model visual inverse dynamics. Specifically, we approximate the dynamics with a Siamese CNN $f_\theta(\cdot, \cdot)$ trained on the random action dataset collected in Section IV-A.3. As in [36], the network factors the action by predicting the pick point p_0 before the pick-conditioned place point p_1 to improve sample efficiency. During policy evaluation, the inputs to the network are the current observation $\mathbf{o}_t$ and the template goal observation $\mathcal{T}$.

C. Folding Algorithms

1) Human Teleoperation (HUMAN): As an upper bound on performance, a human chooses pick and place points for folding through a point-and-click interface.

2) Analytic Shape-Matching (ASM): Since the folding subtask is significantly more well-defined than flattening, we investigate whether an open-loop policy computed via shape matching can successfully fold the shirt. We specify a fixed sequence of folding actions with a single human demonstration. During evaluation, we compute rotations and translations of the corresponding template images to find the best match with $\mathbf{o}_t$ and transform the folding actions in the demonstration accordingly.

3) Learned Pick-Learned Place (LP_0LP_1): This approach is identical to Section IV-A.1 but learns both pick points and place points, as the analytically computed place point is designed for flattening. Since folding demonstrations are difficult to obtain (the garment must be flattened first) and successful folding episodes are short-horizon and visually similar, we collect only two demonstrations and augment the data by a factor of 20 with affine transforms that encourage rotational and translational invariance.

4) Fully Autonomous Flattening with Analytic Shape-Matching (A-ASM): The algorithms above are evaluated after the garment is fully flattened via human teleoperation to

study the folding subtask in isolation. This approach, A-ASM, combines the best-performing autonomous flattening algorithm (i.e., LP_0AP_1) with ASM (Section IV-C.2) to evaluate the performance of a fully autonomous pipeline for manipulating the garment from crumpled to folded.

V. Experiments

A. Experimental Setup

All actions executed on the robot are either a pick-and-place primitive (p_0, p_1) or a drop primitive (for the DROP algorithm). See Appendix VIII-D or the code for the exact implementation details. During data collection, actions are chosen either autonomously (e.g., with RAND in Section IV-B.1) or by a human via a point-and-click graphical user interface (see the appendix). At execution time, actions are parameterized by outputs from trained models.

To improve the performance of the deployed flattening algorithms, we include two additional primitives: (1) a recentering primitive for when the shirt has drifted too far from the center of the workspace, and (2) a recovery primitive that executes a random action when the coverage is stalled for an extended period of time. See the appendix for ablation studies suggesting the usefulness of such primitives.

B. Flattening Metrics

We perform 10 trials of all flattening algorithms from an initially crumpled state (Figure 3). Crumpling is performed autonomously via a series of 6 actions, each of which grabs the T-shirt at a random point, quickly lifts it into the air, and releases, resulting in an initial coverage of $37.5\% \pm 14.9\%$ over 45 trials. In Table II we report maximum coverage as a percentage of the pixel coverage of a fully flattened shirt, i.e. 47,000 pixels in the shirt mask $\mathbf{o}_t^m$. We also report the number of samples used to train the algorithm, the execution time per action, and the number of actions executed, where we allow a maximum of 100 actions but terminate early if a coverage threshold is reached ($C = 45{,}000$ pixels or 96% of maximally flattened).

C. Folding Metrics

We perform 5 trials of all folding algorithms from an initially flattened state. A-ASM initial states are flattened by LP_0AP_1 while all other initial states are flattened via human teleoperation. In Table III we report the number of actions and execution time per action, and we measure the quality of the final state against a goal configuration (Figure 3) according to two metrics: (1) intersection over union (IoU) and (2) a penalty for edges and wrinkles. IoU is calculated between the shirt mask and the goal template, after rotating and translating the goal to best match the shirt mask. The wrinkle penalty calculates the fraction of pixels in the interior of the shirt mask detected as edges by the Canny edge detector [37]. A high-quality folding episode achieves a high IoU score and low edge penalty; for reference, the scores for a fully folded goal image are provided in Table III as GOAL.

D. Flattening Results

See Table II and Figure 6 for results. We find that fully analytical policies such as RAND and AEP are unable to attain high coverage while HUMAN is able to consistently flatten the garment in 11.9 actions on average, suggesting the efficacy of the pick-and-place action primitive and the value of intelligently selecting pick points. Interestingly, we find that despite training an inverse dynamics model on nearly 4,000 real samples, IDYN is unable to outperform RAND. We hypothesize that the fully flattened goal image $\mathcal{T}$ provided as input is too distant from the encountered states, resulting in a data sample outside the training data distribution.

CRL is better able to leverage the large self-supervised dataset as it attains higher coverage, though it does require more time per action due to thousands of forward passes through the network during planning. However, since the dataset is generated by RAND, which achieves an average maximum coverage of only 55.0%, CRL has trouble producing high-quality actions in the high coverage regime, where it has encountered relatively little data. Modifications to the dataset such as actively interleaving data collection and training with policy execution is an interesting direction for future work. KP also achieves a higher maximum coverage than AEP and RAND, but it is prone to executing regressive actions that prevent it from maintaining this coverage. Results suggest that KP can be improved by (1) autonomous labeling, e.g. with fiducial markers, to avoid human error on challenging states with high self-occlusion, and (2) improvements to the analytic corner-pulling policy, which, for example, can struggle when all visible keypoints are positioned correctly but other keypoints are not visible.

We find that LP_0AP_1 significantly outperforms all other algorithms, rivaling HUMAN-level performance by consistently reaching the threshold coverage C in less than 3 times the amount of actions as HUMAN. We hypothesize that this is due to increased sample efficiency from analytic placing in conjunction with the modeling power of the FCN, which exhibits equivariance by sharing parameters for pixel predictions and is an implicit energy-based model like other state-of-the-art architectures [33], [34].

Finally, we find that DROP, which converges through Q-iteration to a policy that executes a drop if coverage is below 45% and LP_0AP_1 otherwise, is unable to improve upon LP_0AP_1. This may occur due to our modeling of the coverage dynamics of LP_0AP_1 as the same regardless of the current coverage, whereas in reality, LP_0AP_1 improves coverage faster in lower-coverage states (Figure 6). Nevertheless, the DROP framework may be an effective way to combine multiple action primitives given more powerful dynamic primitives, such as bimanual actions that can better leverage aerodynamic effects [27].

E. Folding Results

See Table III for results and Figure 7 for folding episodes. The folding subtask presents unique challenges: (1) data collection and evaluation require an initially flattened state,

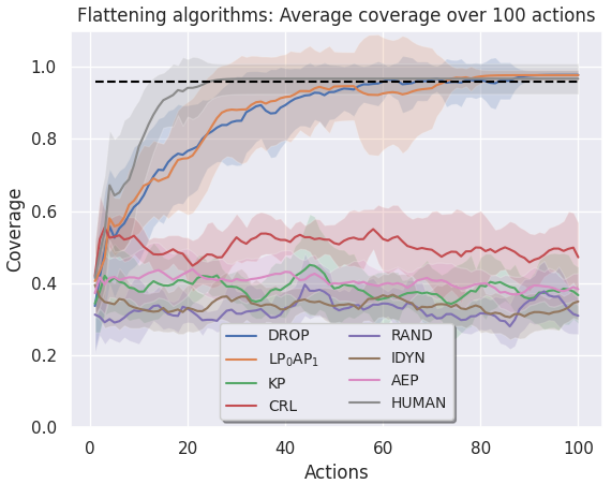

Fig. 6. Coverage vs. time plot for the various flattening policies that we benchmark on the workcell, averaged across 10 rollouts. Shading represents one standard deviation, and the horizontal dashed line is the flattening success threshold (96%).

TABLE II

FLATTENING RESULTS. WE REPORT THE METRICS IN SECTION V-B, WHERE AVERAGES AND STANDARD DEVIATIONS ARE COMPUTED OVER 10 TRIALS.

Algorithm	% Coverage	Actions	Dataset	Time/Act (s)
RAND	55.0 ± 6.0	100.0 ± 0.0	N/A	23.9 ± 2.5
HUMAN	**97.7 ± 3.9**	11.9 ± 5.3	N/A	45.1 ± 18.6
AEP	55.3 ± 5.5	100.0 ± 0.0	N/A	24.6 ± 2.0
IDYN	57.0 ± 5.9	100.0 ± 0.0	3936	23.7 ± 3.7
KP	72.4 ± 9.2	100.0 ± 0.0	681	25.7 ± 2.7
CRL	73.8 ± 8.4	100.0 ± 0.0	3936	32.1 ± 5.3
DROP	**97.7 ± 1.3**	38.6 ± 20.6	524	25.7 ± 0.8
LP_0AP_1	**97.7 ± 1.4**	31.9 ± 17.2	524	25.6 ± 0.9

which is difficult to attain through a remote interface, (2) slightly incorrect actions can dramatically alter the fabric state, often requiring re-flattening the garment, and (3) the single-arm pick-and-place primitive is not well-suited for the precise manipulation required for crisp garment folding. Indeed, we find that even with folding optimizations to pick-and-place (Section V-A), a human teleoperator attains only 76% of the goal IoU on average (Table III). However, we find that both ASM and LP_0LP_1 are able to effectively leverage the primitive to achieve near human-level performance, where ASM performs similarly to LP_0LP_1. We also find that the fully autonomous pipeline A-ASM is able to reach similar performance from an initially *crumpled* state, setting a baseline score for the end-to-end folding task. Although ASM is open-loop and LP_0LP_1 learns from only 2 demonstrations, HUMAN cannot significantly outperform them due to the difficulty of correcting inaccurate actions in folding with only top-down pick-and-place actions. Further progress on the folding subtask will likely require both improved manipulation primitives and algorithmic innovations.

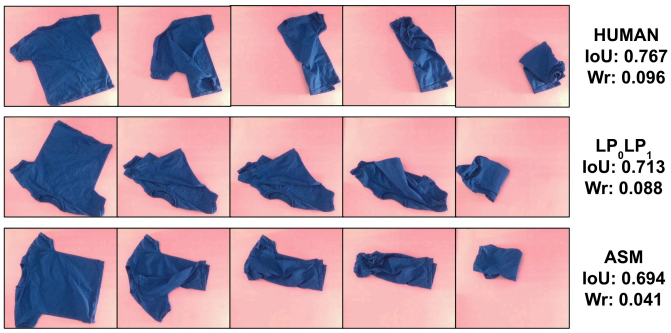

Fig. 7. Representative episodes of the folding subtask executed by HUMAN (Row 1), LP_0LP_1 (Row 2), and ASM (Row 3). LP_0LP_1 and ASM achieve performance competitive with human teleoperation.

TABLE III

FOLDING RESULTS. WE REPORT THE METRICS IN SECTION V-C, WHERE AVERAGES AND STANDARD DEVIATIONS ARE COMPUTED OVER 5 TRIALS.

Algo.	IoU (↑)	Wrinkle (↓)	Actions	Time (s)
GOAL	0.98	0.093	N/A	N/A
HUMAN	**0.74 ± 0.06**	0.088 ± 0.023	4.4 ± 0.5	63.8 ± 15.
ASM	**0.69 ± 0.08**	**0.087 ± 0.038**	4.0 ± 0.0	35.1 ± 1.9
LP_0LP_1	0.68 ± 0.08	0.112 ± 0.032	4.0 ± 0.0	35.7 ± 1.3
A-ASM	0.62 ± 0.12	0.112 ± 0.038	4.0 ± 0.0	35.5 ± 1.7

VI. CONCLUSION AND FUTURE WORK

In this work, we benchmark novel and existing algorithms for T-shirt smoothing and folding tasks on a remote hardware testbed. We find that policies that combine learning with analytical methods achieve the highest performance in practice, suggesting the value of future work in this area.

The ability to access robot hardware remotely, an intuitive API, maintenance by dedicated staff, and the consistency of the task environment all contribute to quick and effective experimentation. On the other hand, onsite technicians have limited availability, variable-latency 2D camera projections are at times insufficient for fully understanding the scene, and manual resets (e.g., flattening the T-shirt) become difficult to perform, suggesting the importance of learning self-supervised reset policies [38].

In future work, we will (1) further optimize performance on the unimanual folding task, (2) evaluate alternative approaches such as continuous control, reinforcement learning, and different action primitives, and (3) evaluate each algorithm's ability to generalize to other garments with variation in color, shape, size, and material.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 737–10 746, 2020.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[4] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," *ArXiv preprint arXiv:2011.07215*, 2020.

[5] J. J. Collins, D. Howard, and J. Leitner, "Quantifying the reality gap in robotic manipulation tasks," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6706–6712, 2019.

[6] A. Wong, A. Zeng, A. Bose, A. Wahid, D. Kalashnikov, I. Krasin, J. Varley, J. Lee, J. Tompson, M. Attarian, P. Florence, R. Baruch, S. Xu, S. Welker, V. Sindhwani, V. Vanhoucke, and W. Gramlich, "Pyreach - python client sdk for robot remote control," https://github.com/google-research/pyreach, 2022.

[7] "Piab pisoftgrip gripper," https://www.piab.com/en-us/suction-cups-and-soft-grippers/soft-grippers/pisoftgrip-vacuum-driven-soft-gripper-/pisoftgrip-/#overview, accessed: 2022-02-21.

[8] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, K. Yamane, S. Iba, J. Canny, and K. Goldberg, "Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

[9] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1699–1706.

[10] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1497–1504.

[11] S. Bauer, F. Widmaier, M. Wüthrich, N. Funk, J. U. D. Jesus, J. Peters, J. Watson, C. Chen, K. Srinivasan, J. Zhang, J. Zhang, M. R. Walter, R. Madan, C. B. Schaff, T. Maeda, T. Yoneda, D. Yarats, A. Allshire, E. K. Gordon, T. Bhattacharjee, S. S. Srinivasa, A. Garg, A. Buchholz, S. Stark, T. Steinbrenner, J. Akpo, S. Joshi, V. Agrawal, and B. Schölkopf, "A robot cluster for reproducible research in dexterous manipulation," *arXiv preprint arXiv:2109.10957*, 2021.

[12] M. Wüthrich, F. Widmaier, F. Grimminger, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, J. Viereck, M. Naveau, L. Righetti, B. Schölkopf, and S. Bauer, "Trifinger: An open-source robot for learning dexterity," in *Conf. on Robot Learning (CoRL)*, 2020.

[13] N. Funk, C. Schaff, R. Madan, T. Yoneda, J. U. De Jesus, J. Watson, E. K. Gordon, F. Widmaier, S. Bauer, S. S. Srinivasa, T. Bhattacharjee, M. R. Walter, and J. Peters, "Benchmarking structured policies and policy optimization for real-world dexterous object manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, p. 478–485, Jan 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3129139

[14] C. Chen, K. P. Srinivasan, J. O. Zhang, and J. Zhang, "Dexterous manipulation primitives for the real robot challenge," *ArXiv preprint arXiv:2101.11597*, 2021.

[15] R. McCarthy, F. R. Sanchez, Q. Wang, D. C. Bulens, K. McGuinness, N. O'Connor, and S. J. Redmond, "Solving the real robot challenge using deep reinforcement learning," *arXiv preprint arXiv:2109.15233*, 2021.

[16] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," *arXiv preprint arXiv:2108.09779*, 2021.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

[18] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman, "Replab: A reproducible low-cost arm benchmark platform for robotic learning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[19] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4368–4375, 2020.

[20] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robotics: Science and Systems (RSS)*, 2017.

[21] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, p. 36–52, Sep 2015. [Online]. Available: http://dx.doi.org/10.1109/MRA.2015.2448951

[22] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Conf. on Robot Learning (CoRL)*, 2019.

[23] C. M. Kim, M. Danielczuk, I. Huang, and K. Goldberg, "Simulation of parallel-jaw grasping using incremental potential contact models," *arXiv preprint arXiv:2111.01391*, 2021.

[24] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth Grasp Point Detection Based on Multiple-View Geometric Cues with Application to Robotic Towel Folding," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010.

[25] A. Doumanoglou, J. Stria, G. Peleka, I. Mariolis, V. Petrík, A. Kargakos, L. Wagner, V. Hlaváč, T.-K. Kim, and S. Malassiotis, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[26] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Conf. on Robot Learning (CoRL)*, 2021.

[27] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conf. on Robot Learning (CoRL)*, 2021.

[28] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation," in *Proc. Robotics: Science and Systems (RSS)*, 2020.

[29] I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenyà, and D. Kragic, "Benchmarking bimanual cloth manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1111–1118, 2020.

[30] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg, "LazyDAgger: Reducing context switching in interactive imitation learning," in *International Conference on Automation Sciences and Engineering (CASE)*, 2021.

[31] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to manipulate deformable objects without demonstrations," in *Proc. Robotics: Science and Systems (RSS)*, 2020.

[32] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, 2017.

[33] P. R. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. S. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conf. on Robot Learning (CoRL)*, 2021.

[34] A. Zeng, P. R. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conf. on Robot Learning (CoRL)*, 2020.

[35] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[36] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2146–2153, 2017.

[37] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[38] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," *arXiv preprint arXiv:2104.11203*, 2021.