FINGR: LEARNING DEXTEROUS HAND CONTROL FOR REAL-WORLD RUBIK'S CUBE SOLVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Manipulating a Rubik's Cube with one single dexterous hand is a challenging test of sustained, contact-rich control: the hand must execute successive layer turns while keeping the cube secure. Each turn requires some fingers to support the cube while others push a moving layer, release contact, and reset for the next move. To learn this coordination, we introduce **FINGR** (**F**uture-supervised **I**nteraction **N**etwork with **G**eometric **R**epresentations), a policy that combines finger-relative geometry with future interaction prediction. A shared point encoder expresses the cube relative to each fingertip and aggregates its points without depending on cubie indexing. Learned future tokens share the observation encoder and receive supervision for contact-force changes, layer-turn progress, and finger joint displacement at multiple time scales. The resulting representation conditions a flow policy that directly generates finger actions. On a real dexterous hand, our policy achieves around 99.0% success over 300 turn attempts, compared with 79.7% for the base flow policy. Integrated with grasping and table-assisted regrasping, the policy solves all ten scrambled $2 \times 2 \times 2$ cubes in a mean complete-system time of approximately 137 seconds.

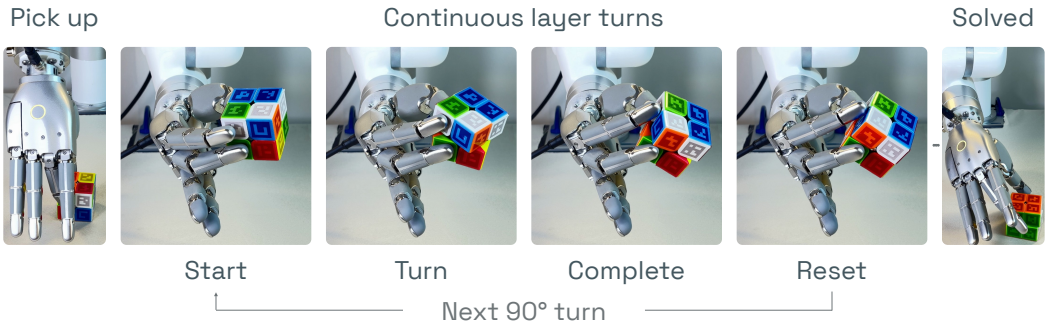


Figure 1: **Real cube solving through continuous layer turns.** The four central frames show one 90° layer turn, from the starting grasp through finger reset. Repeating these learned motions, together with grasping and table-assisted regrasping, completes cube solving.

1 INTRODUCTION

Human cube solvers use finger tricks to turn a layer with a short finger motion, recover the finger, and immediately continue with the next move. Reproducing this behavior with a single hand requires coordination across fingers with different roles. Some fingers must support the cube, while others establish contact with the moving layer, push it through a quarter turn, and recover a configuration suitable for the next action. A successful isolated push is therefore only one part of the skill: the resulting hand–cube configuration must also support continued manipulation.

This coordination has both spatial and temporal structure. Spatially, a finger's available action depends on where the moving layer and supporting surfaces lie relative to it. Encoding the cube globally and each finger separately leaves the policy to infer these relationships. Temporally, contact

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

changes during a turn: a finger loads a surface, advances the layer, releases contact, and moves back. Action imitation alone does not explicitly train the observation encoder to capture how contact and motion will evolve.

We address these two aspects with **FINGR**. First, we translate the cube's geometric points to origins at the five fingertips. A finger-relative cube position encoder and symmetric aggregation produce a geometric feature for each finger, which is added directly to its state token. The representation preserves information about relative directions and distances while remaining invariant to the order of cube points. Second, we introduce learned future tokens at three horizons. These tokens interact with the observation tokens in a transformer encoder and are trained to predict contact-force changes, target-layer progress, and finger joint displacement. The predictive targets connect the representation to the physical events needed for turning and recovery.

The two components are trained with a flow policy that outputs joint-action chunks. Future-token prediction provides auxiliary supervision during learning, while the encoded tokens directly condition action generation at deployment. We implement the policy on a seven-joint arm with a dexterous hand. The thumb and middle finger retain the cube, and the index, ring, and little fingers execute the learned motion. A discrete solver composes upper- and left-layer turns with a planned table-assisted regrasp when a new grasp is required to complete the solution.

Our contributions are a real robotic cube-solving system based on continuous layer turns; a permutation-invariant geometric encoding aligned with individual fingertips; and multi-horizon future-token prediction that jointly supervises contact, layer progress, and finger motion. Across 300 physical layer-turn attempts per method, local geometry increases success from 79.7% to 92.7%, and future-token prediction further increases it to 99.0%. The complete system solves ten randomly scrambled cubes in an average of approximately 137 seconds, including pickup, regrasping, placement, and return to the starting robot posture.

2 RELATED WORK

Dexterous manipulation of articulated objects. Prior work on in-hand manipulation has learned object rotation and reorientation using proprioceptive, tactile, and visual feedback (Qi et al., 2022; Yin et al., 2023; Qi et al., 2023). Subsequent work extends rotation to slender objects and composes rotation skills for goal-directed reorientation (Wang et al., 2024; Qi et al., 2025; Yin et al., 2025; Kedia et al., 2026). Articulated objects additionally require control of internal configurations. Rubik's cube systems combine symbolic planning with learned manipulation to execute multi-step solutions in simulation and on physical hands (Li et al., 2019; OpenAI et al., 2019). These capabilities benefit from scalable simulation and robot-learning platforms (Makoviychuk et al., 2021; Geng et al., 2025; Jiang et al., 2025; Tao et al., 2025), alongside datasets and capture systems that describe hand pose, object motion, and contact (Taheri et al., 2020; Liang et al., 2026b; Fan et al., 2023; Liang et al., 2026a; Brahmbhatt et al., 2020). In particular, articulated interaction data expose the coupled evolution of hand motion and object configuration. We learn continuous layer turns from real demonstrations, organizing the policy around each finger's access to the moving layer and its recovery for the next turn. Integrating this skill with pickup and table-assisted regrasping enables complete physical cube solving.

Hand–object representations. Hand–object interaction representations connect the kinematics of a hand to the geometry it manipulates. MANO provides a compact model of articulated hand shape and pose (Romero et al., 2017), while point-set encoders aggregate object geometry without depending on point ordering (Qi et al., 2017). Representing their relationship explicitly can further expose feasible contacts and motions. ManipNet combines global object occupancy with local distances in a hand-centric coordinate system to synthesize finger motion (Zhang et al., 2021). Contact-based grasp refinement and hand–surface encodings similarly use geometric relationships to guide grasping and dexterous control (Grady et al., 2021; Liu et al., 2023; 2026). Representations based on contact, shared action spaces, and robot–object distances also support transfer across different hand embodiments (Li et al., 2023a; Jiang et al., 2026; Wei et al., 2025). These approaches motivate expressing geometry in coordinates relevant to the acting hand. For continuous cube turns, the relevant relationships change separately for each finger as the layer advances. We therefore encode directional offsets from every fingertip to the reconstructed cube points, pool over the point set, and attach each feature to its

corresponding finger token. This preserves cubie-index invariance while supplying the policy with finger-specific spatial information throughout the turn.

Prediction for action learning. Prediction links an agent's current observations to the outcomes of its actions. Latent world models use learned dynamics for online planning or behavior learning through imagined trajectories (Hafner et al., 2019; 2020; Hansen et al., 2024). In imitation learning, generative policies produce coordinated action sequences using latent-variable models, diffusion, and flow matching (Zhao et al., 2023; Chi et al., 2023; Lipman et al., 2022); reconstruction and future-observation objectives can enrich the representations used for action prediction (Li et al., 2023b; Wu et al., 2023; Tian et al., 2024; Peng et al., 2026b). Recent world-action models bring these directions together by coupling action generation with predictions of the evolving visual world (Zhu et al., 2025; Cen et al., 2025; Peng et al., 2026a; Ye et al., 2026). Across these approaches, future supervision encourages features that capture how interactions unfold. For repeated layer turns, useful future information includes whether contact is changing, whether the target layer is advancing, and how the fingers are moving toward their next configuration. Our future tokens jointly predict these three quantities at multiple horizons, linking immediate contact changes to longer-term turn progress and finger motion. The prediction objective trains the shared observation representation, which directly conditions the action head during deployment.

3 METHOD

3.1 TASK AND POLICY INTERFACE

We consider a $2 \times 2 \times 2$ cube held by a dexterous robot hand. The learned skill performs 90° turns of two layers, denoted U and L in the current grasp. A discrete solver selects these moves and decides when to place and regrasp the cube. During a learned turn, the arm maintains its posture and the thumb and middle finger maintain their holding configuration; the policy commands the 13 joints of the index, ring, and little fingers.

At control time t, the observation contains numeric features s_t^i for each of the five fingers, tactile deformation images for the three controlled fingers, cube points $P_t = \{p_{t,j}\}_{j=1}^{32}$, a move label $m_t \in \{U, L\}$, and a normalized remaining angle r_t. Each s_t^i contains joint positions, velocities and torques, six tactile force/torque channels, and the fingertip position f_t^i. Cube points and fingertip positions share the palm coordinate frame. The points describe the current articulated geometry through 24 face-center samples and eight outer-corner samples reconstructed from the cubie poses. The policy receives their coordinates without cubie identity or color labels.

The policy predicts an action chunk of $H = 20$ joint-position offsets relative to the current controlled joint configuration $q_t \in \mathbb{R}^{13}$. Observations and commands run at 10 Hz. We execute at least five steps before updating the chunk and align commands to the observation timestamp to compensate for inference latency. All evaluated action-learning methods use this interface.

3.2 FINGER-RELATIVE GEOMETRY

A global point-set feature summarizes the cube but does not explicitly associate its surfaces with a particular finger. We construct that association before the observation transformer. For finger i and cube point j, define

$$d_{t,ij} = \frac{p_{t,j} - f_t^i}{\ell}, \qquad g_t^i = W_g \left(\frac{1}{32} \sum_{j=1}^{32} \phi(d_{t,ij}) \right) + b_g, \tag{1}$$

where $\ell = 0.05$ m is a fixed length scale, $\phi : \mathbb{R}^3 \to \mathbb{R}^{64}$ is a shared two-layer point network, and W_g, b_g project its pooled output to the transformer width. The origin moves with the fingertip; the axes remain aligned with the palm. Thus, $d_{t,ij}$ retains the direction and distance from a specific finger to a cube surface point.

We add g_t^i to the projected numeric feature of finger i. Finger identity embeddings distinguish the five fingers, and modality embeddings distinguish numeric, tactile, and global cube information. A shared convolutional network encodes the three tactile images, with finger-specific feature-wise scale

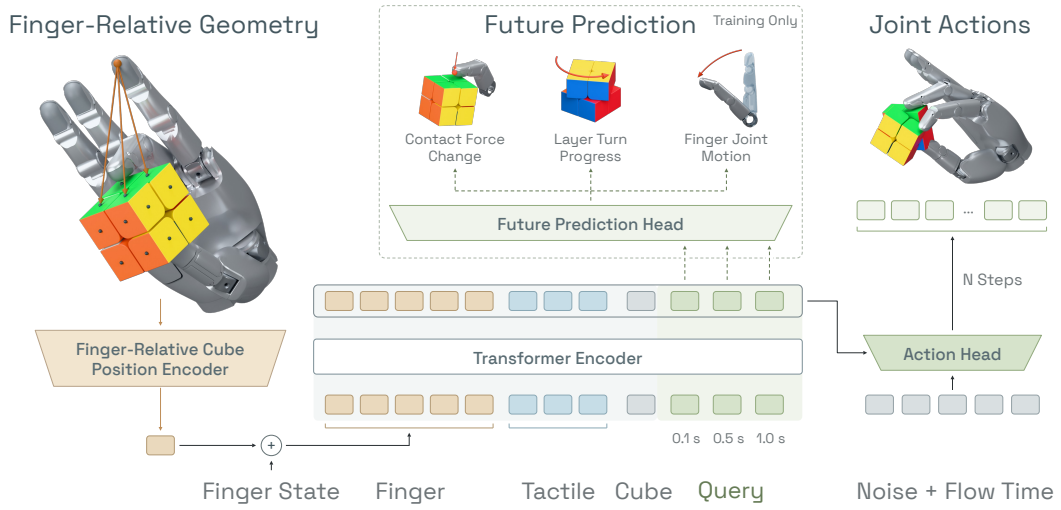


Figure 2: **Policy architecture.** FINGR uses a finger-relative cube position encoder to augment the five finger-state tokens. A transformer encoder processes these tokens together with tactile inputs, global cube geometry, and three future queries. At each future horizon, the future prediction head jointly predicts contact-force change, turn progress, and joint displacement during training. The full encoded memory, including the three future tokens, conditions the action head at deployment.

and shift parameters. A separate point network with max pooling provides one global cube token. The task embedding and a projection of r_t are added to all nine observation tokens.

The mean in Equation 1 is unchanged by any permutation of the cube points, so the finger feature does not depend on cubie indexing. At the same time, translating the points to each fingertip makes the relative geometry explicit within that finger's token. The transformer can use these tokens to coordinate the fingers, while the point network already exposes the local spatial relation needed to select a push, maintain support, or recover from contact.

3.3 FUTURE INTERACTION PREDICTION

Finger-relative geometry describes where each finger can interact with the cube. To connect this geometry to the unfolding turn, we train future tokens to predict how contact, layer progress, and finger motion change together. Each learned query z_h corresponds to an offset $h \in \mathcal{H} = \{1, 5, 10\}$ control steps. The transformer jointly encodes the current observation and these three queries:

$$M_t = E_\theta\big([O_t, z_1, z_5, z_{10}]\big), \qquad \widehat{y}_{t,h} = D_\psi(M_t^{(h)}), \tag{2}$$

where O_t denotes the nine observation tokens, E_θ is the transformer encoder, $M_t^{(h)}$ is the encoded future token for horizon h, and D_ψ is the future prediction head shared across horizons.

Let $c_t \in \mathbb{R}^9$ concatenate the three-axis contact-force measurements of the three controlled fingers. We supervise each query with

$$y_{t,h} = [\, c_{t+h} - c_t, \quad r_t - r_{t+h}, \quad q_{t+h} - q_t \,] \in \mathbb{R}^{23}. \tag{3}$$

The targets describe three coupled aspects of a turn: contact loading and unloading, progress toward the target layer angle, and measured joint displacement during pushing and recovery. Supervising them together requires each future token to relate object progress to the contact and finger motion that accompany it. Short horizons emphasize immediate changes; longer horizons cover more of the turn and reset sequence.

Targets are scaled by their empirical standard deviations separately for each horizon and coordinate. A target is valid only when its current and future frames lie in the same continuously tracked turn

segment. We average squared errors within each target group and then average the three groups:

$$\mathcal{L}_{\text{pred}} = \frac{1}{3|\mathcal{V}|} \sum_{(t,h)\in\mathcal{V}} \left(\frac{\|e^c_{t,h}\|_2^2}{9} + |e^r_{t,h}|^2 + \frac{\|e^q_{t,h}\|_2^2}{13} \right), \tag{4}$$

where $\mathcal{V}$ is the set of valid current-frame/horizon pairs and e^c, e^r, e^q are errors in the standardized targets. This averaging gives contact, progress, and motion equal weight despite their different dimensions.

Action learning imitates successful, well-aligned turns. Future prediction also uses continuous interaction segments excluded from action imitation, extending the supervision of contact and motion dynamics. Both objectives update the shared observation encoder. At deployment, the encoder constructs M_t from the current observation and learned queries, and the action head attends to all encoded tokens. The auxiliary prediction head is omitted; the future tokens carry the learned predictive features directly into action generation.

3.4 Action Learning and Execution

We use a flow policy with an action denoising parameterization. Let $A_t \in \mathbb{R}^{H\times 13}$ be the normalized demonstrated action chunk and $\epsilon \sim \mathcal{N}(0, I)$ a noise chunk. For noise level $\tau \in (0, 1]$, the training input is

$$X_\tau = (1-\tau)A_t + \tau\epsilon, \qquad v_\theta(X_\tau, \tau, M_t) = \frac{X_\tau - \widehat{A}_\theta(X_\tau, \tau, M_t)}{\tau}. \tag{5}$$

The action head alternates action self-attention, cross-attention to M_t, and feed-forward layers. Noise-level conditioning modulates each block. We train the clean-action estimate $\widehat{A}_\theta$ with masked joint-space reconstruction error; Appendix A.3 gives the exact normalization. At inference, four Euler steps integrate Equation 5 from noise at $\tau = 1$ to the action chunk at $\tau = 0$.

The combined training objective is

$$\mathcal{L}(k) = \mathcal{L}_{\text{act}} + \lambda(k)\mathcal{L}_{\text{pred}}, \qquad \lambda(k) = 0.003 \max\left(0, 1 - \frac{k}{0.5K}\right), \tag{6}$$

where k is the optimizer step and K is the total number of training steps. Predictive supervision shapes the representation during the first half of training, after which the action objective continues to optimize the same transformer encoder and action head. Deployment uses exponential-moving-average model parameters.

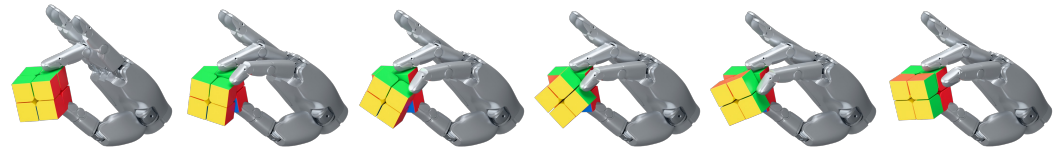
3.5 Complete Cube-Solving System

The system first reconstructs the cube pose, plans a pregrasp, closes the supporting fingers, and lifts the cube into the manipulation pose. Multi-view tracking of coded surface markers estimates cubie poses. After pickup, the system observes the discrete cube state and computes a sequence of U/L moves. The solver minimizes the number of positive quarter turns and then the cost of repeated turns of the same layer.

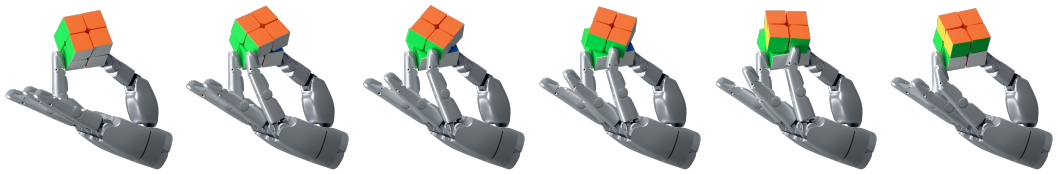
Some scrambles require changing the supporting grasp. The planner then first creates a compatible solved two-cubie block, places the cube on the table in a selected orientation, and grasps that block before completing a second U/L sequence. Placement and regrasping use kinematic planning and collision checks. The system verifies the cube after a sequence, performs an additional alignment motion when needed, and replans from the observed state. Once all cubies have a common solved orientation, the robot places the cube on the table and returns to its starting posture. This connects the learned continuous manipulation skill to complete solutions of scrambled cubes.

4 Experiments

We evaluate whether the proposed representation improves physical layer turning and whether this skill supports complete cube solving. The first experiment compares action-learning methods under a common observation and deployment interface. The second isolates local geometry and future-token prediction. The final experiment evaluates the complete system on ten random scrambles.



(a) U turn



(b) L turn

Figure 3: **Layer-turn demonstrations.** Each row shows six hand–cube configurations at approximately 18° increments across one 90° turn.

4.1 Experimental Setup

Hardware and demonstrations. The platform uses an xArm7 and a 22-joint Sharpa hand. The cube has a nominal edge length of 52 mm and coded markers for reconstruction. Our demonstration set comprises 36 teleoperated sequences with 576 turns. Action training uses 435 successful turns of at most eight seconds: 226 U turns and 209 L turns, totaling 18k frames. Figure 3 visualizes representative U and L demonstrations. All methods use this action-training set. Future prediction uses 25k frames with valid future targets from the same demonstration set. Appendix A.1 specifies the sample selection criteria.

Baselines. We compare ACT (Zhao et al., 2023), Diffusion Policy (Chi et al., 2023), and a Base Flow Policy. ACT uses a conditional variational action decoder, and Diffusion Policy uses a transformer noise predictor with ten inference steps. Both use the same numeric, tactile, and global cube observations as the Base Flow Policy. The base flow model has the same action decoder as our method and a global point-set encoding. The ablation adds finger-relative geometry to this model, followed by the complete future-token prediction component. All models use a hidden width of 384, four observation-encoder layers, four action-decoder layers, a 20-step action horizon, and 1,000 training epochs. Detailed method-specific settings appear in Appendix A.

Layer-turn protocol. Each method is evaluated for three deployment rounds using one fixed checkpoint and deployment seeds 0, 1, and 2. Each round contains 100 attempts: 50 U and 50 L, shuffled by the seed. A turn succeeds when the estimated target-angle error remains below 15° for three observation frames. Each attempt has a ten-second timeout. A drop is detected when the cube moves more than 10 cm from its reference position in palm coordinates for three frames. After a timeout, the benchmark proceeds to the next scheduled attempt; dropped cubes are returned before continuation. We report the mean and sample standard deviation across the three rounds.

For N attempts, mean time per attempt is

$$T_{\text{attempt}} = \frac{1}{N} \sum_{n=1}^{N} t_n, \tag{7}$$

where t_n is the elapsed time from policy execution start to success, timeout, or drop. Reset intervals between attempts are excluded. Appendix B gives the results for each round.

4.2 Comparison with Action-Learning Baselines

Table 1 shows that FINGR achieves $99.0 \pm 1.0\%$ overall success. ACT, Diffusion Policy, and Base Flow achieve 77.7%, 78.7%, and 79.7%, respectively. The improvement over Base Flow is 19.3

Table 1: **Real layer-turn comparison.** Success, timeout, and drop entries are percentages. Time is the mean duration per attempt in seconds. Values are mean ± sample standard deviation over three rounds of 100 attempts. Bold marks column-wise best values, including ties.

Method	U success ↑	L success ↑	Timeout ↓	Drop ↓	All success ↑	Time ↓
ACT	72.0 ± 8.7	83.3 ± 6.4	22.3 ± 6.5	$\mathbf{0.0 \pm 0.0}$	77.7 ± 6.5	6.39 ± 0.62
Diffusion Policy	78.0 ± 11.1	79.3 ± 13.0	20.3 ± 6.7	1.0 ± 0.0	78.7 ± 6.7	6.06 ± 0.14
Base Flow Policy	82.7 ± 13.0	76.7 ± 3.1	19.7 ± 7.5	0.7 ± 0.6	79.7 ± 8.0	5.67 ± 0.14
FINGR	$\mathbf{99.3 \pm 1.2}$	$\mathbf{98.7 \pm 1.2}$	$\mathbf{1.0 \pm 1.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{99.0 \pm 1.0}$	$\mathbf{5.25 \pm 0.20}$

Table 2: **Representation ablation.** Local Geometry adds the finger-relative encoding to Base Flow. FINGR additionally includes future tokens with the joint contact, progress, and motion prediction objective.

Method	U success ↑	L success ↑	Timeout ↓	Drop ↓	All success ↑	Time ↓
Base Flow Policy	82.7 ± 13.0	76.7 ± 3.1	19.7 ± 7.5	0.7 ± 0.6	79.7 ± 8.0	5.67 ± 0.14
Local Geometry	92.7 ± 5.8	92.7 ± 1.2	7.0 ± 1.7	0.3 ± 0.6	92.7 ± 2.3	$\mathbf{5.24 \pm 0.04}$
FINGR	$\mathbf{99.3 \pm 1.2}$	$\mathbf{98.7 \pm 1.2}$	$\mathbf{1.0 \pm 1.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{99.0 \pm 1.0}$	5.25 ± 0.20

percentage points. It appears in both directions: U success rises from 82.7% to 99.3%, and L success rises from 76.7% to 98.7%. The resulting policy supports consecutive turns of either layer under a common grasp.

The improvement comes mainly from fewer timeouts: three across 300 attempts, compared with 59 for Base Flow. Drops decrease from two to zero. Mean time per attempt is 5.25 ± 0.20 seconds, compared with 5.67 ± 0.14 for Base Flow, 6.06 ± 0.14 for Diffusion Policy, and 6.39 ± 0.62 for ACT.

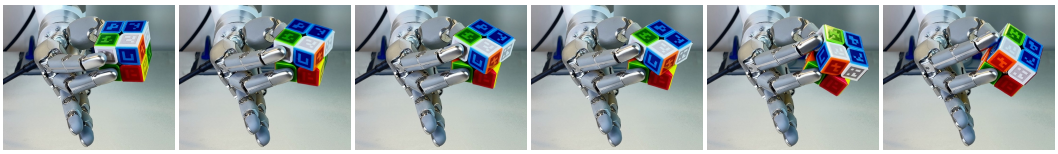
4.3 REPRESENTATION ABLATION

Finger-relative geometry raises success to $92.7 \pm 2.3\%$, a gain of 13.0 percentage points over Base Flow (Table 2). Both directions reach 92.7%, while the timeout rate falls from 19.7% to 7.0%. The geometry module uses the same reconstructed point set as the base policy and adds only its relation to each fingertip. These gains show that expressing cube geometry relative to each fingertip helps the policy complete turns more reliably.

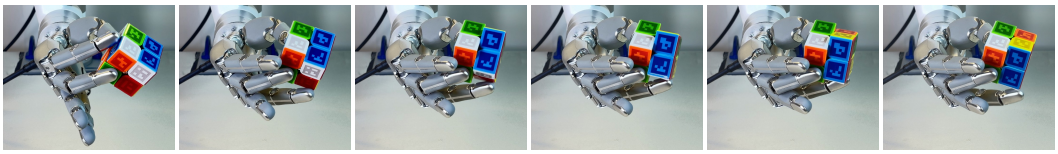
Future-token prediction adds another 6.3 percentage points, increasing successful attempts from 278 to 297 out of 300. Timeouts fall from 21 to three, and drops from one to zero. Mean time per attempt remains close: 5.25 seconds for the full policy and 5.24 seconds for local geometry. Future prediction improves completion at a similar time cost by supplementing instantaneous geometry with supervision on layer progress, contact changes, and finger motion.

4.4 SOLVING REAL SCRAMBLED CUBES

We evaluate ten random 11-move scrambles using R, F, and U quarter, half, and inverse turns (Appendix C). The solver uses the same layer-turn policy as the isolated benchmark. The complete solver uses an 18° completion threshold for three observation frames and verifies the discrete state after each sequence.



(a) U turn



(b) L turn

Figure 4: **Real layer turns.** Six stages of each $90°$ turn show layer advancement and finger reset while the supporting fingers retain the cube.

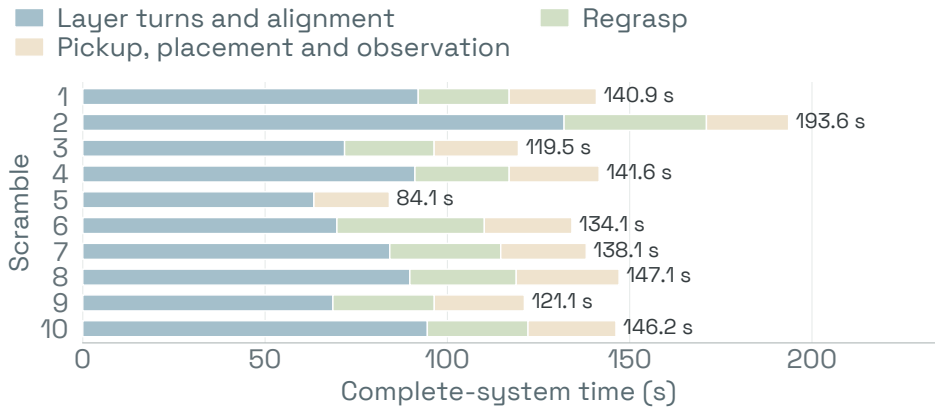


Figure 5: **Time spent in complete cube solving.** Layer turns dominate execution; nine solves include a regrasp. Bars cover the full system time, detailed in Table 6.

Table 3: **Complete physical cube solves.** Counts and times cover planned $90°$ attempts. Plan includes motion planning and checks; Total spans initial grasp planning through final placement and robot return. Times are seconds.

Scramble	#U	#L	U time ↓	L time ↓	Time/$90°$ ↓	Regrasps	Plan ↓	Total ↓
1	8	10	6.30	3.98	5.01	1	1.56	140.87
2	9	8	9.92	5.34	7.76	1	1.97	193.59
3	7	5	5.69	6.26	5.93	1	1.55	119.48
4	10	9	5.28	4.26	4.79	1	1.83	141.62
5	8	9	3.35	4.07	3.73	0	**0.50**	**84.08**
6	8	8	4.50	4.21	4.36	1	3.95	134.14
7	8	12	4.29	4.16	4.21	1	3.10	138.09
8	11	9	4.88	3.74	4.37	1	2.92	147.09
9	13	7	**3.18**	**3.43**	**3.27**	1	2.90	121.12
10	12	14	3.64	3.63	3.63	1	3.23	146.23

The system solves all ten cubes, achieving $10/10$ complete-solve success (Table 3). Nine executions use one table-assisted regrasp; one completes under the initial grasp. Figure 4 shows physical U and L turns, and Figure 6 shows a complete execution. Complete-system time averages 136.63 ± 27.47 seconds, ranging from 84.08 to 193.59 seconds. Solutions require 12–26 planned quarter-turn attempts, with a per-solve average of 4.71 ± 1.32 seconds per quarter turn.

Across these solves, 181 of 185 planned turns succeed. Reobservation, alignment, and replanning recover from state mismatches to complete every solve. All four alignment corrections, ten initial pickups, nine regrasps, and ten final placements and returns succeed. State verification and recovery therefore allow the system to complete full solutions even when individual layer turns fail. Figure 5 and Appendix C break down its execution time.



Figure 6: **From pickup to a solved cube.** Left to right, top to bottom. Each pair of turn rows shows the first two 90° rotations after pickup or regrasp (L then U); later turns are omitted.

5 Discussion and Conclusion

Continuous layer turns require a policy to coordinate each finger's spatial access to the target layer with changing contact and motion. Finger-relative geometry produces the largest improvement over the base policy, and adding future-token prediction further improves completion across both layers. Together, these components support repeated layer turns and integrate with pickup and table-assisted regrasping to solve scrambled cubes on real hardware.

The present evaluation uses one hand and a tagged $2 \times 2 \times 2$ cube, with marker-based reconstruction and planned pickup and placement. Extending the representation to unmarked articulated objects and other hands would test how broadly its spatial and temporal structure transfers. The complete solves demonstrate the integrated capability; a larger set of scrambles and starting grasps would characterize its operating range.

AI USE STATEMENT

Generative AI assisted with literature retrieval, analysis code, figure preparation, and manuscript drafting and editing.

REPRODUCIBILITY STATEMENT

Appendix A specifies the observations, architecture, training objectives, and deployment settings. Appendix B reports all evaluation rounds and timing definitions, and Appendix C gives the complete scramble and solution sequences and system timing breakdown.

REFERENCES

Samarth Brahmbhatt, Chengcheng Tang, Christopher D. Twigg, Charles C. Kemp, and James Hays. ContactPose: A Dataset of Grasps with Object Contact and Hand Pose. In *European Conference on Computer Vision*, 2020.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. WorldVLA: Towards Autoregressive Action World Model. *arXiv preprint arXiv:2506.21539*, 2025. URL https://arxiv.org/abs/2506.21539.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin CM Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J. Black, and Otmar Hilliges. ARCTIC: A Dataset for Dexterous Bimanual Hand-Object Manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Haoran Geng, Feishi Wang, Songlin Wei, Yuyang Li, Bangjun Wang, Boshi An, Charlie Tianyue Cheng, Haozhe Lou, Peihao Li, Yen-Jen Wang, Yutong Liang, Dylan Goetting, Chaoyi Xu, Haozhe Chen, Yuxi Qian, Yiran Geng, Jiageng Mao, Weikang Wan, Mingtong Zhang, Jiangran Lyu, Siheng Zhao, Jiazhao Zhang, Jialiang Zhang, Chengyang Zhao, Haoran Lu, Yufei Ding, Ran Gong, Yuran Wang, Yuxuan Kuang, Ruihai Wu, Baoxiong Jia, Carlo Sferrazza, Hao Dong, Siyuan Huang, Yue Wang, Jitendra Malik, and Pieter Abbeel. RoboVerse: Towards a Unified Platform, Dataset and Benchmark for Scalable and Generalizable Robot Learning. *arXiv preprint arXiv:2504.18904*, 2025. URL https://arxiv.org/abs/2504.18904.

Patrick Grady, Chengcheng Tang, Christopher D. Twigg, Minh Vo, Samarth Brahmbhatt, and Charles C. Kemp. ContactOpt: Optimizing Contact to Improve Grasps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the International Conference on Machine Learning*, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*, 2020.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations*, 2024.

Guangqi Jiang, Haoran Chang, Ri-Zhao Qiu, Yutong Liang, Mazeyu Ji, Jiyue Zhu, Zhao Dong, Xueyan Zou, and Xiaolong Wang. GSWorld: Closed-Loop Photo-Realistic Simulation Suite for Robotic Manipulation. *arXiv preprint arXiv:2510.20813*, 2025. URL https://arxiv.org/abs/2510.20813.

Guangqi Jiang, Yutong Liang, Jianglong Ye, Jia-Yang Huang, Changwei Jing, Rocky Duan, Pieter Abbeel, Xiaolong Wang, and Xueyan Zou. Cross-Hand Latent Representation for Vision-Language-Action Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Kushal Kedia, Tyler Ga Wei Lum, Jeannette Bohg, and C Karen Liu. Simtoolreal: An object-centric policy for zero-shot dexterous tool manipulation. *arXiv preprint arXiv:2602.16863*, 2026.

Puhao Li, Tengyu Liu, Yuyang Li, Yiran Geng, Yixin Zhu, Yaodong Yang, and Siyuan Huang. GenDexGrasp: Generalizable Dexterous Grasping. In *IEEE International Conference on Robotics and Automation*, 2023a.

Tingguang Li, Weitao Xi, Meng Fang, Jia Xu, and Max Qing-Hu Meng. Learning to Solve a Rubik's Cube with a Dexterous Hand. *arXiv preprint arXiv:1907.11388*, 2019. URL https://arxiv.org/abs/1907.11388.

Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S. Ryoo. Crossway Diffusion: Improving Diffusion-based Visuomotor Policy via Self-supervised Learning. *arXiv preprint arXiv:2307.01849*, 2023b. URL https://arxiv.org/abs/2307.01849.

Yutong Liang, Quanquan Peng, Ri-Zhao Qiu, and Xiaolong Wang. ConTrack: Constrained Hand Motion Tracking with Adaptive Trade-off Control. *arXiv preprint arXiv:2606.03177*, 2026a. URL https://arxiv.org/abs/2606.03177.

Yutong Liang, Shiyi Xu, Yulong Zhang, Bowen Zhan, He Zhang, and Libin Liu. DexterCap: Affordable and Automated Capture of Complex Hand-Object Interactions. *Computer Graphics Forum*, 2026b. doi: 10.1111/cgf.70330.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Qingtao Liu, Yu Cui, Qi Ye, Zhengnan Sun, Haoming Li, Gaofeng Li, Lin Shao, and Jiming Chen. DexRepNet: Learning Dexterous Robotic Grasping Network with Geometric and Spatial Hand-Object Representations. *arXiv preprint arXiv:2303.09806*, 2023. URL https://arxiv.org/abs/2303.09806.

Qingtao Liu, Zhengnan Sun, Yu Cui, Haoming Li, Gaofeng Li, Lin Shao, Jiming Chen, and Qi Ye. DexRepNet++: Learning Dexterous Robotic Manipulation with Geometric and Spatial Hand-Object Representations. *IEEE Transactions on Robotics*, 42:799–818, 2026. doi: 10.1109/TRO.2026.3651669.

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. *arXiv preprint arXiv:2108.10470*, 2021. URL https://arxiv.org/abs/2108.10470.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019. URL https://arxiv.org/abs/1910.07113.

Quanquan Peng, Yutong Liang, Rui Yan, Nicklas Hansen, and Xiaolong Wang. FACT: Failure-Aware Causal Training for World-Action Models. *arXiv preprint arXiv:2608.10232*, 2026a. URL https://arxiv.org/abs/2608.10232.

Quanquan Peng, Yunfeng Lin, Yufei Xue, Jiangmiao Pang, and Weinan Zhang. Embodiment-aware generalist specialist distillation for unified humanoid whole-body control. *arXiv preprint arXiv:2602.02960*, 2026b.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-Hand Object Rotation via Rapid Motor Adaptation. In *Proceedings of the Conference on Robot Learning*, 2022.

Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General In-Hand Object Rotation with Vision and Touch. In *Proceedings of the Conference on Robot Learning*, 2023.

Haozhi Qi, Brent Yi, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. From Simple to Complex Skills: The Case of In-Hand Object Reorientation. In *IEEE International Conference on Robotics and Automation*, 2025.

Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics*, 36(6):245:1–245:17, 2017. doi: 10.1145/3130800.3130883.

Omid Taheri, Nima Ghorbani, Michael J. Black, and Dimitrios Tzionas. GRAB: A Dataset of Whole-Body Human Grasping of Objects. In *European Conference on Computer Vision*, 2020.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. ManiSkill3: GPU Parallelized Robotics Simulation and Rendering for Generalizable Embodied AI. In *Proceedings of Robotics: Science and Systems*, 2025.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive Inverse Dynamics Models are Scalable Learners for Robotic Manipulation. *arXiv preprint arXiv:2412.15109*, 2024. URL https://arxiv.org/abs/2412.15109.

Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from Learning to Spin "Pens". In *Proceedings of the Conference on Robot Learning*, 2024.

Zhenyu Wei, Zhixuan Xu, Jingxiang Guo, Yiwen Hou, Chongkai Gao, Zhehao Cai, Jiayu Luo, and Lin Shao. $\mathcal{D}(\mathcal{R}, \mathcal{O})$ Grasp: A Unified Representation of Robot and Object Interaction for Cross-Embodiment Dexterous Grasping. In *IEEE International Conference on Robotics and Automation*, 2025.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation. *arXiv preprint arXiv:2312.13139*, 2023. URL https://arxiv.org/abs/2312.13139.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without Seeing: Towards In-hand Dexterity through Touch. In *Proceedings of Robotics: Science and Systems*, 2023.

Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. ManipNet: Neural Manipulation Synthesis with a Hand-Object Spatial Representation. *ACM Transactions on Graphics*, 40(4): 121:1–121:14, 2021. doi: 10.1145/3450626.3459830.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified World Models: Coupling Video and Action Diffusion for Pretraining on Large Robotic Datasets. In *Proceedings of Robotics: Science and Systems*, 2025.

A Implementation Details

A.1 Demonstrations and Training Data

We use a motion-capture glove to teleoperate the index, ring, and little fingers. Geometric retargeting maps the operator's finger configurations to robot joint commands while the supporting fingers maintain their grasp. Hand state, tactile feedback, and cube geometry from multi-view reconstruction are synchronized at 10 Hz.

The demonstration set comprises 36 sequences, 576 turns, and 25,520 frames. Action training uses successful turns with at least three samples, valid cube poses and remaining angles within 10° over the final three samples, and off-axis rotation within 10° throughout. Each turn lasts at most eight seconds (80 samples). The resulting action dataset contains 18,113 frames with valid cube poses from 435 turns: 226 U and 209 L turns.

Future prediction uses all 36 demonstration sequences, including turns that do not meet the action-training criteria. Each current/future pair spans a single uninterrupted turn with valid cube poses throughout. Non-increasing timestamps and gaps greater than 0.15 seconds separate continuous segments. The prediction dataset contains 24,943 current frames with at least one valid future target, yielding 24,943, 22,635, and 19,750 pairs at offsets of one, five, and ten samples. We sample action and prediction batches separately.

A.2 Observations and Encoders

The five finger-state vectors use the order index, ring, little, thumb, and middle. Each has 24 entries: five joint-position slots, five velocity slots, five torque slots, six tactile force/torque channels, and three fingertip coordinates. Joint quantities are zero-padded for fingers with fewer than five joints. The index, ring, and little fingers contribute four, four, and five controlled joints, respectively. The remaining-angle input is measured in units of a 90° turn.

The cube representation is constructed from eight tracked cubie centers. The reconstruction identifies the two four-cubie layers, then derives 24 face-center points and eight outer corners. Both these points and the fingertip positions are transformed into the palm frame. Numeric observations and global cube coordinates are standardized using training-set statistics and clipped to $[-5, 5]$. Local geometry in Equation 1 uses the unnormalized, unclipped palm-frame coordinates to preserve metric distances between the fingers and cube.

Each finger has a linear numeric projection to 384 dimensions. The global point network has dimensions $3 \to 384 \to 384$, with a GELU between its linear layers, followed by maximum pooling over the points. The local network has dimensions $3 \to 64 \to 64$, a GELU, mean pooling, and a $64 \to 384$ projection. The same local network is shared across fingers and points.

Tactile deformation images from the three controlled fingertips are resized to 96×96 and scaled to $[0, 1]$. The shared convolutional encoder has channel widths 32, 64, 128, and 384, stride two at every layer, and kernels of size five in the first layer and three thereafter. A learned finger-specific scale and shift modulate every layer, followed by GELU activation. Global spatial averaging yields one token per tactile image. The observation transformer has four pre-normalized layers, eight attention heads, width 384, feed-forward width 1,536, zero dropout, and a final layer normalization. Its feed-forward sublayers use ReLU. The three learned future queries in the full model increase its memory from nine to 12 tokens. A shared $384 \to 384 \to 23$ MLP with GELU predicts the targets from each encoded future token.

A.3 Action Normalization and Objective

Let $u_{t+k} \in \mathbb{R}^{13}$ denote a demonstrated commanded joint position and q_t the measured configuration at the current observation. The action target is the offset $a_{t,k} = u_{t+k} - q_t$. For joint j, let $Q_{.01,j}$ and $Q_{.99,j}$ be the first and 99th percentiles of the training offsets. We use

$$b_j = \frac{Q_{.01,j} + Q_{.99,j}}{2}, \qquad s_j = \max\left(\frac{Q_{.99,j} - Q_{.01,j}}{2}, 0.01\right), \qquad A_{t,k,j} = \frac{a_{t,k,j} - b_j}{s_j}, \quad (8)$$

where the minimum scale is in radians. The action horizon is 20 samples. A binary mask $w_{t,k}$ excludes targets beyond the available turn segment. Writing $\bar{s}^2 = \frac{1}{13}\sum_j s_j^2$, the optimized flow-policy loss is

$$\mathcal{L}_{\text{act}} = \frac{\sum_{t,k,j} w_{t,k} s_j^2 (\widehat{A}_{t,k,j} - A_{t,k,j})^2}{13\bar{s}^2 \sum_{t,k} w_{t,k}}. \tag{9}$$

Thus the clean-action reconstruction error is measured in joint space and divided by the mean squared action scale. Noise levels are sampled from $\mathrm{Beta}(1.5, 1)$. The velocity in Equation 5 is the parameterization used for integration; Equation 9 is the training objective.

The four action-decoder blocks contain self-attention, cross-attention to the encoded observation memory, and a GELU feed-forward network. Each sublayer receives a noise-dependent shift and scale. A sinusoidal noise-level embedding uses logarithmically spaced periods from 0.004 to 4, followed by a two-layer SiLU MLP. Learned embeddings specify the 20 action positions. Four Euler updates with step size -0.25 evaluate the field at noise levels 1, 0.75, 0.5, and 0.25. The sampled normalized offsets are converted back with b_j, s_j and added to the current measured joint configuration.

For future prediction, each target coordinate is divided by its empirical standard deviation for that horizon, with a minimum scale of 10^{-6}. The standardization does not subtract a target mean. The contact target contains the three measured force components for each controlled finger; torque channels are observation inputs but are not part of this target. Equation 4 averages only valid temporal pairs, with equal weight for the three target groups.

A.4 Optimization and Baseline Settings

We train each policy for 1,000 epochs (71,000 optimizer steps) with batches of 256 action examples. Training uses AdamW with $\beta = (0.9, 0.95)$, weight decay 10^{-4}, a 1,000-step learning-rate warmup followed by cosine decay, gradient-norm clipping at one, and bfloat16 mixed precision. Independent Gaussian noise with standard deviation 0.01 augments normalized numeric features and cube coordinates for action training. Exponential moving averages use decay 0.99 and provide the deployment parameters. All flow policies use learning rate 5×10^{-4}; ACT and Diffusion Policy use 10^{-4}.

Our predictive branch draws an additional batch of 256 eligible current frames at each step while its weight is nonzero. The weight begins at 0.003 and decreases linearly to zero at step 35,500. Both action and prediction gradients update the same observation encoder. The action objective continues for the remaining training steps, retaining the learned future queries and the same action-conditioning memory.

ACT uses the shared observation interface with a 32-dimensional variational latent, a four-layer posterior transformer, an L_1 action reconstruction objective, and KL weight 10. The posterior reads the current joint configuration and the demonstrated chunk during training; inference fixes the latent to zero. Diffusion Policy uses 100 training diffusion steps with a cosine schedule, predicts noise with a four-layer transformer decoder, and uses ten deterministic denoising updates at deployment. Both methods have the same width, action horizon, and observation-encoder depth as Base Flow. Their input encoders use the global cube feature without the proposed local geometry or future-token prediction.

A.5 Deployment and Discrete Planning

Inference runs asynchronously from the 10 Hz command loop. An action sample is aligned to its observation timestamp, and execution selects the chunk entry corresponding to the current time. A newly inferred chunk can replace the active chunk after at least five executed samples; expired chunks can be replaced immediately. Commanded joint positions are clipped to robot joint limits. The benchmark uses the same execution rules for every method.

The discrete planner represents the orientations of the eight cubies and searches over positive U/L quarter turns. Its objective first minimizes the number of quarter turns and then prefers lower repeated-layer cost among tied solutions. A compatible solved two-cubie block defines a holding grasp. If a complete solution is unavailable under the current grasp, the system creates such a block, places the cube, and picks it up in the new grasp before solving the remaining state. State observations between sequences determine whether alignment or another plan is needed.

Table 4: **All layer-turn evaluation rounds.** Outcome entries are percentages. U, L, and All times average every attempt of the corresponding direction or the full round, including failures, in seconds. Each row contains 50 U and 50 L attempts.

Method	Seed	Outcomes (%)					Mean time (s)		
		U success ↑	L success ↑	Timeout ↓	Drop ↓	All success ↑	U ↓	L ↓	All ↓
ACT	0	66	76	29	**0**	71	7.29	6.76	7.02
ACT	1	68	88	22	**0**	78	6.69	6.05	6.37
ACT	2	82	86	16	**0**	84	6.04	5.54	5.79
Diffusion Policy	0	80	92	13	1	86	6.40	5.42	5.91
Diffusion Policy	1	66	80	26	1	73	6.30	6.08	6.19
Diffusion Policy	2	88	66	22	1	77	5.59	6.57	6.08
Base Flow Policy	0	70	74	27	1	72	5.65	5.74	5.69
Base Flow Policy	1	82	76	20	1	79	5.59	6.00	5.79
Base Flow Policy	2	96	80	12	**0**	88	5.30	5.74	5.52
Local Geometry	0	96	92	6	**0**	94	5.44	5.03	5.23
Local Geometry	1	96	92	6	**0**	94	5.30	5.26	5.28
Local Geometry	2	86	94	9	1	90	5.51	**4.89**	5.20
FINGR	0	**100**	**100**	**0**	**0**	**100**	5.42	5.31	5.36
FINGR	1	**100**	98	1	**0**	99	**4.83**	5.19	**5.01**
FINGR	2	98	98	2	**0**	98	5.36	5.36	5.36

B COMPLETE LAYER-TURN RESULTS

For a round with $N = 100$ attempts, the success, timeout, and drop percentages use all N outcomes as the denominator. Direction-specific success uses the 50 attempts of that direction. For any per-round metric x_r, the reported summary is

$$\bar{x} = \frac{1}{3}\sum_{r=1}^{3} x_r, \qquad s_x = \sqrt{\frac{1}{2}\sum_{r=1}^{3}(x_r - \bar{x})^2}. \tag{10}$$

The three rounds use deployment seeds 0, 1, and 2 with one checkpoint per method. Table 4 gives the per-round results; Base Flow and FINGR use the same evaluations in the baseline and ablation comparisons. For FINGR, mean U and L times are 5.21 ± 0.32 and 5.29 ± 0.09 seconds, respectively, including failed attempts.

C COMPLETE CUBE-SOLVING EVALUATION

C.1 SCRAMBLES, SOLUTIONS, AND CORRECTIONS

Table 5 lists the ten scramble sequences and their corresponding solution plans. In a scramble, a prime denotes a negative quarter turn and a suffix 2 denotes a half turn in conventional cube notation. In a solution, U and L refer to the two controllable layers in the current robot grasp; suffixes 2 and 3 expand to two and three positive quarter turns. A slash marks the table-assisted regrasp between two solution sequences. The layer coordinate convention is therefore updated at that boundary. An arrow marks replanning under the same grasp.

The first solve requires a 1.80-second L alignment after its first sequence. After the final planned U turn in the third solve (6.30 seconds), a 0.70-second U alignment completes the solution. The eighth and ninth solves use L and U alignment corrections of 2.30 and 3.20 seconds, respectively, before regrasping. These partial-angle corrections are reported separately from the planned $90°$ turns. In solve 8, the post-regrasp plan stops after its twelfth attempt, a U state mismatch; reobservation produces the plan `L3`. Its first L attempt also produces a state mismatch, after which the next observation verifies a solved cube. Thus, solve 8 executes 20 planned quarter-turn attempts. Solve 9 similarly verifies completion after a state mismatch on its final planned U turn.

Table 5: **Scramble and solution formulas.** Solution formulas use grasp-relative positive U/L turns. The slash denotes a regrasp; an arrow denotes replanning under the same grasp. Small alignment corrections are timed separately.

ID	Scramble	Planned solution sequences
1	R′ F U2 R2 U2 R′ F U′ F R2 U2	UL2U3L3 / LUL2ULU2L
2	R′ U2 F U F U′ F R2 F′ R U′	LU2LUL2 / U3LULULUL
3	U′ F′ R U2 R U′ R2 U′ F′ U2 R	LU3L / L2ULU3
4	U′ R′ U2 R′ F U′ F′ U2 R U′ R	LULU2L3UL / ULU2LULU2
5	U R′ F′ R2 U2 R2 U′ F U2 F′ R2	LULULULUL3U2LULU
6	U′ R′ U′ F′ U R′ U2 R U2 F U2	LU / U2LUL2U2LULUL2
7	U R′ F U2 F2 R U′ F′ R U2 R′	U2L3UL / ULULU2L3UL3
8	R U2 R2 U′ R′ U2 R′ F U′ R U′	U2LULUL / LULULULULU3L3 → L3
9	R′ U′ R U F U′ R′ U′ F U2 F	ULU2 / U2LU2LULUL2U2LU2
10	R′ F′ R U2 R U′ R′ U′ F U′ F	LULU2L3U2LU / U2LULUL3ULUL2

Table 6: **Time breakdown for each complete solve.** Seconds. Pickup includes initial planning, pregrasp, approach, closure, lift, and rotation into the manipulation pose. Regrasp includes intermediate placement and pickup in the new grasp. Observe includes the remaining state-observation and transition intervals.

ID	Pickup ↓	Layer turns ↓	Alignment ↓	Regrasp ↓	Place/return ↓	Observe ↓	Total ↓
1	8.72	90.20	1.80	24.95	11.40	3.80	140.87
2	8.85	132.00	**0.00**	39.05	10.61	3.08	193.59
3	9.02	71.10	0.70	24.56	10.30	3.79	119.48
4	**8.06**	91.10	**0.00**	25.85	12.96	3.66	141.62
5	8.31	**63.40**	**0.00**	**0.00**	**10.10**	**2.26**	**84.08**
6	9.24	69.70	**0.00**	40.43	11.38	3.40	134.14
7	9.27	84.21	**0.00**	30.43	11.31	2.88	138.09
8	8.93	87.40	2.30	29.18	13.70	5.58	147.09
9	8.95	65.40	3.20	27.80	10.83	4.94	121.12
10	8.46	94.50	**0.00**	27.60	12.16	3.49	146.23

C.2 System Success and Timing

During complete cube solves, each turn continues until completion or a detected execution error. The ten-second per-attempt timeout applies to the isolated layer-turn benchmark.

Complete-system time spans initial grasp planning through final placement and robot return. For each solve, U and L times average all planned attempts of that direction. Time per 90° is the total time of all such attempts divided by their count. We report the mean and sample standard deviation across ten solves. The full-solve success rate is $10/10$, and the planned-turn success rate is $181/185$. Across solves, mean U and L times are 5.10 ± 1.97 and 4.31 ± 0.86 seconds, respectively.

Table 7 reports outcomes for each system stage. Motion planning and its associated checks (Table 3) take 2.35 ± 1.04 seconds per solve and are included in the corresponding arm-motion stages.

Table 7: **System-stage outcomes and durations.** Success rates are percentages; time is mean ± sample standard deviation in seconds per operation. The first four stages form initial pickup.

Stage	Completed / attempted ↑	Success ↑	Duration ↓
Pregrasp and initial planning	10 / 10	**100.0**	3.33 ± 0.06
Approach	10 / 10	**100.0**	1.00 ± 0.00
Grasp closure	10 / 10	**100.0**	2.00 ± 0.05
Lift and manipulation-pose rotation	10 / 10	**100.0**	2.46 ± 0.41
Planned layer turn	181 / 185	97.8	4.59 ± 3.13
Alignment	4 / 4	**100.0**	2.00 ± 1.04
Table-assisted regrasp	9 / 9	**100.0**	29.98 ± 5.86
Final placement and return	10 / 10	**100.0**	11.48 ± 1.16