Generalizable Robotic Insertion with World Models

Nicklas Hansen[12], Iretiayo Akinola[1], Yijie Guo[1], Jie Xu[1], Bingjie Tang[13],
Hao Su[2], Xiaolong Wang[12], Abhishek Gupta[1], Dieter Fox[1], Yashraj Narang[1]

[1]**NVIDIA** [2]**UC San Diego** [3]**USC**

***Abstract*—Robotic assembly in high-mixture settings requires adaptable systems that can handle diverse parts, yet current approaches typically rely on policies specialized to each insertion task. Although this can reach high success rates, it makes the process of deploying systems for new problems tedious and time consuming. We present a framework for generalizable insertion using world models that combine robot proprioceptive information with raw visual observations captured by a wrist-mounted camera. Our model-based approach trains a *single* world model on up to 90 insertion tasks with geometrically diverse parts, achieving 56% zero-shot success on unseen objects with unknown geometry compared to just 7% with a model-free baseline. Importantly, performance improves as more objects are included in the training dataset, demonstrating strong scalability. Lastly, finetuning the generalist model on held-out objects significantly enhances data-efficiency compared to training from scratch and, in some cases, achieves better asymptotic performance. To our knowledge, this is the first system capable of assembling unseen objects in an entirely data-driven manner, and thus represents a significant step toward scalable, generalizable robotic assembly.**

I. INTRODUCTION

Autonomous robotic assembly [36, 12, 35, 33, 31, 32, 23] is a challenging yet important research topic in manufacturing and automation, where precise manipulation of diverse objects is needed. State-of-the-art approaches in industry typically rely on hand-crafted controllers that cannot easily generalize across different object geometries and scene configurations. Existing learning-based systems in research predominantly use specialist policies, requiring extensive training (and often part-specific tuning) for each specific object pair, which fundamentally limits scalability and adaptability. The key challenge in going beyond specialist policies has been the inability of the currently-chosen algorithms to tractably obtain controllers that can achieve both dexterity and generalization.

In contrast to current systems, we envision future robotic assembly systems that have the flexibility and robustness required to manipulate new objects with minimal additional engineering, assumptions, or privileged information. To realize this vision, we argue that such a generalizable robotic assembly system demands adaptive control policies as well as robust visual perception capable of inferring detailed object geometry and poses directly from raw visual feedback within a closed-loop control system. While the current paradigms for acquiring learning-based controllers for assembly do not conceptually preclude such a vision, they struggle to scale up the learning of generalist behaviors to large numbers of tasks and rich perceptual inputs [32]. This begs the question - as opposed to simply exposing methods to more data and compute, does a change in learning *algorithm* potentially help develop generalist assembly policies more effectively?

Our research addresses these questions by introducing a learning-based framework that enables zero-shot insertion of unseen objects with entirely unknown object geometries. Concretely, we develop a system for general-purpose robotic assembly that can leverage world models and visual observations to combine robot proprioceptive information with raw depth observations captured by a wrist-mounted camera. Departing from the typical on-policy reinforcement learning (RL) approach for acquiring robotic assembly controllers, we posit that off-policy world-model based approaches enable more efficient multi-task, vision-based policy learning. Doing so allows for training generalist, vision-based policies across wider ranges of problems, without compromising dexterity, a challenge that has proven difficult with prior policy gradient-driven RL approaches to assembly.

Our approach learns a generalizable policy without relying on object-specific priors. By training a multi-task policy on up to 90 diverse assemblies from Tang et al. [32], our method achieves a remarkable 56% zero-shot success rate on unseen assemblies with *unknown* geometry, and we show that performance improves as more assemblies are added, demonstrating strong scaling behavior. Lastly, we find that finetuning a generalist model on held-out assemblies significantly enhances data-efficiency compared to training from scratch and, in some cases, achieves better asymptotic performance. Our key contribution in this work is not to propose a new algorithm, but rather to demonstrate that visual model-based RL algorithms have immense untapped potential for industrial assembly, allowing for much broader and more performant generalist, multi-task policies. We carefully study the impact of various decisions in the construction of such a system, and provide a practitioners' guide to building general-purpose assembly systems using world-model-based controller synthesis methods.

II. RELATED WORK

Our work is at the intersection of robotic assembly and RL, areas that have both enjoyed remarkable progress in recent years. This section aims to summarize previous work in these two areas, with a particular emphasis on literature related to our contributions.

Autonomous robotic assembly tasks of interest to the research community often include mating of two distinct parts, with insertion of one object into another being a common yet difficult task due to intricate part geometries, asymmetry, precise control requirements, and the generally small scale of parts to be assembled [7, 36, 12, 5, 35, 33]. Several

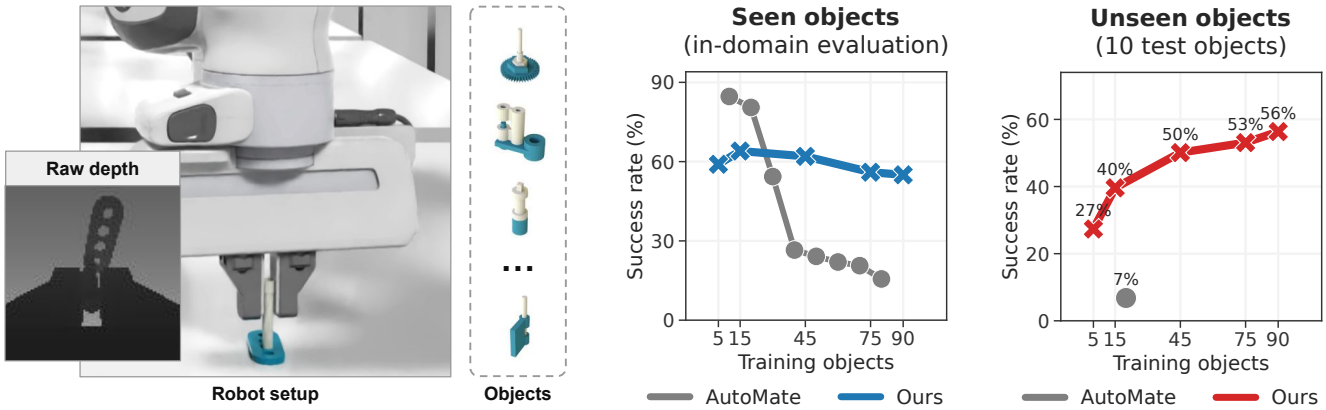

Fig. 1. **A generalist for insertion.** When pretrained on a large number of objects (**blue**), our method scales substantially better than the previous state-of-the-art generalist from Tang et al. [32] (**gray**). Furthermore, our method generalizes zero-shot to unseen object geometries (**red**).

recent works have explored learning of policies for robotic assembly via simulation environments designed to mimic a real-world setup, and these works demonstrate that such policies subsequently can be deployed on real hardware with limited reduction in performance [6, 31, 32, 23]. However, due to the immense work required to build such simulation environments, development of better algorithms for policy learning has received comparably less attention from the research community. For example, AutoMate [32] proposes a system for training part-specific specialist policies to perform insertion in simulation which are then deployed on a real robot setup that closely matches the simulation environment. To do so, the authors curate and preprocess a set of 100 assembly asset pairs originally introduced by Willis et al. [35], Tian et al. [33], and build a parallelizable simulation environment with observation and action spaces that are also accessible in the real world, as well as a shaped reward function that encourages reliable top-down insertion strategies. The authors demonstrate that their system enables training of Proximal Policy Optimization (PPO) [29] via online RL on individual object pairs, and include an exploratory experiment in which a number of learned specialist policies are distilled into a (blind) generalist insertion policy. In this work, we focus on the algorithmic aspect of robotic assembly and set out to develop a learning-based system that can insert *unseen* objects with unknown geometry in a zero-shot manner, relying purely on raw visual observations from a wrist-mounted camera.

Reinforcement learning for robotics has been of great interest to the research community over the past decade, with applications including tabletop manipulation [14, 18, 26, 22, 38, 37, 16, 4], dexterous manipulation [24, 39, 27, 9], and outdoor locomotion [17, 15, 21, 2, 3]. A common property of these applications is that they require learning robust control policies that can operate in increasingly noisy and unstructured environments (cannot easily be programmed), yet *the tasks themselves are often quite forgiving in terms of precision of motions* [40]. In this work, we focus on learning *precise* visual policies for assembly of (unseen) objects with unknown geometry, a significantly more challenging problem setting than what RL is typically used for in robotics literature. To achieve our goal, we build upon model-based RL algorithm TD-MPC2 [10, 11], which has been shown empirically to outperform contemporary RL algorithms across a variety of continuous control tasks. Notably, this is (to the best of our knowledge) the first successful application of model-based RL algorithms in the area of robotic assembly.

III. PRELIMINARIES

Setup. We aim to develop a learning-based robotic assembly system capable of high-precision insertion of unseen objects with unknown geometry, entirely from raw visual observations captured by a wrist-mounted camera and robot proprioceptive information. To do so, we leverage Isaac [20], a GPU-accelerated and highly parallelizable simulation tool, for iterative data collection and RL. Concretely, our experimental setup consists of *(1)* a Franka Panda robotic manipulator with a parallel jaw gripper mounted to a table-top on which assembly occurs, *(2)* a wrist-mounted camera mimicking the common Intel RealSense D435 RGB-D camera, and *(3)* 200 simulated assets (100 object pairs) for training of continuous control policies for the highly precision-based task of insertion. While our experiments are conducted in simulation, such a setup can in principle be replicated on real hardware as demonstrated by recent related work [6, 31, 32, 23], though they rely solely on pose estimation rather than visual inputs. Refer to Figure 1 (*left*) for an illustration of our setup.

Robotic insertion with RL. We model the robotic insertion problem as a Markov Decision Process (MDP) [1] characterized by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{T}, \mathscr{R}, \gamma)$ where $\mathbf{s} \in \mathscr{S}$ are states, $\mathbf{a} \in \mathscr{A}$ are actions (6 DoF delta end-effector poses normalized to the $[-1, 1]$ interval), $\mathscr{T}: \mathscr{S} \times \mathscr{A} \mapsto \mathscr{S}$ is the environment transition (dynamics) function, $\mathscr{R}: \mathscr{S} \mapsto \mathbb{R}$ is a scalar reward function that encourages completion of the insertion task, and γ is a constant discount factor. Since the full state of the environment $\mathbf{s}$ cannot be easily determined in real-world applications, we instead rely on raw depth observations

$\mathbf{x} \in \mathbb{R}^{96\times96}$ and robot proprioceptive information $\mathbf{q} \in \mathbb{R}^{24}$, and approximate $\mathbf{s}$ as $[\mathbf{x}, \mathbf{q}]$, resulting in a *partially observable* MDP [13]. The objective is then to learn a policy $\pi : \mathscr{S} \mapsto \mathscr{A}$ such that cumulative discounted rewards $\mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$, $r_t \doteq \mathscr{R}(\mathbf{s}_t)$ (denoted as *return*) is maximized. In this work, we derive our policy π from iteratively optimizing a learned world model on collected data, and collecting new interaction data by planning with the learned model.

Learning a world model with TD-MPC2 [11], a model-based RL algorithm that plans actions via local trajectory optimization in the latent space of a learned world model. Specifically, TD-MPC2 learns a latent decoder-free world model from environment interaction data and leverages Model Predictive Path Integral (MPPI) [34] with a learned policy prior as a derivative-free (sampling-based) trajectory optimizer during inference (*planning*). All components of the world model are learned end-to-end from data using an optimization objective that combines auto-regressive joint-embedding prediction [8], reward prediction, and a temporal difference (TD) [30, 19] loss, *without* decoding raw observations. In this work, we build upon TD-MPC2 due to its demonstrably strong empirical performance on simpler robotic manipulation tasks, and develop a system for generalizable, vision-based robotic insertion. In the following, we provide more details on the TD-MPC2 algorithm as used in our work; details not pertaining to our use of the algorithm are omitted for brevity.

−Model architecture. TD-MPC2 learns a latent world model that consists of 5 components:

Encoder	$\mathbf{z} = h(\mathbf{s})$	▷ Produces latent state
Dynamics	$\mathbf{z}' = d(\mathbf{z}, \mathbf{a})$	▷ Predicts next latent state
Reward	$\hat{r} = R(\mathbf{z})$	▷ Predicts reward r
Terminal value	$\hat{q} = Q(\mathbf{z}, \mathbf{a})$	▷ Predicts future return
Policy prior	$\hat{\mathbf{a}} \sim p(\mathbf{z})$	▷ Predicts optimal action

where $\mathbf{z}$ is the latent state. During training, our agent autonomously collects data via interaction, and selects actions via planning. Our agent maintains a limited capacity (first-in-first-out) dataset (*replay buffer*) $\mathscr{B}$ of collected trajectories, which is used to optimize the world model.

−Training objective. The h, d, R, Q components of the world model are jointly optimized to minimize the auto-regressive prediction objective $\mathscr{L}(\theta) \doteq \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')_{0:H}\sim\mathscr{B}}\left[\sum_{t=0}^{H} \lambda^t L(\theta; t)\right]$, where $\lambda \in \mathbb{R}_+$ is a constant coefficient that weighs near-term predictions higher than predictions further into the future, H is a fixed sequence length (horizon), and L is a single-step loss

$$L(\theta; t) = \underbrace{\mathrm{CE}(\hat{r}_t, r_t)}_{\text{Reward prediction}} + \underbrace{\mathrm{CE}(\hat{q}_t, q_t)}_{\text{Value prediction}} + \underbrace{\| \mathbf{z}'_t - \mathrm{sg}(h(\mathbf{s}'_t)) \|_2^2}_{\text{Joint-embedding prediction}} . \tag{1}$$

Here, CE is a cross-entropy objective used for discrete regression of the scalars (r_t, q_t), and sg is a stop-grad operator that prevents gradients from flowing back through latent state prediction targets. The policy prior is trained to maximize entropy and terminal value $Q(\mathbf{z}, \mathbf{a})$ at each step. Refer to Hansen et al. [10, 11] for more details on the TD-MPC2 algorithm.

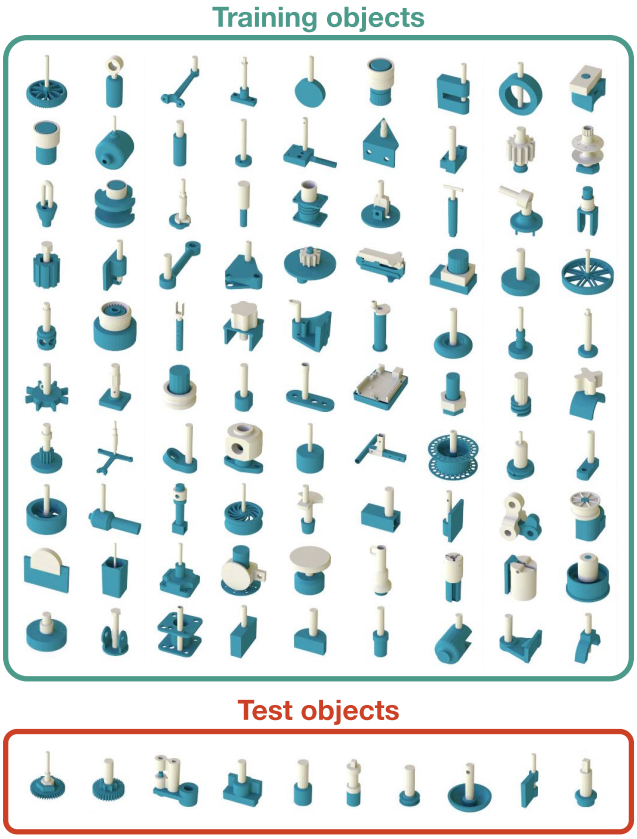

Fig. 2. **Assemblies.** We consider a total of 100 assemblies from Tang et al. [32], which vary greatly in geometry. We reserve 10 assemblies for testing and train on the last 90.

IV. DATASET AND SIMULATION ENVIRONMENT

We first introduce our chosen asset dataset, and then describe our simulation environment. Additional details are provided in Appendix C.

Dataset for robotic assembly. We base our experiments on the assembly asset dataset proposed by Tang et al. [32], which consists of 100 geometrically diverse two-part assemblies that themselves are based on previous work Willis et al. [35] and Tian et al. [33]. Despite significant progress in simulation tools in recent years, making assembly assets simulatable remains a tedious, human labor-intensive task, and thus we limit ourselves to these 100 object pairs. We emphasize, however, that our method is not limited to a fixed number of objects and could – without any algorithmic changes – be applied to larger assembly datasets as they become available. The assemblies considered in this work are geometrically diverse and often feature asymmetries along at least one axis. As a result, insertion strategy can vary significantly between objects. Figure 2 provides an overview of our assets; the ***plug*** is to be inserted into the ***socket***.

Simulation environment. We build upon the environment proposed by Tang et al. [32], which features a Franka Panda robotic manipulator equipped with a parallel-jaw gripper, and closely follow their experimental setup to ensure that comparison to prior work is fair. The socket is randomly

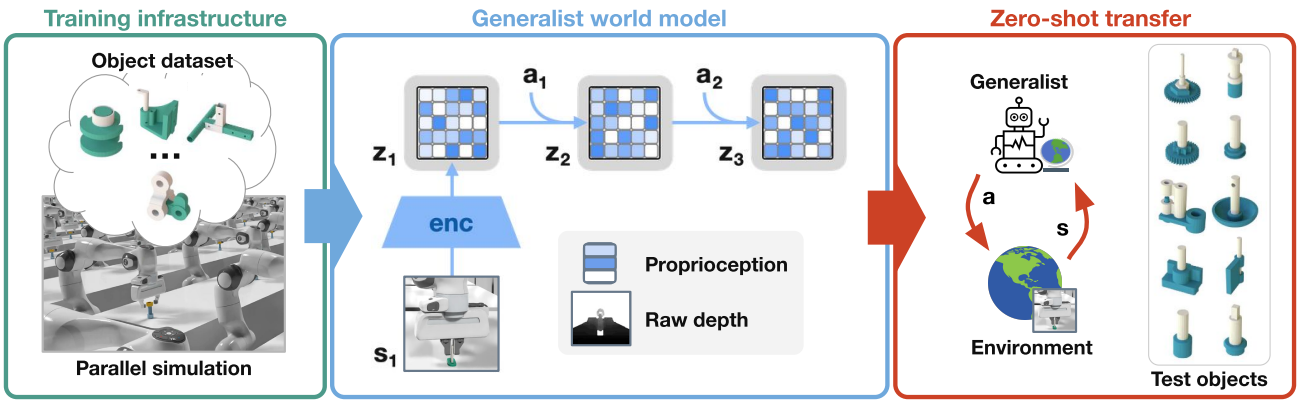

Fig. 3. **Overview.** We present a learning-based system for **zero-shot** insertion of unseen objects with unknown geometry. Our framework learns a **generalizable world model** that at each time step takes robot proprioceptive information and raw depth observations captured by a wrist-mounted camera as input, and outputs delta end-effector pose commands for the robot.

initialized within reach of the robot, in an upright pose such that an approximately top-down insertion is possible. The plug is similarly randomly initialized and is grasped via motion planning before the start of each RL trajectory; this ensures that our learning framework is focused on the object insertion itself for which classical robot planning methods are prone to fail. To facilitate learning and improve rate of convergence, we employ a simple training curriculum that adjusts the initial state distribution based on current training success rate. The curriculum is adjusted to each assembly individually when training multi-assembly models. Observations and rewards are defined as follows:

– ***Observations.*** We consider observations that include raw depth observations $\mathbf{x} \in \mathbb{R}^{96\times96}$ from a wrist-mounted camera, as well as robot proprioceptive information $\mathbf{q} \in \mathbb{R}^{24}$. The proprioceptive vector consists of robot joint angles ($\mathbb{R}^7$), noisy estimates of the 6D end-effector and socket poses ($2\times\mathbb{R}^7$) with white noise applied to mimic real-world sensor noise, and a goal position ($\mathbb{R}^3$) at which the plug is considered fully inserted. See Appendix 10 for a visualization of observations.

– ***Reward function.*** Our reward function has four terms: *(1)* negative distance between plug and goal, *(2)* a penalty for interpenetration errors in the rare event that they occur, *(3)* a binary term that rewards task success, and *(4)* an imitation-based reward that encourages the policy to mimic a small set of demonstrations derived by reversing procedurally-generated disassembly trajectories [25, 32].

V. GENERALIZABLE ROBOTIC INSERTION

We present a learning-based system for *zero-shot robotic insertion of unseen objects with unknown geometry*. Our framework learns a generalizable world model that at each time step takes robot proprioceptive information and raw depth observations captured by a wrist-mounted camera as input, and outputs delta end-effector pose commands for the robot to execute via planning with the learned world model. Crucially, our method does not rely on *any* privileged information during deployment, thus making deployment on real hardware feasible in the future. Figure 3 provides an overview of our proposed system. In this section, we describe algorithmic details specific to our problem setting while a more general description of the RL training pipeline is provided in Section III.

Encoder for multi-modal observations. Previous work on learning-based robotic assembly [31, 32, 23] focus on learning object-specific (specialist) policies for each unique assembly, which does not require substantial knowledge of object geometry beyond what can be deduced through robot proprioceptive information. However, we argue that visual feedback is necessary if we are to learn a *single* policy (or world model) that can manipulate *multiple* objects, including objects not seen during training. To this end, we additionally provide our world model with raw depth information from a wrist-mounted camera, and design the encoder h to consist of three modules: *(1)* a shallow ConvNet for depth observations, *(2)* a 2-layer MLP for robot proprioceptive information, and *(3)* a 2-layer MLP that fuses the two encoder outputs into a single latent state $\mathbf{z}$, formally defined as

$$h(\mathbf{x},\mathbf{q}) \doteq h_{\text{fuse}}\left(h_{\text{depth}}(\mathbf{x}) + h_{\text{prop}}(\mathbf{q})\right), \tag{2}$$

where each of $h_{\text{fuse}}, h_{\text{depth}}, h_{\text{prop}}$ apply SimNorm [11] normalization to their output. We use multi-modal observations for both our single-object and multi-object world models. As our experimental results will reveal, the addition of visual observations is crucial to the performance of our framework.

Zero-shot generalization. We train both single-object insertion specialists and generalizable multi-object insertion policies via environment interaction. To fully leverage the parallelizable nature of our simulation environment, we choose to train our agents across multiple robotic insertion instances simulated in parallel. When training specialists, we randomly initialize the same assembly with different initial scene configurations across each environment instance, while we maintain one environment instance per unique assembly when training multi-task world models, *i.e.*, we simulate 90 environment instances each with one of 90 unique assemblies from our dataset. This guarantees a uniform distribution across assemblies. While providing the model with *e.g.* a one-hot encoding

of object IDs could potentially improve performance on *known* objects in a multi-task setting, such a representation cannot easily be transferred zero-shot to new objects. Thus, we opt to rely solely on visual information for object disambiguation in a data-driven manner.

VI. EXPERIMENTS

We validate our approach through rigorous experimental evaluation in the setup described in Section IV, including both single-assembly *specialist* policies as well as multi-assembly *generalist* policies, with our key performance metric being the average insertion success rate across all available assets.

A. Experimental Details

Evaluation. All assemblies introduced in Tang et al. [32] have uniquely numbered identifiers. We choose to reserve the last 10 of 100 assemblies (according to their IDs) for evaluation of out-of-domain generalization (*i.e.*, we maintain a fixed set of *"unseen"* objects), and train generalist world models on the remaining 90 objects. When training generalists on only a subset of these objects, we simply select subsets according to their IDs in ascending order to mitigate any selection bias. See Appendix C for a numbered list of assemblies. Generalist success rates are averaged over all assemblies in the training (in-domain) or test (unseen) sets, respectively, and across 100 trials per assembly. Specialist policies are trained independently on each of 100 assemblies, and we report the mean success rate across 5 independent training runs (random seeds) as well as 100 trials per run.

Baselines. Our primary point of comparison is **AutoMate** [32], the current state-of-the-art method for robotic insertion with RL. AutoMate trains insertion policies in simulation with PPO [29], focusing mostly on specialist (single-assembly) policies with proprioceptive information as input. The trained specialist policies are blind (*i.e.*, no visual information) and are not explicitly provided information about object geometry. In addition to these specialist policies, AutoMate also includes exploratory experiments on a generalist policy. This generalist policy is obtained by first training single-assembly specialist policies, and subsequently distilling them into a single multi-assembly policy using a combination of supervised behavioral cloning, DAgger [28], and online RL finetuning, while conditioning the policy on point cloud embeddings obtained from CAD models of each assembly. This contrasts with our generalist model that relies solely on raw visual inputs and is trained purely with online RL. Our approach greatly simplifies the training pipeline in comparison to AutoMate, and does not require any knowledge of object geometry (such as CAD models). All AutoMate results are reproduced using the official codebase. Additionally, **we compare against a number of variations of our framework**, in particular *(1) "blind"* world models without visual inputs, *(2)* world models with an alternative wrist camera placement, *(3)* world models trained with and without a curriculum that adaptively adjusts initial conditions to be farther or closer to insertion success depending on current training success rate (following the curriculum setting in AutoMate), and *(4)* a generalist with access to 100 expert rollouts (generated by our specialist policies) per assembly. Naturally, access to such expert rollouts during training hinges on the availability of pretrained expert policies, thus severely limiting the scalability of the generalist training pipeline to larger sets of assemblies. The default formulation of our framework thus leverages a training curriculum rather than expert rollouts, and we merely include this additional baseline for completeness.

Implementation details. All world models considered in this work have 5M parameters. Observations include raw depth observations $\mathbf{x} \in \mathbb{R}^{96\times96}$ from a wrist-mounted camera and robot proprioceptive information $\mathbf{q} \in \mathbb{R}^{24}$, and actions are 6 DoF delta end-effector poses normalized to $[-1,1]$. *Specialist* world models are trained for 2M steps vs. 50M steps for AutoMate, and our *generalists* are trained for 7M steps. Training a specialist with visual inputs takes 40h on a NVIDIA RTX 3090 GPU, and training a 90-object generalist takes 3 days. See Section E for more details.

B. Results

Single-assembly specialists. We evaluate specialist policies across all 100 assemblies, and report both individual and aggregate performance metrics for our method and AutoMate in Figure 8. For visual clarity, we order objects from left to right along the horizontal axis according to the average success rate achieved by our approach; see Appendix B for alternative visualizations. Our approach achieves 79.78% success across all 100 assemblies vs. 70.31% for AutoMate. While such a substantial performance improvement is to be celebrated, we do observe a decrease in success rates across a few assemblies, *e.g.* `296` and `726` (visualized in Figure 8). We conjecture that this may be due to differences in exploration between PPO (AutoMate) and TD-MPC2 (ours), as well as visual occlusion. Visual inspection of objects for which there is a large performance gap between our method and AutoMate – positive or negative gain – indicates that our approach tends to struggle with large sockets, presumably due to visual occlusion, while our method on the other hand performs significantly better than AutoMate on small, thin sockets. Regardless, these results clearly demonstrate the efficacy of our approach even in a single-assembly case.

Multi-assembly generalists. Our generalist results are shown in Figure 1 (*right*); we report numbers for both in-domain evaluations (*seen* objects; left) and out-of-domain evaluations (*unseen* objects; right), as well as reference AutoMate numbers obtained from Tang et al. [32]. We note that the exact training and test splits for AutoMate are not known, but that our experimental setups are nonetheless comparable and that the difference in performance cannot be explained by a difference in training/test splits. It is also worth noting that the AutoMate generalist assumes access to point cloud embeddings of ground-truth CAD models for each assembly (including test assemblies), whereas our method relies solely on raw visual observations. Our results demonstrate that our

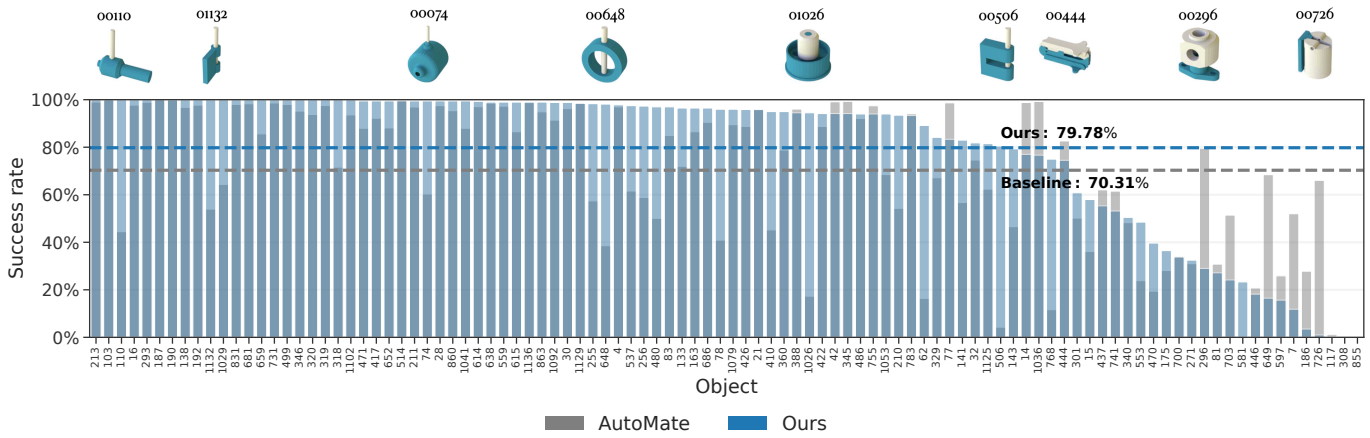

Fig. 4. **Specialist policies.** Our method (**blue**) achieves strong single-assembly performance across 100 diverse objects; 79.78% overall success rate vs. 70.31% for AutoMate (**gray**). Assemblies are ordered by success rate; mean of 5 runs per assembly.

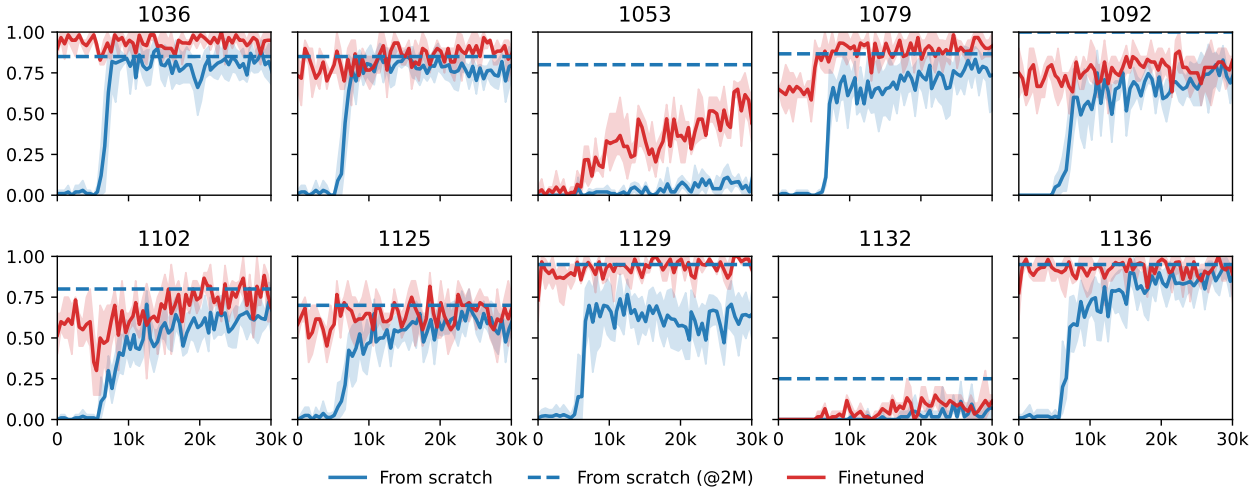

Fig. 5. **Finetuning.** Success rate vs. environment steps for two variants of our method: *(1)* specialists learned from scratch, and *(2)* our 90-object generalist finetuned on each of 10 held-out assemblies. The unique ID of each assembly is reported above each subplot. Finetuning greatly improves data-efficiency. Average of 5 random seeds; shaded area denotes 95% CIs.

model-based RL approach achieves considerably better scaling wrt. number of assemblies, achieving a 55.0% success rate on 90 seen assemblies compared to just 15.6% for AutoMate on 80 seen assemblies. Additionally, we find that the zero-shot generalization to unseen assemblies increases with the number of training assemblies, achieving a remarkable 56% ***zero-shot*** success rate when trained on 90 assemblies. This is a significant achievement as zero-shot performance on out-of-domain assemblies reaches in-domain performance at this scale. We hypothesize that in-domain and out-of-domain performances will saturate around this success rate as more assemblies become available in the future.

Finetuning a generalist on held-out assemblies. One of the most useful capabilities of a generalist is to serve as a pretrained model that can be efficiently finetuned on problem-specific data. This may especially be true in the context of robotic assembly where assembly-specific data is limited. To further demonstrate the value of our approach, we finetune our 90-assembly generalist to each of 10 held-out assemblies; results are shown in Figure 5. We find that starting from a generalist model greatly improves data-efficiency on new assemblies and, in some cases, exceeds the asymptotic performance of learning from scratch (objects `1036` and `1041`).

Additional experimental results are presented in the appendices, namely Appendix A and Appendix B.

VII. CONCLUSIONS & LIMITATIONS

We present a learning-based system for *zero-shot insertion of unseen objects with unknown geometry* that achieves substantially better scaling than previous work. However, opportunities for improvement remain: *(i)* performance degrades on assemblies with significant visual occlusion, *(ii)* we observe a slight decline in in-domain performance with more training objects, and *(iii)* there is currently limited availability of object datasets for training generalists. We believe that future research in visual perception, further algorithmic improvements, as well as dataset development has immense potential.

Acknowledgements. NH is supported by an NVIDIA Graduate Fellowship. NH and BT completed the work during an internship at NVIDIA.

References

[1] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

[2] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[3] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.

[4] Yunhai Feng, Nicklas Hansen, Ziyan Xiong, Chandramouli Rajagopalan, and Xiaolong Wang. Finetuning offline world models in the real world. *Conference on Robot Learning*, 2023.

[5] Zachary Ferguson, Minchen Li, Teseo Schneider, Francisca Gil-Ureta, Timothy Langlois, Chenfanfu Jiang, Denis Zorin, Danny M. Kaufman, and Daniele Panozzo. Intersection-free rigid body dynamics. *ACM Trans. Graph.*, 40(4), July 2021. ISSN 0730-0301. doi: 10.1145/3450626.3459802. URL https://doi.org/10.1145/3450626.3459802.

[6] Bowen Fu, Sek Kun Leong, Xiaocong Lian, and Xiangyang Ji. 6d robotic assembly based on rgb-only object pose estimation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4736–4742. IEEE, 2022.

[7] Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. Interlinked sph pressure solvers for strong fluid-rigid coupling. *ACM Trans. Graph.*, 38(1), January 2019. ISSN 0730-0301. doi: 10.1145/3284980. URL https://doi.org/10.1145/3284980.

[8] Jean-Bastien Grill, Florian Strub, Florent Altch'e, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 2020.

[9] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, and Yashraj Narang. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984, 2023. doi: 10.1109/ICRA48891.2023.10160216.

[10] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

[11] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[12] Zhenzhong Jia, Ankit Bhatia, Reuben M. Aronson, David Bourne, and Matthew T. Mason. A survey of automated threaded fastening. *IEEE Transactions on Automation Science and Engineering*, 16(1):298–310, 2019. doi: 10.1109/TASE.2018.2835382.

[13] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

[14] Jens Kober, J. Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 09 2013. doi: 10.1177/0278364913495721.

[15] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems*, 2021.

[16] Patrick Lancaster, Nicklas Hansen, Aravind Rajeswaran, and Vikash Kumar. Modem-v2: Visuo-motor world models for real-world robot manipulation. *arXiv preprint*, 2023.

[17] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL https://www.science.org/doi/abs/10.1126/scirobotics.abc5986.

[18] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. URL http://jmlr.org/papers/v17/15-522.html.

[19] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

[20] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[21] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *Robotics: Science and Systems*, 2022.

[22] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/7ec69dd44416c46745f6edd947b470cd-Paper.pdf.

[23] Michael Noseworthy, Bingjie Tang, Bowen Wen, Ankur Handa, Chad Kessens, Nicholas Roy, Dieter Fox, Fabio Ramos, Yashraj Narang, and Iretiayo Akinola. Forge: Force-guided exploration for robust contact-rich manip-

ulation under uncertainty, 2025. URL https://arxiv.org/abs/2408.04587.

[24] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[25] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311.

[26] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning, 2017. URL https://arxiv.org/abs/1710.06542.

[27] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *Conference on Robot Learning (CoRL)*, 2022.

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1998.

[31] Bingjie Tang, Michael A Lin, Iretiayo Akinola, Ankur Handa, Gaurav S Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In *Robotics: Science and Systems*, 2023.

[32] Bingjie Tang, Iretiayo Akinola, Jie Xu, Bowen Wen, Ankur Handa, Karl Van Wyk, Dieter Fox, Gaurav S. Sukhatme, Fabio Ramos, and Yashraj Narang. Automate: Specialist and generalist assembly policies over diverse geometries. In *Robotics: Science and Systems*, 2024.

[33] Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D.D. Willis, and Wojciech Matusik. Assemble them all: Physics-based planning for generalizable assembly by disassembly. *ACM Trans. Graph.*, 41 (6), 2022.

[34] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

[35] Karl D.D. Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Joinable: Learning bottom-up assembly of parametric cad joints. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15828–15839, 2022. doi: 10.1109/CVPR52688.2022.01539.

[36] Jing Xu, Zhimin Hou, Zhi Liu, and Hong Qiao. Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies. *arXiv preprint arXiv:1904.05240*, 2019.

[37] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. Learning visual robotic control efficiently with contrastive pre-training and data augmentation. *arXiv preprint arXiv:2012.07975*, 2020.

[38] Marvin Zhang, Sharad Vikram, Laura Smith, P. Abbeel, Matthew J. Johnson, and Sergey Levine. Solar: Deep structured latent representations for model-based reinforcement learning. *ArXiv*, abs/1808.09105, 2018.

[39] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[40] Íñigo Elguea-Aguinaco, Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Ibai Inziarte-Hidalgo, Simon Bøgh, and Nestor Arana-Arexolaleiba. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, 81:102517, 2023. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2022.102517. URL https://www.sciencedirect.com/science/article/pii/S0736584522001995.

APPENDIX A
ANALYSIS & ABLATIONS

We conduct a series of ablations in both specialist (trained on each of 100 objects) and generalist (90 training objects) and report results in Figure 6. First, we investigate the importance of vision by comparing our approach with *(i)* a "blind" version of our method with access to proprioceptive information only, and *(ii)* an alternative placement of the wrist-mounted camera which provides a better view of the grasped plug but is more prone to visual occlusion. Our results indicate that vision is indeed critical to performance, and our ablation on camera placement corroborates our earlier observation that our approach is prone to failure when the plug is large and (partially) occludes the socket. Our second ablation, shown in Figure 6 (bottom), quantifies the effect of a curriculum that dynamically adapts initial conditions to the current per-assembly success rate. For completeness, we evaluate two such curricula: *(i)* our default curriculum which starts at the easiest initial state distribution and gradually increases difficulty as success rate increases, and *(ii)* an inverse curriculum that starts at the *hardest* setting. Our results indicate that a curriculum consistently improves specialist and generalist performance when evaluated on the full state distribution, while the particular curriculum is less important. We also compare to a privileged version of our generalist that has access to 100 expert rollouts (demonstrations) per assembly; we observe that this improves performance marginally but do no adopt this approach for our proposed generalist training pipeline as it scales poorly with larger object datasets due to the assumption of trained experts.

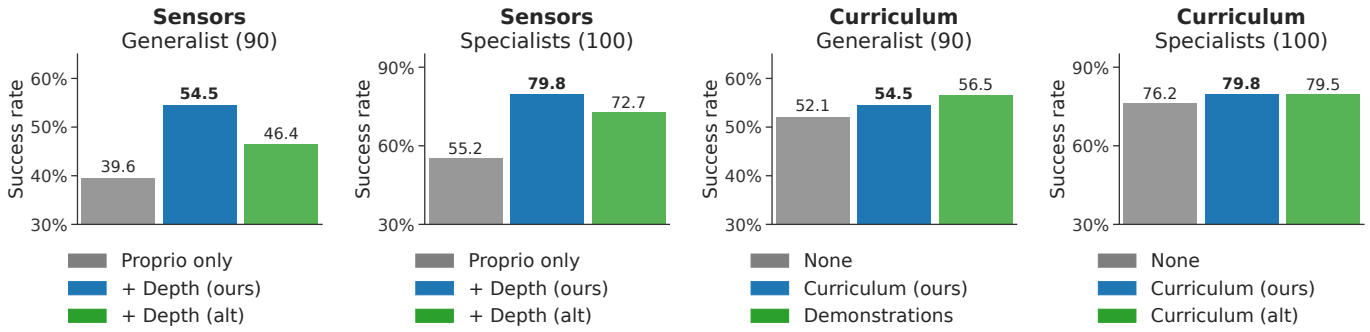

Fig. 6. **Ablations.** Success rate on in-domain objects (generalists trained on 90 objects), or averaged across all single-object specialist policies (trained on each of 100 objects). Our ablations highlight the relative importance of each design choice; **blue** is the default formulation of our method.

APPENDIX B
ADDITIONAL SPECIALIST RESULTS

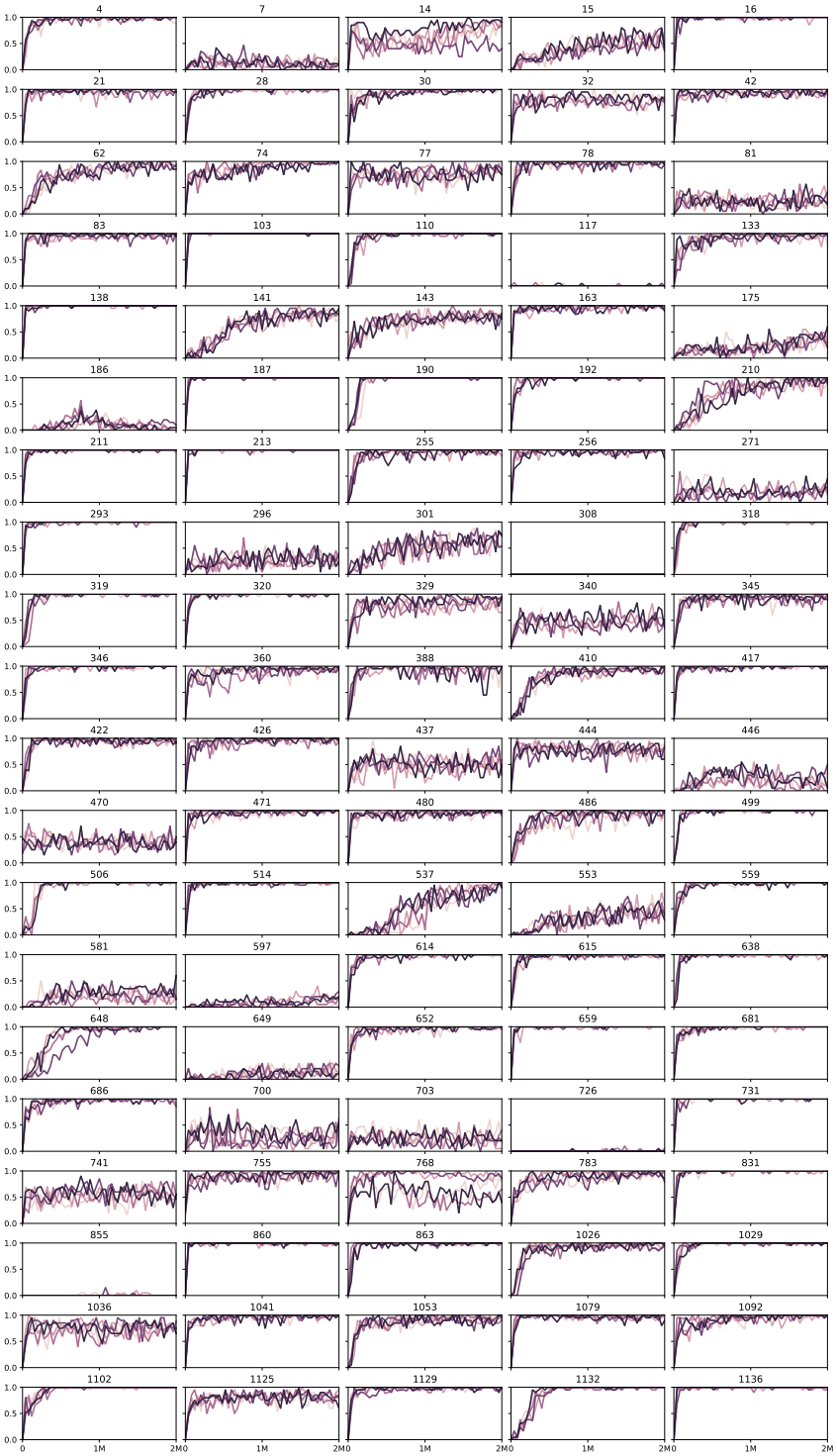

Fig. 7. **Learning curves for 500 specialists trained across 100 assemblies.** Assemblies sorted by unique identifier. Each line corresponds to one of five independent training runs (random seeds).

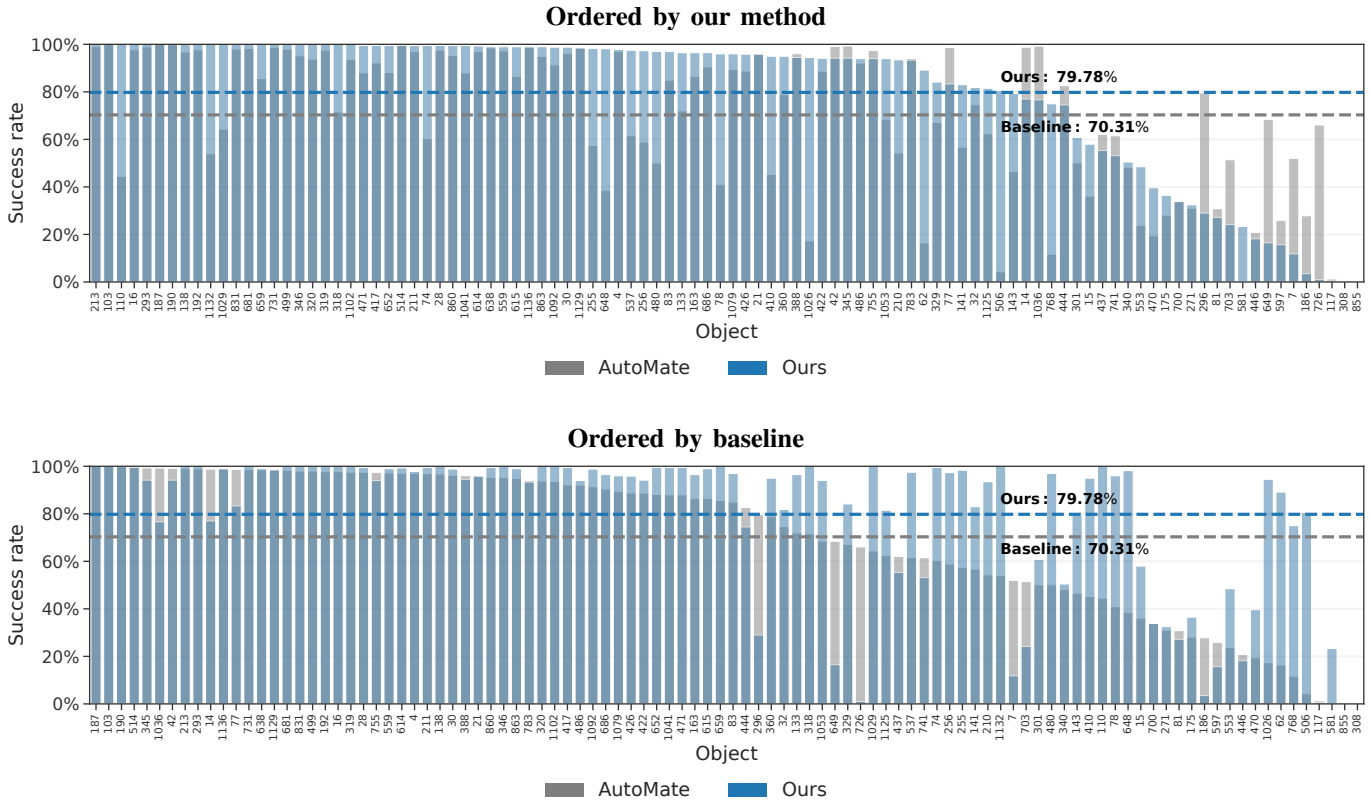

Fig. 8. **Specialist policies.** Success rate of our method (**blue**) and AutoMate (**gray** for each of 100 assemblies, ordered by success rate of our method (*top*) and baseline (*bottom*). Mean of 5 runs per assembly.

APPENDIX C
ADDITIONAL ENVIRONMENT DETAILS

This section supplements Section IV with additional details about our dataset and simulation setup.

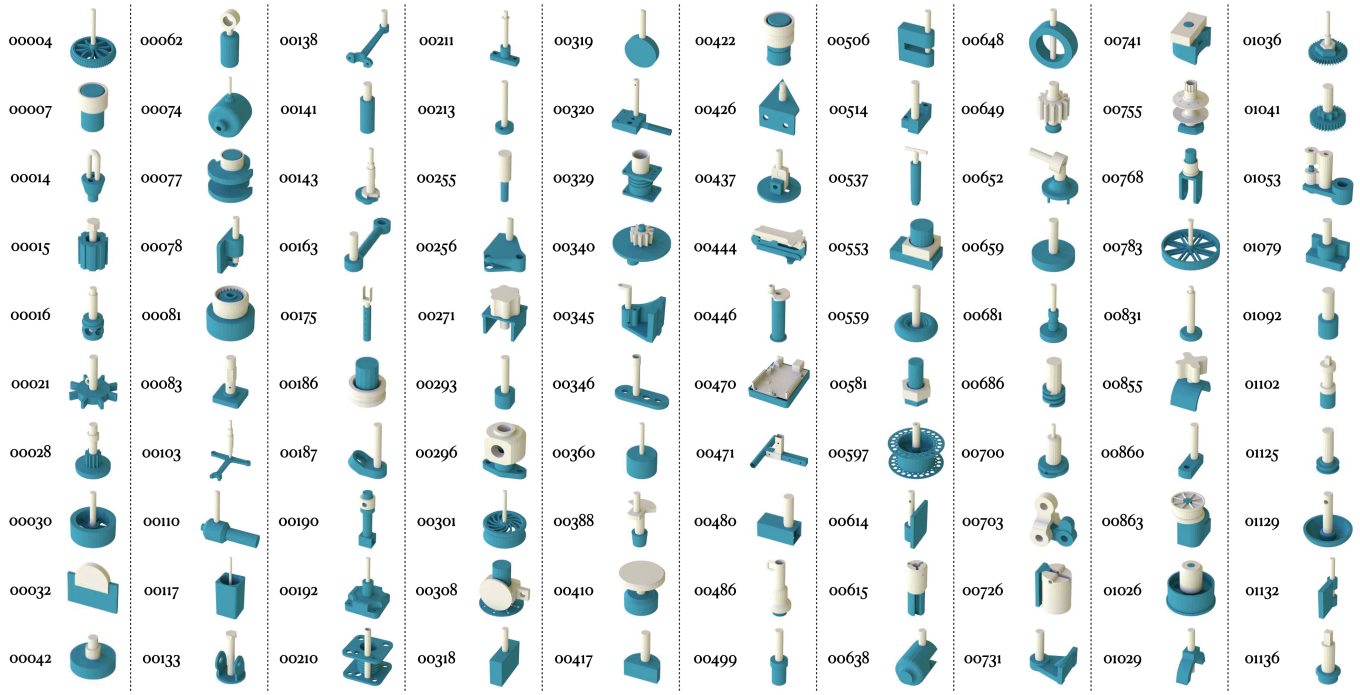

Fig. 9. **Objects sorted by unique identifier.** Overview of all objects used in our work. The first 9 columns are used for training, and the last column is used for out-of-domain evaluation.

Assembly dataset. We leverage the assembly asset dataset from Tang et al. [32], which consists of 100 assemblies that originate from Willis et al. [35], Tian et al. [33] but have been further refined such that their meshes have no interpenetration when fully assembled, making them compatible with common simulators such as Isaac, as well as real-world 3D printers. Despite significant progress in simulation tools in recent years, the process of eliminating interpenetration and other simulation errors remains a tedious human labor-intensive task, and we thus limit ourselves to these 100 object pairs in this work. The assemblies considered in this work all consist of two parts – one part is to be inserted into the other. Objects are initialized in the environment such that the part to be inserted can be inserted in an approximately top-down manner, however, insertion strategy can vary significantly across objects due to their unique and varied geometries. Figure 9 provides an overview of the 100 assemblies that we consider, ordered by their unique identifier consistent with prior work. We reserve 10 (right-most row) of the 100 assemblies for evaluation of out-of-domain generalization ability, and consider the remaining 90 assemblies as our available training set (in-domain).

Reward function. We use a reward function

$$\mathscr{R}(\mathbf{s},\mathbf{a}) \doteq r_{\text{penetration}}(r_{\text{dist}} + r_{\text{imitation}} + r_{\text{success}}) \tag{3}$$

where $r_{\text{penetration}}$ is a scaling penalty for interpenetration errors in the rare event that such simulation errors occur due to *e.g.* very small or thin meshes, r_{dist} is the negative distance between plug and goal, $r_{\text{imitation}}$ is an imitation-based reward that encourages the policy to mimic a small set of demonstrations derived by reversing procedurally-generated disassembly trajectories [25, 32], and r_{success} is a binary term that rewards task success. We omit constant coefficients that balance each term for clarity. This reward function was originally proposed by Tang et al. [31] and later adopted by Tang et al. [32]; we use the reward function without modification and merely provide this description for completeness. Readers are referred to Tang et al. [31] for a detailed discussion of the reward function.

APPENDIX D
VISUALIZATION OF WRIST CAMERA

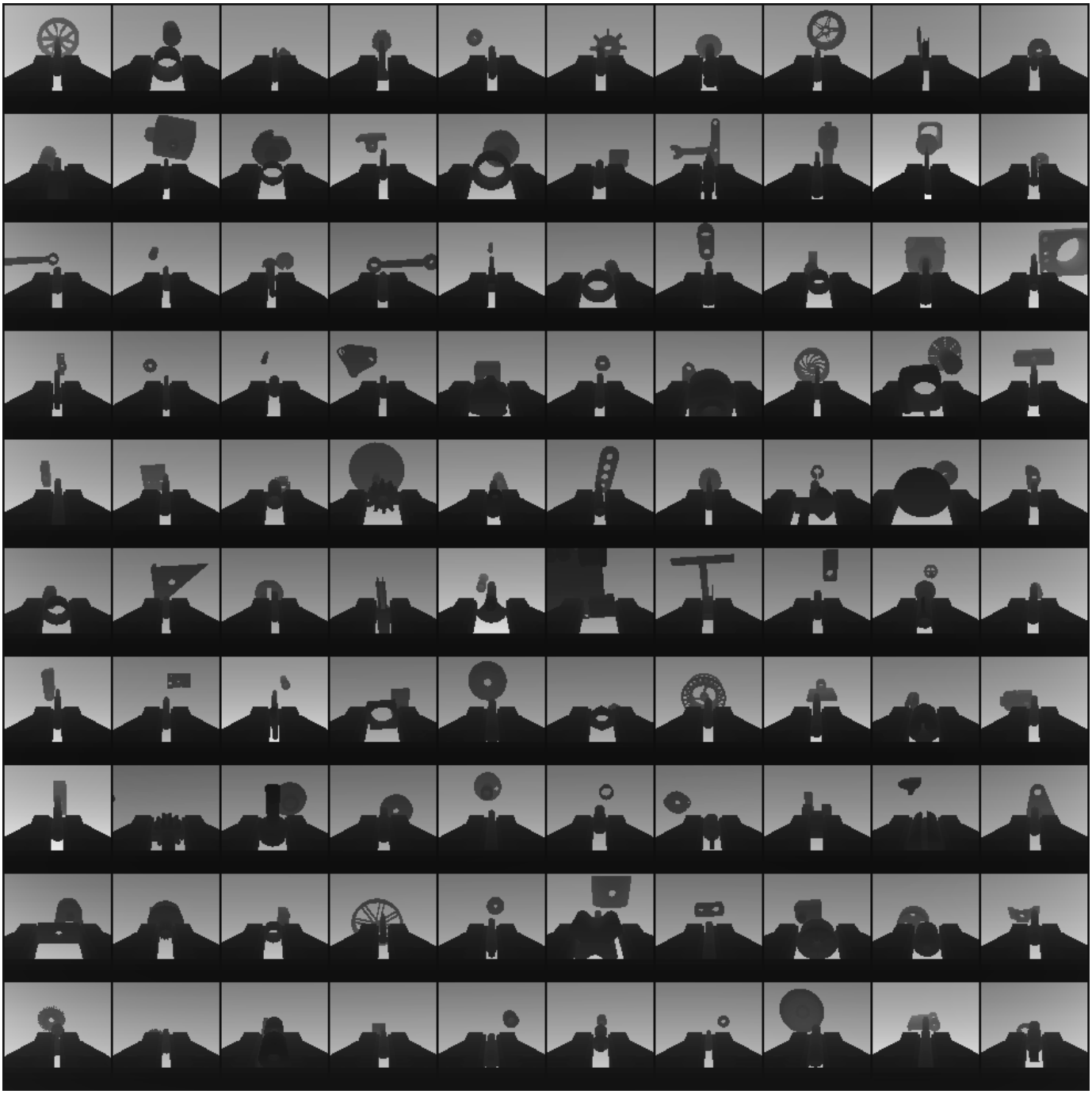

Fig. 10. **Visualization of wrist-mounted camera.** Raw depth observations as provided to our world model for each of the 100 assemblies considered. Observations provide sufficient information for the model to disambiguate objects but is still prone to occlusion when the plug is large.

APPENDIX E
IMPLEMENTATION DETAILS

AutoMate. All AutoMate [32] specialist results are reproduced using the official codebase available at https://github.com/isaac-sim/IsaacGymEnvs/tree/automate without modification and using default hyperparameters. We observe a slightly lower overall success rate than in the original AutoMate work, but remark that we have confirmed the correctness of our results via direct correspondence with the AutoMate authors. They noted, through our correspondence, that the slight discrepancy in results might *"possibly stem from a change in controller or changes in underlying physics"* between submission of the AutoMate manuscript and associated code release. However, we would like to remark that irrespective of any low-level changes that may affect reproducibility, comparison between our world model results and the reproduced AutoMate results remain fair since they were both run with the *same* exact simulator and environment. We obtain generalist AutoMate results (both in-domain and out-of-domain) directly from the authors, which means that baseline generalist numbers are likely to be (slightly) inflated compared to numbers produced by our method. While these reproducibility issues do not change our conclusions in any meaningful way, we provide this additional context to maintain absolute transparency with readers.

TD-MPC2. We base our implementation off of the official TD-MPC2 [11] codebase available at https://github.com/nicklashansen/tdmpc2 and use default hyperparameters throughout our experiments. However, we list all hyperparameters in Table I for completeness. We summarize our modified architecture (to support multi-modal depth and proprioceptive inputs) using PyTorch-like notation:

```
Architecture: WorldModel(
  (encoder): ModuleDict(
    (depth): Sequential(
      (0): ShiftAug(pad=4)
      (2): Conv2d(1, 32, kernel_size=(7, 7), stride=(2, 2))
      (3): ReLU(inplace=True)
      (4): Conv2d(32, 32, kernel_size=(5, 5), stride=(2, 2))
      (5): ReLU(inplace=True)
      (6): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2))
      (7): ReLU(inplace=True)
      (8): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2))
      (9): ReLU(inplace=True)
      (10): Flatten(start_dim=1, end_dim=-1)
      (11): Linear(in_features=512, out_features=Z, bias=True)
      (12): SimNorm(dim=8))
    (state): Sequential(
      (0): NormedLinear(in_features=S, out_features=256, bias=True, act=Mish)
      (1): NormedLinear(in_features=256, out_features=Z, bias=True, act=SimNorm))
    (fuse): Sequential(
      (0): NormedLinear(in_features=Z, out_features=256, bias=True, act=Mish)
      (1): NormedLinear(in_features=256, out_features=Z, bias=True, act=SimNorm))
    (dynamics): Sequential(
      (0): NormedLinear(in_features=Z+A, out_features=Z, bias=True, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
      (2): Linear(in_features=512, out_features=Z, bias=True))
    (reward): Sequential(
      (0): NormedLinear(in_features=Z+A, out_features=512, bias=True, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
      (2): Linear(in_features=512, out_features=101, bias=True))
    (pi): Sequential(
      (0): NormedLinear(in_features=Z, out_features=512, bias=True, act=Mish)
      (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
      (2): Linear(in_features=512, out_features=2A, bias=True))
    (Qs): Vectorized ModuleList(
      (0-4): 5 x Sequential(
        (0): NormedLinear(in_features=Z+A, out_features=512, bias=True, act=Mish)
        (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
        (2): Linear(in_features=512, out_features=101, bias=True))))
```

where `S` is the input dimensionality, `Z` is the latent state dimension (512), and `A` is the action space dimensionality. The total number of learnable parameters is 5M for both specialists and generalists.

TABLE I. **Hyperparameters.** We use the same hyperparameters across all objects, and for both specialists and generalists. Our hyperparameters are adopted from the official implementation of TD-MPC2 without modification.

Hyperparameter	Value
Planning	
Horizon (H)	3
Iterations	8
Population size	512
Policy prior samples	24
Number of elites	64
Temperature	0.5
Policy prior	
Log std. min.	-10
Log std. max.	2
Replay buffer	
Capacity	$1,000,000$
Sampling	Uniform
Architecture	
Encoder dim	256
MLP dim	512
Latent state dim	512
Activation	LayerNorm + Mish
Number of Q-functions	5
Optimization	
Update-to-data ratio	1
Batch size	256
Joint-embedding coef.	20
Reward prediction coef.	0.1
Value prediction coef.	0.1
Temporal coef. (λ)	0.5
Q-fn. momentum coef.	0.99
Policy prior entropy coef.	1×10^{-4}
Policy prior loss norm.	Moving $(5\%, 95\%)$ percentiles
Optimizer	Adam
Learning rate	3×10^{-4}
Encoder learning rate	1×10^{-4}
Gradient clip norm	20
Discount factor	0.9
Seed steps	$5,000$