2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

MANIP: A Modular Architecture for Integrating Interactive Perception for Robot Manipulation

Justin Yu*[1], Tara Sadjadpour*[1], Abby O'Neill[1], Mehdi Khfifi[1], Lawrence Yunliang Chen[1], Richard Cheng[2], Muhammad Zubair Irshad[2], Ashwin Balakrishna[2], Thomas Kollar[2], Ken Goldberg[1]

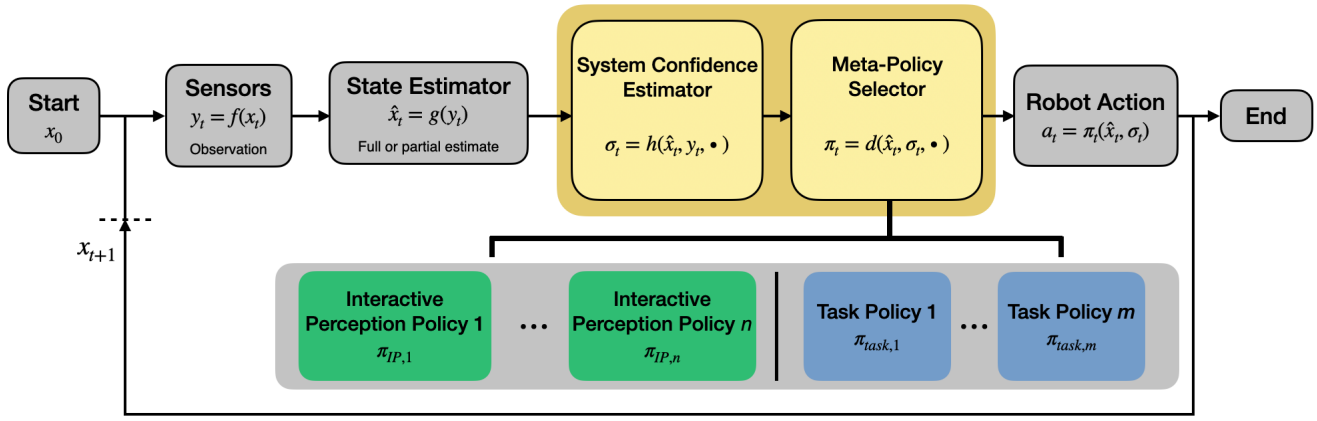


Fig. 1: **MANIP: General MANIP System Architecture** At discrete timestep t, system sensors measure true system state x_t yielding observation y_t, which is processed by the perception model g to estimate state $\hat{x}_t$. The system confidence estimator evaluates both state observability and state viability (see Section III), represented by vector σ_t. Using $\hat{x}_t$ and σ_t, a meta-policy switches between subpolicies for task completion or interactive perception to generate action a_t to advance to state x_{t+1}. This cycle continues until the meta-policy terminates with task completion or an error report.

***Abstract*— We propose a modular systems architecture, MANIP, that can facilitate the design and development of robot manipulation systems by systematically combining learned subpolicies with well-established procedural algorithmic primitives such as Inverse Kinematics, Kalman Filters, RANSAC outlier rejection, PID modules, etc. (aka "Good Old Fashioned Engineering (GOFE)"). The MANIP architecture grew from our lab's experience developing robot systems for folding clothes, routing cables, and untangling knots. To address failure modes, MANIP can facilitate inclusion of "interactive perception" subpolicies that execute robot actions to modify system state to bring the system into alignment with the training distribution and / or to disambiguate system state when system state confidence is low. We demonstrate how MANIP can be applied with 3 case studies and then describe a detailed case study in cable tracing with experiments that suggest MANIP can improve performance by up to 88%. Code and details are available at: https://berkeleyautomation.github.io/MANIP/**

I. Introduction

Many are excited about the potential for generalist robot policies, monolithic end-to-end neural networks that map sensory observations to low-level control outputs, to solve a range of robot control problems [1, 2]. However at least in the near term, it may be helpful to combine learned models with well-established procedural algorithms, aka. "Good Old Fashioned Engineering (GOFE)") to increase generality and reliability [3].

Interactive Perception, where robots change the environment to improve observability, has potential to increase the generality and reliability of robot systems, as Bohg, et al. [4] and others have noted [5–8]. Interactive Perception [4] builds on Active Perception by adding physical interactions with the environment, such as moving an object or occlusion. This paper provides an architecture for incorporating interactive perception into robot systems.

We propose *MANIP*, a **M**odular **A**rchitecture for i**N**tegrating **I**nteractive **P**erception into robot systems. MANIP integrates classical procedural algorithms with learned policies to address different stages of the task using a meta-policy that determines which policy or primitive to activate based on a general vector of confidence. With MANIP, the system objective can switch between minimizing uncertainty with interactive perception policies and maximizing reward with task policies.

To develop MANIP, we began with an initial flowchart similar to Figure 1 and applied it to two prior systems our lab developed for surgical needle handover [9] and cable untangling [10], and one prior system developed by another lab for cable routing [11]. We iterated, adjusting the architecture to make it compatible with each system.

In our experience, we have found MANIP to be useful as a diagrammatic structure to compare system architectures and to identify opportunities where interactive perception primitives (procedural or learned) can be added to existing systems to address failure modes and increase system performance.

This paper makes 3 contributions:

* Equal contribution
[1]The AUTOLab at UC Berkeley (automation.berkeley.edu).
[2]Toyota Research Institute, Los Altos, CA.

1) MANIP, a modular architecture for integrating interactive perception primitives with learned robot manipulation policies.
2) Case studies mapping 3 prior systems [9–11] to MANIP to illustrate how MANIP may be helpful for comparing and extending existing systems.
3) A detailed case study with physical experiments demonstrating how MANIP can be used to significantly improve the performance of our existing cable tracing system [12].

II. Related Work

A. Robot Systems Architectures

There is a rich history of research on robot systems architectures [13–18]. The Shakey robot [19] in the late 1960s decomposed its architecture into 3 functional components: sensing, planning, and executing [20]. Many systems use a variant of the *sense-plan-act* (SPA) structure [16] or the subsumption architecture [21], which is built hierarchically from layers of interacting finite-state machines called behaviors. Examples include Arkin's motor-control schemas [22] and the autonomous robot architecture (AuRA) [23, 24]. One of the most common architectural designs is the three-tier structure, comprising hierarchical connections between planning, executive, and behavioral control levels [18]. Examples include the reactive action packages (RAPs) system [25], ATLANTIS [26], LAAS [27], Syndicate [28], and NASREM [29]. In general, the decomposition into hierarchical modular subsystems can be along the temporal dimension [30] or based on task abstraction [21, 31–33]. For more details and examples, see the excellent review by Kortenkamp et al. [16].

Recent advances in machine learning and deep learning attempt to replace procedural policies with learned models. For example, the perception model can be a learned Convolutional Neural Network (CNN) [34] or Vision Transformer (ViT) [35], the motion planning optimization solver can be warm-started by a neural network [36], and the low-level controller can be a trained RL policy [37] or trained from behavioral cloning [38].

System dynamics for many tasks can be framed as a partially observable Markov decision process (POMDP), but exact solutions to this problem are intractable since they operate in the continuous state space and may require a full history of actions and observations. One option to finding approximate POMDP solutions is reinforcement learning (RL) which balances exploration and exploitation. Yet it remains a challenge for RL to learn long-horizon tasks with large state-action spaces [39, 40] and sparse rewards [41, 42]. MANIP does not assume probabilistic state and transition dynamics, nor an explicit reward or value function, nor a discounted time horizon. MANIP's state estimator and system confidence estimator are generalizations of the POMDP belief state estimator, as they may be functions of state history or other intrinsic or extrinsic variables. MANIP can be trivially extended to represent non-Markovian processes, whereby state and system history can be integrated.

In addition to recent and ongoing efforts to develop Large end-to-end Vision, Language, Action (VLA) models to robotics [1, 2], large language and vision models (LLMs and VLMs) are being used for end-to-end learned perception [43, 44], planning [45–47], and action modules [48, 49]. End-to-end models have shown promise for sub-problems in navigation [50], locomotion [51, 52], and manipulation [53, 54]. Yet results are mixed [55] and it seems likely that for the near future, Good Old Fashioned Engineering (GOFE) will continue to provide useful modular primitives for tasks such as cable untangling [10] and routing [11], bagging [56], assistive cooking [57], and navigation for extended periods [58].

B. Active Perception

Active perception was introduced concurrently in the computer vision and robotics literature in 1988. Aloimonos et al. [59] argued that interpreting a single, static image from a passive observer presents an ill-posed problem that is nonlinear, lacks regularization, and yields an unstable solution that is susceptible to noisy inputs. By implementing an "active vision" framework for five computer vision tasks, the authors experimentally demonstrate that an active observer that collects multiple views of the scene can define a well-posed, linear problem with a unique, stable solution that requires little or no assumptions. Aloimonos's active vision fundamentally differs from MANIP due to its lack of physical interaction with the environment.

Ruzena Bajcsy pioneered research using vision, touch, and active perception to improve robot perception. In 1984, Goldberg and Bajcsy [60] presented active perception using a robot to actively move a touch sensor to trace object contours. In 1985, Bajcsy explored the differences between active and passive perception [61], and in 1987 Bajcsy et al. [62] consider object exploration through visual and haptic sensing. In 1988, Bajcsy defined active perception as a control strategy for intelligent data acquisition using touch and visual sensing [63]. While active perception aims to explore the environment through multiple modalities, *"interactive perception"* differs in that it physically modifies the environment to facilitate perception [4].

III. MANIP System Architecture

See the caption to Figure 1 for a summary of the MANIP framework.

The system confidence estimator h produces a vector σ_t. This includes a measure of confidence in the state estimate at time t, and also some measure of *state viability*, such as cost-to-go or reward as in RL. Even if state estimation confidence is high, it is important for the meta-policy to identify if the estimated state is unsafe, unfamiliar (out of distribution), or likely to lead to task failure.

The meta-policy can be a procedural or learned function that takes as input the current state and confidence vector and outputs one of the available modular sub-policies. These sub-policies can be either 1) procedural or learned *task* policies which perform actions to advance progress toward task completion, or 2) procedural or learned *interactive perception*

policies which perform actions that aim to improve system perception confidence. These sub-policies should be modular with well-defined trigger conditions based on the current system state and confidence vector so that the meta-policy can operate effectively. If trigger conditions are not disjoint, a meta-policy may select a sub-policy non-deterministically.

We have found that organizing sub-policies in this way can facilitate building hybrid systems that include a mix of procedural and learned policies. This organization can also facilitate system fine-tuning as failure modes are encountered, allowing explicit insertion of sub-polices such as interactive perception that can anticipate and correct for failure modes.

At each timestep t, MANIP executes action a_t from the current sub-policy, moving to timestep $t+1$. The meta-policy terminates the cycle either when success conditions are met, or returns a failure signal, or the system outputs a signal requesting human input when a failure condition is met, a timeout is reached, or the current state and confidence vector do not meet any of the subpolicy trigger conditions.

We think of the meta-policy as Markovian as it primarily uses the current system state and confidence vector values. It also includes a cycle counter to check the timeout condition. But the meta-policy can include a buffer of prior system states that may be useful to avoid looping or facilitate multi-step actions during long-horizon tasks.

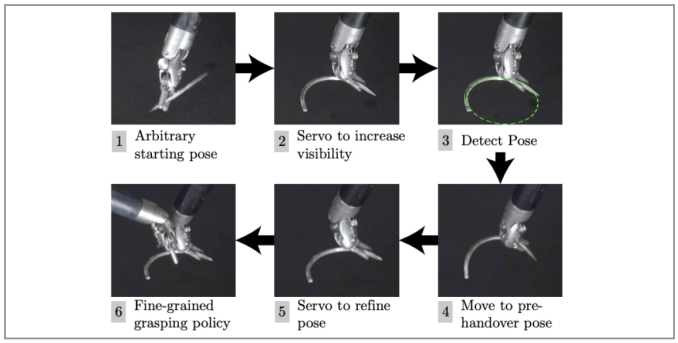
IV. 3 Case Studies with Prior Manipulation Systems

In this section we map 3 prior systems [9–11] to MANIP to explore its utility for comparing and improving existing systems.

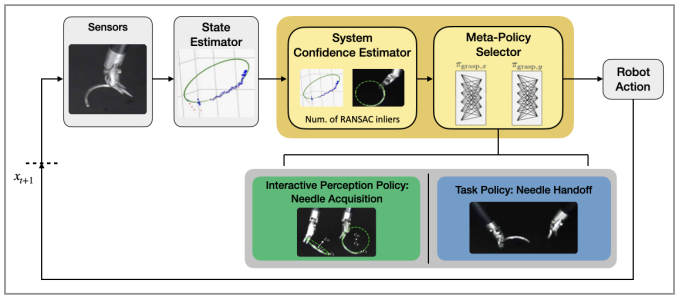
A. Case Study #1: Surgical Needle Handover

As illustrated in Figure 2, Handoff of Unmodified, Surgical, Tool-Obstructed Needles (HOUSTON) is a system from our lab that automates the handover of a surgical needle from one end effector to another [9]. HOUSTON operates on unmodified needles and achieves state-of-the-art success rates on out-of-distribution needles. An interactive perception policy is implemented with visual servoing to actively align the needle until the system confidence estimator determines that a sufficient number of inlier points are identified for RANSAC circle fitting. The task policy subsequently executes the needle handover.

One of the failure modes reported in the paper is incorrect grasp positioning due to uncertainty in needle pose. Mapping the HOUSTON system into the MANIP framework (Figure 2ii) suggests the potential for integrating additional interactive perception policies. One example is a pre-handoff gripper alignment verification using interactive perception. Prior to the second grasp, the receiving gripper can slowly move bi-directionally along the fitted-circle normal direction. A change in the fitted circle pose at both extremes of the gripper jaw can verify that the needle is accurately positioned for a grasp, addressing the reported failure modes and increasing system robustness.



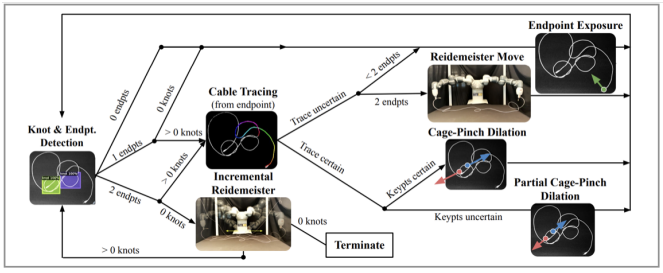
(i) Original System Diagram [7]



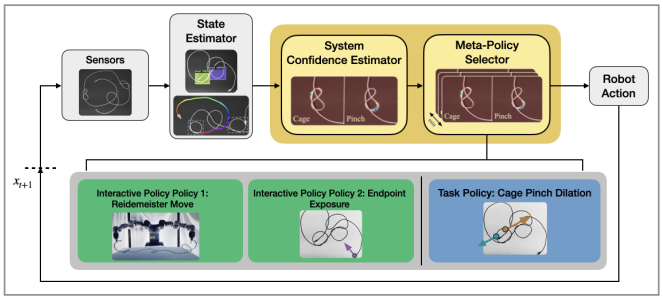
(ii) MANIP

Fig. 2: Two block diagrams of our Surgical Needle Handover System.

B. Case Study #2: Cable Untangling



(i) Original System Diagram [8]



(ii) MANIP

Fig. 3: Two block diagrams of our Cable Untangling System.

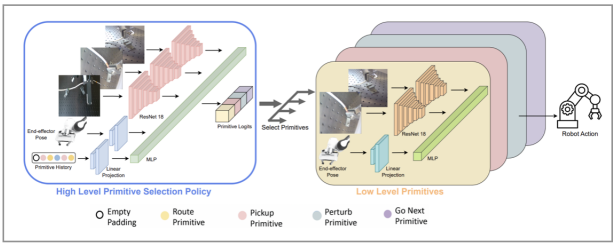
As shown in Figure 3, Sliding and Grasping for Tangle Manipulation (SGTM) 2.0 is a cable untangling system from our lab [10]. Mapping SGTM 2.0 into the MANIP architecture (Figure 3ii) clarifies how the knot and cable endpoint detection systems output spatial probability distributions, which are quantitative measures of uncertainty. The task policy for untangling individual knots is identified as the cage-pinch dilation. To adapt to varying levels of system uncertainty, SGTM 2.0 employs two conditional variations of

cage-pinch dilation actions and Reidemeister moves, and has a meta-policy selector that switches to a partial cage dilation primitive to perturb the state and improve perception rather than attempting to untangle an uncertain knot.

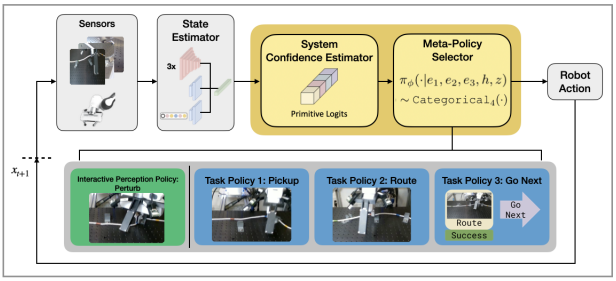
The MANIP architecture (Figure 3ii) suggests how SGTM 2.0 could be extended to address occasional cable grasp failures. We could add an interactive perception primitive after each grasp that attempts to lift and move the grasped cable slightly away from the robot base. By comparing the local pose of the cable before and after this motion, the system can confirm whether or not the cable has been successfully grasped (if no change, the grasp was unsuccessful and should be retried). This could further increase the success rate of SGTM 2.0.

C. Case Study #3: Cable Clip Routing

Luo et al. [11] proposes a multi-stage robot manipulation system to route a cable through a series of clips. The authors use a multimodal perception system to switch between a combination of hand-coded procedural and learning-based primitives. The state estimator takes RGB images of the



(i) Original System Diagram [9]



(ii) MANIP

Fig. 4: Two block diagrams of prior Cable Routing system.

cable, the end effector pose, and the history of executed primitives, and extracts feature maps using neural networks. The system confidence estimator uses an additional neural network to transform a feature map into a probabilistic distribution over 4 motion primitives. It uses a cable perturbation primitive to push the cable out of challenging configurations. The task policy is characterized by a pick-up cable move, a go-next move routing the cable into the next clip, and a clip insertion primitive learned from demonstrations.

Mapping the system into the MANIP architecture (Figure 4ii) suggests a new interactive perception primitive for cases where the clip is not fully within view of the wrist-mounted camera; this primitive would incrementally perturb the wrist until the clip position in the camera image is well within the training distribution. Another interactive perception primitive could address failures where the gripper does not have sufficient clearance around the cable; in such cases, the robot could just push the cable away from the clip.

These 3 case studies suggest that MANIP can facilitate comparing systems and that its modular structure can offer potential extensions, in particular interactive perception primitives to address failure modes. In the next section we report a detailed case study with physical experiments comparing the system before and after mapping it into the MANIP architecture.

V. Implemented Case Study: Cable Tracing

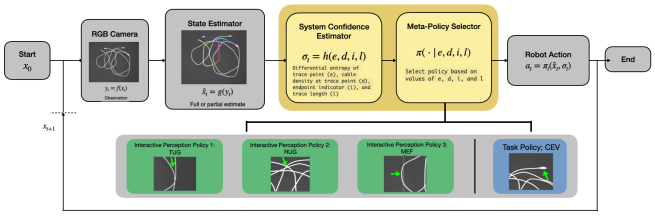


Fig. 5: **HANDLOOM 2.0** in MANIP format. See Section V-C detailing the mapping

Next, we use MANIP to develop a new version of our recently-published cable tracing system called HANDLOOM 1.0 [12]. The resulting system, HANDLOOM 2.0, is shown in Figure 5.

A. Problem Statement

In cable tracing, the system starts with up to 4 standard, white 6 feet-long USB-C to USB-C charging cables randomly arranged on a black planar workspace. When one endpoint of any cable is selected, the objective is to successfully trace the entire cable through various distractors to its other endpoint. All HANDLOOM variations employ a bi-manual ABB YuMi robot and an overhead Photoneo PhoXi Camera, 1m above the flat workspace. The algorithm receives an input of a 773 x 1032 grayscale RGB image (no depth) of the cables on the workspace and a random starting endpoint. The algorithm outputs a trace from the provided starting endpoints. For experiments, we define 4 tiers of difficulty that depend on the number of cables present and the types of cable contacts as illustrated in Figure 7.

Our recently published HANDLOOM 1.0 algorithm [12] uses a vision-based, autoregressive cable tracer, which is not reliable when there is high cable density in the workspace. It often fails on the 3 contacts types outlined in Figure 7, i.e. crossings, tangential contact, and endpoint contact.

B. HANDLOOM 1.5

HANDLOOM 1.5 is an improved version of HANDLOOM 1.0, where multiple analytic trace candidates are generated by HANDLOOM 1.0 and ensembled to choose a cable path candidate with the most traversals. However, HANDLOOM 1.5 still inherits the many endpoint termination failure modes of HANDLOOM 1.0, motivating us to apply the MANIP architecture to improve it.

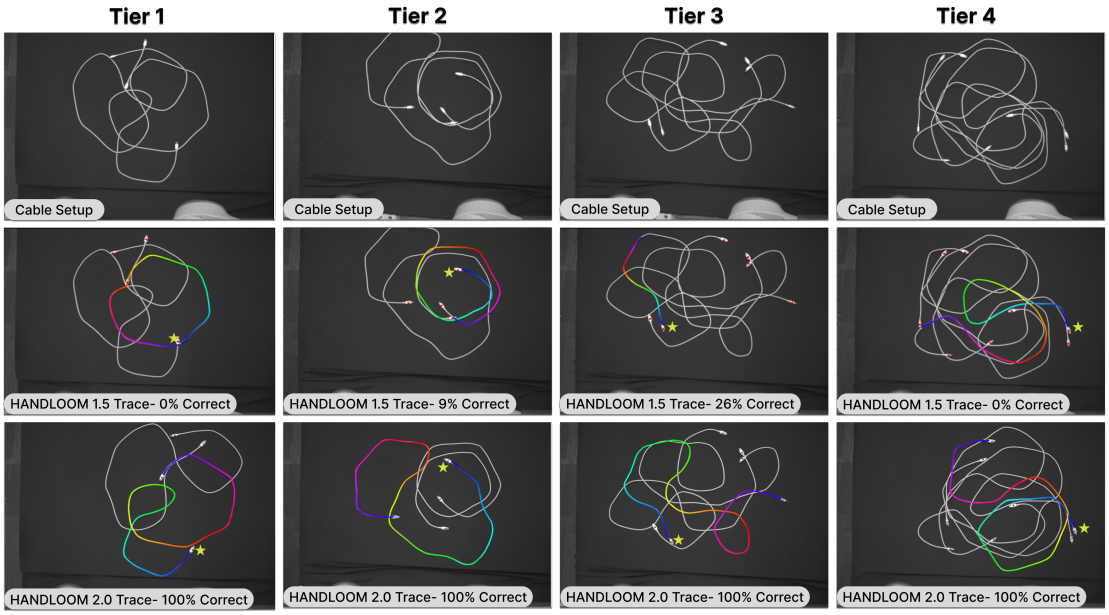


Fig. 6: For each tier, we show one example where HANDLOOM 2.0 outperforms HANDLOOM 1.5. The yellow star is included to represent the chosen starting endpoint.

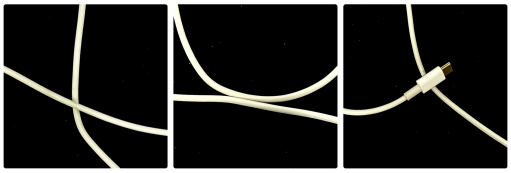


Fig. 7: Three potentially ambiguous configurations for cable tracing. **Left:** Cable crossing. **Center:** Tangential contact. **Right:** Endpoint contact.

C. HANDLOOM 2.0

As MANIP requires an explicit system confidence estimator, at each timestep we fit a 2D Gaussian to the heatmap around for each point p_i on the cable trace to obtain the local mean and covariance, i.e. $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$. Differential entropy, or the uncertainty and randomness of a random variable X, is defined as

$$H(X) = -\int_{\mathcal{X}} p(x) \ln p(x) dx. \tag{1}$$

Applying this to the multivariate Gaussian $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ where $n = 2$, we get

$$\begin{aligned} H(X_i) &= \frac{n}{2}\ln(2\pi) + \frac{1}{2}\ln(|\Sigma_i|) + \frac{n}{2} \\ &= \frac{1}{2}\ln(|\Sigma_i|) + (\ln(2\pi) + 1). \end{aligned} \tag{2}$$

So for every heatmap h_i, uncertainty is measured by the determinant of Σ_i. Higher $|\Sigma_i|$ corresponds to higher uncertainty. Next, for each point p_i, we use the same 65x65 pixel window in the grayscale RGB image and calculate the normalized cable density ρ_i, a value proportionate to the number of white pixels in the fixed window. Higher cable density corresponds to lower state viability, since it is harder to perform cable tracing in areas where there is high overlap and proximity between cables. We calculate an uncertainty quantification U along the trace using a sliding a window filter composed of a linear combination of the differential entropy and density, that is

$$U(p_t) = \Sigma_{i=0}^{T-1} \alpha_i |\Sigma_{i+t}| + \Sigma_{i=0}^{T-1} \beta_i \rho_{i+t}. \tag{3}$$

A parameter sweep suggests that window size $T = 3$ and coefficients $\alpha_0 = \alpha_2 = 0.15$, $\alpha_1 = 0.20$, and $\beta = 0.1\bar{6}$ perform best.

Additionally, we add to the system confidence vector two more values: 1) the length of the current trace in pixel space l, which is a reward measure, and 2) a nominal endpoint "indicator" value: $i \in \{\text{EN, RE, ED, UN}\}$ (endpoint, retrace, edge, or undertermined).

Again, the state confidence vector includes 4 values: 1) state confidence from differential entropy e, 2) state viability from cable density d, 3) trace length l, and 5) endpoint indicator i).

The meta-policy uses the 4 values in σ_t to compute which module to activate. As the MANIP architecture makes sub-policies explicit, we create 4 new sub-policies. Three of these facilitate interactive perception: **T**race **U**ncertainty disambi**G**uation (TUG), **R**etrace **U**ncertainty disambi**G**uation (RUG), and **M**ove **E**ndpoint into **F**ield-of-view (MEF).

When $i = UN$ – the most common case – the TUG interactive perception policy is activated to find $p_c = U_{argmax}$, the point along the trace that has the highest combined differential entropy and cable density, i.e. the highest state uncertainty and lowest state viability. One gripper then perturbs the cable there while the other gripper holds down the starting endpoint.

If the endpoint trace point indicator suggests the final endpoint is at the edge of the field-of-view ($i = ED$),

then one robot gripper picks that ending trace point and moves that portion of the cable back into the camera field-of-view, while the other robot gripper holds down the starting endpoint. If $i = RE$, then the RUG policy is deployed. For RUG, one robot gripper perturbs the area where the retrace occurred, while the other gripper pins down the starting endpoint for the trace.

We define a new subpolicy to confirm task completion, called **C**able **E**ndpoint **V**erification (CEV). When the estimated cable trace length is the expected length of 6-feet and $i = EN$, then the meta-policy selector chooses the task policy, CEV, which uses one robot gripper to hold down the initial cable point, and the other gripper to slightly perturb the current endpoint. Then, the state estimator (HANDLOOM 1.5) is run again and if the trace length stays the same, HANDLOOM 2.0 terminates successfully; otherwise, the meta-policy selects one of the 3 interactive perception policies which perturb the cable to faciliate perception. For example, TUG computes a linear motion of the closed gripper that sweeps between tangentially aligned cable segments in an effort to separate them.

D. Results from 240 Physical Experiments

We compare HANDLOOM 2.0 with HANDLOOM 1.5 using cables tangled into 4 tiers of difficulty. To evaluate performance, we compute the percentage of the cable (with given endpoint) that is correctly traced as measured in pixel space. We set the maximum time horizon to $T = 10$ iterations.

The tiers of difficulty are based on number of cables and types of contact as shown in Figure 7. For all tiers of difficulty, we put no upper bounds on crossings and endpoint contacts. Tier 1 consists of 2 cables with at most 2 tangential contacts. Tier 2 is also 2 cables, but it has more than 2 tangential contacts. Tier 3 is any configuration with 3 cables without an upper bound on tangential contacts. Tier 4 includes 4 cables with no upper bound on tangential contacts. We do our best to randomize initial cable states by dropping cables from above the workspace and using rejection sampling. The robot operates 30 times using each algorithm on each tier of difficulty, constituting a total of 240 trials.

TABLE I: **Results from 240 physical cable tracing experiments comparing HANDLOOM 1.5 with HANDLOOM 2.0 (developed with MANIP)** Values are percent of the cable length that is accurately traced.

Method	Tier 1	Tier 2	Tier 3	Tier 4
HANDLOOM 1.5	62.4%	45.8%	44.8%	60.0%
HANDLOOM 2.0	**89.5%**	**86.2%**	**60.0%**	**68.2%**
% Improvement	*43.4%*	*88.2%*	*33.9%*	*13.7%*

Results in Table I suggests that HANDLOOM 2.0 developed with MANIP increased system performance across all tiers of difficulty.

VI. Limitations and Future Work

In the extended case study with cable tracing, we found that HANDLOOM 2.0 struggles to accurately recognize cable endpoint contacts (see Figure 7), so in future work we will use MANIP to develop HANDLOOM 3.0 with additional interactive perception primitives that can resolve these ambiguities.

We fully acknowledge that MANIP requires human engineering to customize system design and tune parameters for different tasks, and that all systems are prone to failure when assumptions are violated. We are applying MANIP to other prior and new robot manipulation systems in our lab and we welcome others to consider using it. For convenience, the system block diagram in different formats is available on the project website.

Acknowledgement

This research was supported in part by a donation from the Toyota Research Institute.

References

[1] O. X.-E. Collaboration *et al.*, *Open X-Embodiment: Robotic learning datasets and RT-X models*, 2024.

[2] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine, *Octo: An open-source generalist robot policy*, `https://octo-models.github.io`, 2023.

[3] K. Goldberg, *Is data all you need? large robot action models and good old fashioned engineering*, `https://bit.ly/Is_Data_All_You_Need`, 2024.

[4] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[5] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, "Object finding in cluttered scenes using interactive perception," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 8338–8344.

[6] R. M. Martin and O. Brock, "Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 2494–2501.

[7] D. Katz and O. Brock, "Manipulating articulated objects with interactive perception," in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 272–277.

[8] D. Katz, A. Orthey, and O. Brock, "Interactive perception of articulated objects," in *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, Springer, 2014, pp. 301–315.

[9] A. Wilcox, J. Kerr, B. Thananjeyan, J. Ichnowski, M. Hwang, S. Paradis, D. Fer, and K. Goldberg, "Learning to localize, grasp, and hand over unmodified surgical needles," in *2022 ICRA*, IEEE, 2022.

[10] K. Shivakumar, V. Viswanath, A. Gu, Y. Avigal, J. Kerr, J. Ichnowski, R. Cheng, T. Kollar, and K. Goldberg, "Sgtm 2.0: Autonomously untangling long cables using interactive perception," in *2023 ICRA*, IEEE, 2023.

[11] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine, "Multi-stage cable routing through hierarchical imitation learning," *IEEE Transactions on Robotics*, 2024.

[12] V. Viswanath, K. Shivakumar, M. Parulekar, J. Ajmera, J. Kerr, J. Ichnowski, R. Cheng, T. Kollar, and K. Goldberg, "Handloom: Learned tracing of one-dimensional objects for inspection and manipulation," in *CoRL*, PMLR, 2023.

[13] T. L. Dean and M. P. Wellman, *Planning and control*. Morgan Kaufmann Publishers Inc., 1991.

[14] T. L. Dean, *Robot architectures*. [Online]. Available: `https://cs.brown.edu/people/tdean/courses/cs148/02/architectures.html`.

[15] F. G. Martin, *Robotic explorations: A hands-on introduction to engineering*. Prentice Hall PTR, 2000.

[16] D. Kortenkamp, R. Simmons, and D. Brugali, "Robotic systems architectures and programming," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Cham: Springer International Publishing, 2016, pp. 283–306. [Online]. Available: `https://doi.org/10.1007/978-3-319-32552-1_12`.

[17] R. R. Murphy, *Introduction to AI robotics*. MIT press, 2019.

[18] J. Bohg, M. Pavone, and D. Sadigh, *14 robot system architectures*, Feb. 2021. [Online]. Available: https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_14.pdf.

[19] N. J. Nilsson, *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.

[20] N. J. Nilsson, *Principles of artificial intelligence*. Springer Science & Business Media, 1982.

[21] R. Brooks, "A robust layered control system for a mobile robot," *IEEE journal on robotics and automation*, vol. 2, no. 1, 1986.

[22] R. C. Arkin, "Motor schema—based mobile robot navigation," *The International journal of robotics research*, vol. 8, no. 4, 1989.

[23] R. C. Arkin, "Integrating behavioral, perceptual, and world knowledge in reactive navigation," *Robotics and autonomous systems*, vol. 6, no. 1-2, pp. 105–122, 1990.

[24] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2-3, 1997.

[25] R. J. Firby, *Adaptive execution in complex dynamic worlds*. Yale University, 1989.

[26] E. Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots," in *AAAi*, vol. 1992, 1992, pp. 809–815.

[27] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "An architecture for autonomy," *The International Journal of Robotics Research*, vol. 17, no. 4, pp. 315–337, 1998.

[28] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated multiagent teams and sliding autonomy for large-scale assembly," *Proceedings of the IEEE*, vol. 94, no. 7, 2006.

[29] J. S. Albus, H. McCain, and R. Lumia, "Nasa/nbs standard reference model for telerobot control system architecture (nasrem)," 1989.

[30] J. S. Albus, "Rcs: A reference model architecture for intelligent systems," in *Working Notes: AAAI Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents*, 1995, pp. 1–6.

[31] R. J. Firby, "An investigation into reactive planning in complex domains.," in *AAAI*, vol. 87, 1987, pp. 202–206.

[32] R. G. Simmons, "Structured control for autonomous robots," *IEEE transactions on robotics and automation*, vol. 10, no. 1, 1994.

[33] J.-J. Borrelly, É. Coste-Maniere, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, and N. Turro, "The orccad architecture," *IJRR*, vol. 17, no. 4, 1998.

[34] 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," 2017.

[35] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[36] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, eabd7710, 2020.

[37] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine, "Zero-shot robotic manipulation with pretrained image-editing diffusion models," *arXiv arXiv:2310.10639*, 2023.

[38] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng, "Do as i can and not as i say: Grounding language in robotic affordances," in *6th CoRL*, 2022.

[39] A. Fayad and M. Ibrahim, "Behavior-guided actor-critic: Improving exploration via learning policy behavior representation for deep reinforcement learning," *arXiv preprint arXiv:2104.04424*, 2021.

[40] Z. Ren, R. Guo, Y. Zhou, and J. Peng, "Learning long-term reward redistribution via randomized return decomposition," in *International Conference on Learning Representations*, 2021.

[41] H. Rahmandad, N. Repenning, and J. Sterman, "Effects of feedback delay on learning," *System Dynamics Review*, vol. 25, no. 4, pp. 309–338, 2009.

[42] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[43] M. Kwon, H. Hu, V. Myers, S. Karamcheti, A. Dragan, and D. Sadigh, "Toward grounded social reasoning," *arXiv preprint arXiv:2306.08651*, 2023.

[44] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation," in *Proceedings of the IEEE/CVF CVPR*, 2023.

[45] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 11 523–11 530.

[46] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," 2022.

[47] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*, PMLR, 2022, pp. 9118–9147.

[48] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9493–9500.

[49] M. G. Arenas, T. Xiao, S. Singh, V. Jain, A. Z. Ren, Q. Vuong, J. Varley, A. Herzog, I. Leal, S. Kirmani, *et al.*, "How to prompt your robot: A promptbook for manipulation skills with code as policies," in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[50] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 ICRA*, IEEE, 2018.

[51] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *RSS*, 2021.

[52] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on Robot Learning*, PMLR, 2023, pp. 2226–2240.

[53] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[54] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[55] N. J. Kumar, *Will scaling solve robotics? the idea of solving the biggest robotics challenges by training large models is sparking debate*, https://spectrum.ieee.org/solve-robotics, 2024.

[56] L. Y. Chen, B. Shi, D. Seita, R. Cheng, T. Kollar, D. Held, and K. Goldberg, "Autobag: Learning to open plastic bags and insert objects," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3918–3925.

[57] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K. Y. Chen, N. Chin, P. Dan, X. Fan, *et al.*, "Mosaic: A modular system for assistive and interactive cooking," *arXiv preprint arXiv:2402.18796*, 2024.

[58] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra, *et al.*, "Goat: Go to any thing," *arXiv preprint arXiv:2311.06430*, 2023.

[59] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International journal of computer vision*, vol. 1, pp. 333–356, 1988.

[60] K. Y. Goldberg and R. Bajcsy, "Active touch and robot perception," *Cognition and Brain Theory*, vol. 7, no. 2, pp. 199–214, 1984.

[61] R. Bajcsy, "Active perception vs. passive perception," in *Proc. of IEEE Workshop on Computer Vision*, 1985, pp. 55–62.

[62] R. Bajcsy, S. J. Lederman, and R. L. Klatzky, "Object exploration in one and two fingered robots," in *Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Computer Society Press, New York*, vol. 3, 1987, pp. 1806–1810.

[63] R. Bajcsy, "Active perception," *Proceedings of IEEE*, vol. 76, no. 8, 1988.