Visual Dexterity: In-Hand Reorientation of Novel and Complex Object Shapes

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, Pulkit Agrawal

Science Robotics, 2023 · MIT CSAIL · Meta AI

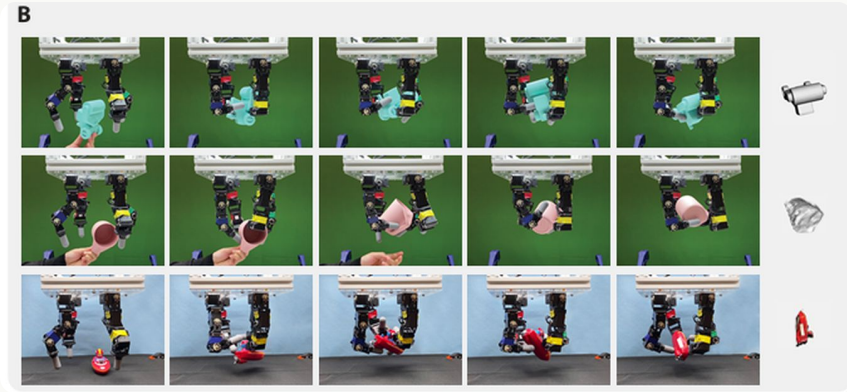


Fig. 1B — one controller, three new objects

01

Introduction

Why in-hand reorientation matters, and why it is hard

Why dexterous hands?

Human hands do

- Re-arranging objects
- Loading dishes into a dishwasher
- Fastening bolts
- Cutting vegetables
- Many other forms of **tool use**

Robots today

Despite long-standing interest, robots are still far behind in

- Versatility
- Dexterity
- Robustness

In-hand reorientation: a precursor to tool use

> **Task**
> Rotate a hand-held object from an **arbitrary initial** orientation to an **arbitrary target** orientation

- After picking a tool, the robot must orient it before use
- e.g., a screwdriver works only if its head is aligned with the screw
- A **litmus test** for dexterity and an **enabler** of downstream tasks

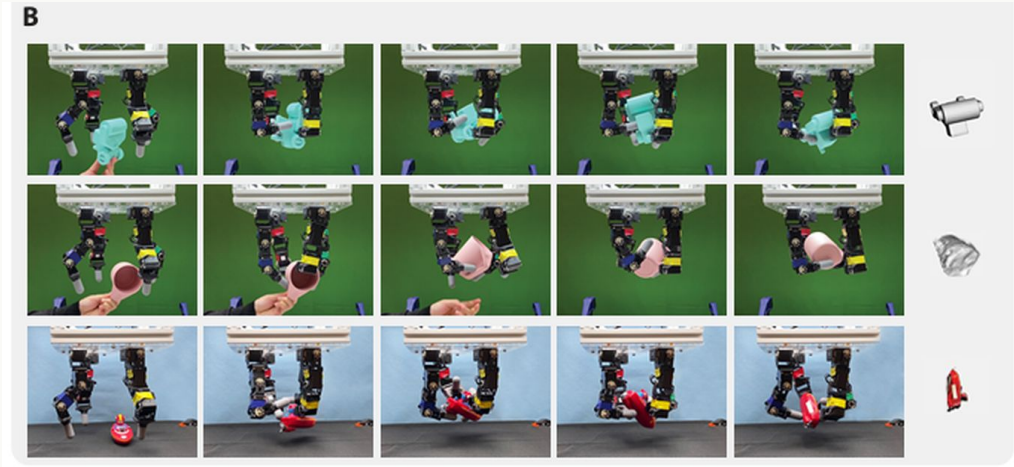


Fig. 1B — rows 1–2: in the air; row 3: with a table. Right: target

What a real-world-ready system needs

Any orientation

Full space of rotations, not one axis

New objects

Generalize to shapes never seen in training

Real time

Using commodity sensors only

“Benign” setup choices that break deployment

- **Multiple cameras** around the workspace → reduces occlusion, but impractical on a mobile manipulator
- **Upward-facing hand** → much easier control, but limits downstream tool use

Upward- vs downward-facing hand

	Upward-facing (prior work)	Downward-facing (this work)
Object	Rests on the palm	Held from the top
Gravity	Helps keep the object	Must be counteracted
Small finger error	Object usually stays	Object falls — irreversible
Control	Easier	Much harder
Tool use	Limited	Needed for most tool use

This paper tackles the **harder, more useful** setting.

Challenge ① Control

Continuous–discrete optimization

- Many fingers → high-dimensional control space
- **How much** force, **when** → continuous
- **Where** fingers make and break contact → discrete

→ Often computationally intractable

Prior simplifications

- Simple convex shapes (polygons, cylinders)
- Hand-designed finger motion patterns
- Fingers never break contact
- Upward-facing hand
- Quasi-static manipulation

Result: limited objects, scenarios, or orientations (e.g., a single axis)

Challenge ② Perception

The controller needs state: object pose, surface friction, finger contact, ...

Touch sensors

Only local contact info; not plug-and-play

Cameras

Fingers **occlude** the object during manipulation

RGBD pose estimators

One estimator **per object** → no generalization to new shapes

No prior work had demonstrated a real-world-ready reorientation system.

Where prior work stands

Work	Objects	Hand	Rotation	Sensing	Setting
OpenAI (8)	One cube	Upward, Shadow Hand ($100k+)	Passes the goal; no stopping	3 RGB cameras + motion capture	Real
Duck reorientation (23)	Duck	Downward, under-actuated	Two axes only	Object-specific pose tracker	Real, quasi-static (70 s+)
Chen et al. (7)	Diverse objects	Simulated	Full SO(3)	Full state	Simulation only
This work	New, complex shapes	Downward, D'Claw (<$5k)	Full SO(3), stops at goal	One depth camera	Real, dynamic (~7 s)

Numbers in parentheses = reference numbers in the paper

RL works in simulation — with full state

Given object position, velocity, pose and proprioception, deep RL finds an approximate solution to reorientation for diverse objects **in simulation** (7).

But it needs lots of data and full state → trainable only in simulation. Two open questions:

Q1

How to train with sensors available in the real world, such as vision?

Q2

Does a controller trained in simulation transfer to the real world?

Key contributions

1. A **single vision controller** that reorients new, complex shapes by any rotation in **SO(3)**, in real time (12 Hz), from **one depth camera**
2. **Two-stage student training**: synthetic → rendered point clouds, **5× faster** training
3. **Sparse 3D CNN** acting directly on point clouds — no per-object pose estimator
4. **In-air reorientation** from one reward term + four fingers; one controller for table and air
5. **Open-source hardware under $5,000**, usable beyond reorientation

02

Method

Teacher–student RL, two-stage point-cloud student, sim-to-real

Problem setup

Item	Setting
Initial state	Random object, random pose in SE(3)
Goal	Point cloud of the object in the target orientation (SO(3))
Observation	1 RealSense depth camera + joint positions
Action	Relative joint-position change, 12 Hz
Hands	D'Claw 3-finger (9 DoF), 4-finger (12 DoF)
Transfer	Zero-shot sim-to-real

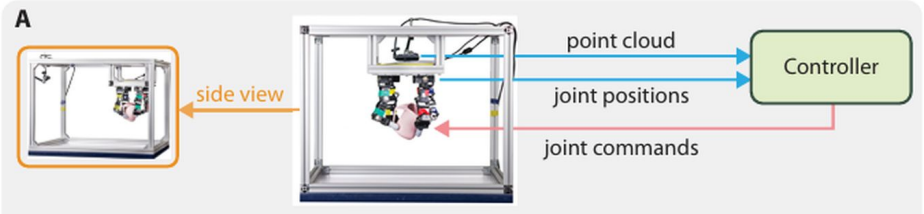
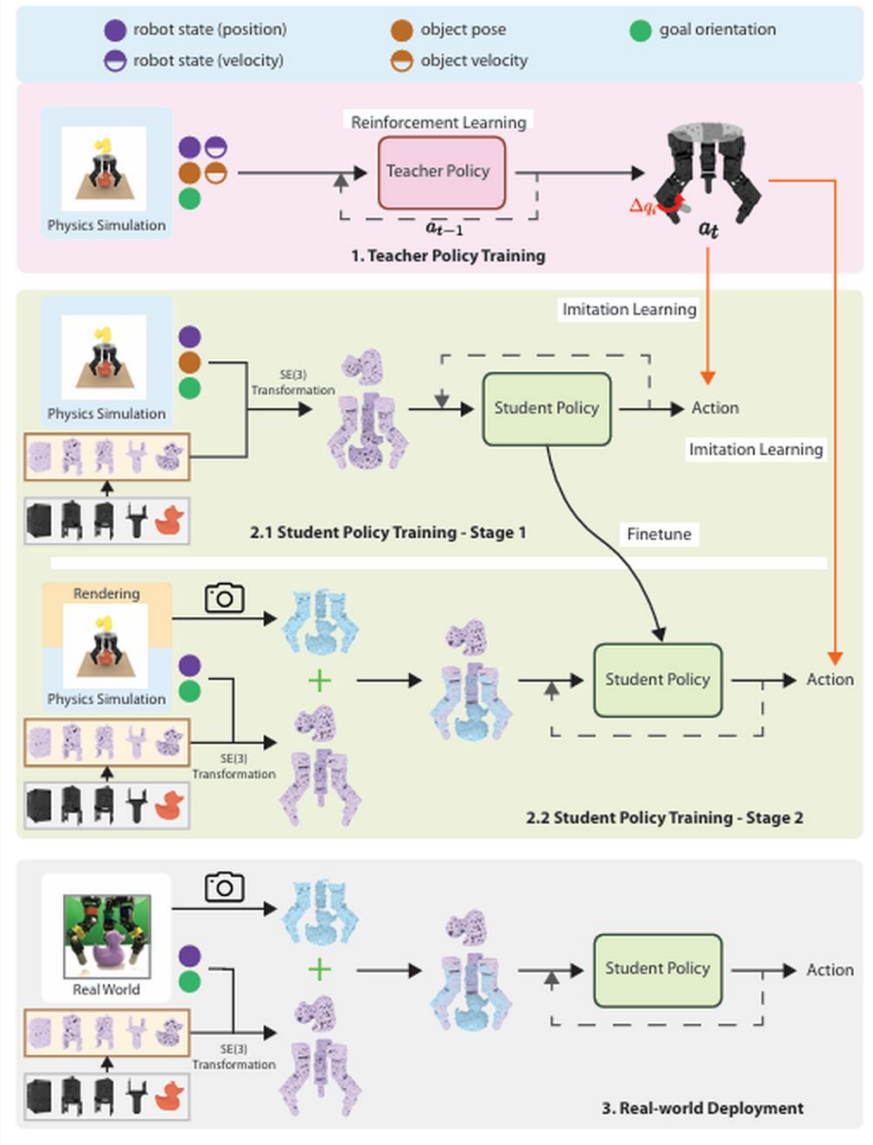


Fig. 1A — real-world setup and control loop

Key terms

Symbol	Meaning
π_E	Teacher policy — RL with privileged (simulator-only) state
π_S	Student policy — vision input, imitates the teacher
$o_t,\ a_t$	Observation and action at time t
$a_t \in \mathbb{R}^{3G}$	Relative joint-position change; G = number of fingers (3 joints each)
$\Delta\theta_t$	Rotational distance between current and target orientation
P_t^s	Scene point cloud (camera + hand)
P^g	Goal point cloud: the object in the target orientation
$\mathrm{SO}(3),\ \mathrm{SE}(3)$	All 3D rotations; all 3D rotations + translations

Overall framework: teacher → two-stage student



1. Teacher

RL (PPO) with privileged state: robot pos/vel, object pose/vel, goal

2.1 Student, stage 1

Imitate the teacher from **synthetic, complete** point clouds

2.2 Student, stage 2

Fine-tune on **rendered, occluded** point clouds

3. Deployment

Real camera + computed hand points → student controls the robot

Fig. 6 — training framework

Teacher policy: RL with privileged information

Training

- **PPO**, one policy across many objects
- Maximize $\mathbb{E}\left[\sum_t \gamma^t r_t\right]$
- Input: joint pos/vel, object pose/vel, goal, a_{t-1}
- Strictly a POMDP (no shape), modeled as an **MDP** — no memory

Action (12 Hz position control)

Exponential moving average for smooth control

$$\bar{a}_t = \alpha\, a_t + (1-\alpha)\, \bar{a}_{t-1}$$
$$\alpha = 0.8$$
$$q_{t+1}^{\text{tgt}} = q_t + \bar{a}_t$$

Analogy: the teacher has the answer sheet; the student only sees through a camera.

Derivations → Appendix p. 48–49

KEY INSIGHT

The teacher never sees the object’s shape — yet reorients diverse, new objects

No shape input
Only position, pose, velocity

No memory
Not recurrent → cannot build shape from a sequence of touches

Now real
Seen before only in simulation (7); now confirmed in the real world

Why it matters: without this, the teacher’s input could not be a low-dimensional state — and teacher–student training would not work.

Reward r_1: reorientation on a table

Eq.	Term	Sign	Purpose
(1)	$\mathbf{1}(\text{task successful})$	+	Sparse task reward — the real goal
(2)	$\frac{1}{\lvert\Delta\theta_t\rvert + \epsilon}$	+	Dense reward: closer to target, larger reward
(3)	$\sum_i \left\lVert \text{fingertip}_i - \text{object center} \right\rVert^2$	–	Keep fingertips near the object; without it, fingers barely touch it
(4)	$\lvert\text{joint velocity}\rvert^\top \lvert\text{torque}\rvert$	–	Energy penalty
(5)	$\mathbf{1}(\text{object pushed too far})$	–	Penalty + episode terminated

The sparse term alone is not enough to learn → **reward shaping** with dense terms (2)–(5)

Full formula and derivations → Appendix p. 50

Stopping at the goal, and reward r_2 for in-air

Success criterion

- Orientation-only criterion → object **oscillates** around the target
- Add penalties on **finger and object motion**
- Success = reach the target **and stay still**

$r_2 = r_1$ + two terms

- (7) – object touches the table
- (8) – fingertip above a height → stops pushing with the penultimate joint (not critical, more natural)

Trained on a table, yet in-air reorientation emerges — no gravity curriculum, one controller.

Full r_2 formula → Appendix p. 51

Student input: everything as one point cloud

Camera points
Depth → point cloud in robot base frame (occluded)

Hand points
Joint angles → forward kinematics → finger surfaces (complete)

Goal points P^g
The object as it should look at the end

$$P_t = P_t^s \cup P^g$$

Merged **before** the network → faster learning (ablation, Suppl. S5.4)

- **Why a goal point cloud?** A pose needs a reference frame; no common frame exists across objects
- **Why hand points?** Fingers and object in the same 3D space → spatial relations visible to the CNN

How the points are computed → Appendix p. 52

Student network architecture

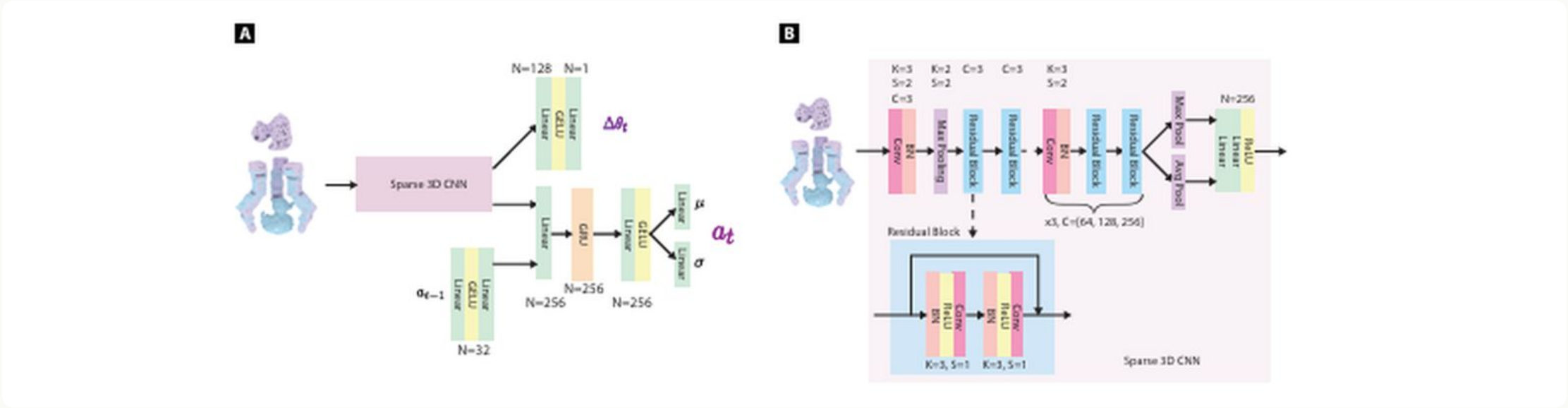


Fig. 7A, 7B — policy network and its sparse 3D CNN

Sparse 3D CNN

Minkowski Engine, residual blocks; larger than (7)

GRU

No object velocity → integrate time

Outputs

$a_t \sim \mathcal{N}(\mu, \sigma)$

$\Delta\theta_t$ head for stopping

Training

DAgger imitation of π_E

Gaussian policy → Appendix p. 51

Why a sparse 3D CNN?

WHY CNN?

- Point cloud = unordered set → voxelize into a grid
- Captures **local shape**: edges, curves, bumps
- Shared filters → same shape, same response anywhere

WHY SPARSE?

- Voxels grow with resolution³
- Points lie only on surfaces → most voxels empty
- Compute only occupied voxels → **12 Hz**

WHY IT MATTERS

- Local shapes recur across new objects → generalization
- A slipping object needs fast feedback

The paper gives real-time speed and capacity as reasons; no direct comparison with PointNet or Transformers is reported.

Two-stage student training

Rendering from scratch: **20+ days** on one V100 — rendering eats time and GPU memory

	Stage 1: synthetic	Stage 2: rendered
Point source	Pre-sampled mesh points, moved by link pose	Simulated depth image + hand via forward kinematics
Occlusion	None (complete)	Yes (like a real camera)
Wall-clock	< 3 days (7× faster)	~1 day fine-tuning
Role	Learn the behavior fast	Adapt to realistic input

$$P_t^{l_j} = P^{l_j}\left(R_t^{l_j}\right)^\top + \left(p_t^{l_j}\right)^\top, \qquad P_t^s = \bigcup_j P_t^{l_j}$$ → matrix multiplication only

Bonus: stage 1 is camera-agnostic → a new camera pose only needs stage 2

Derivation → Appendix p. 52

Two-stage training: faster and better

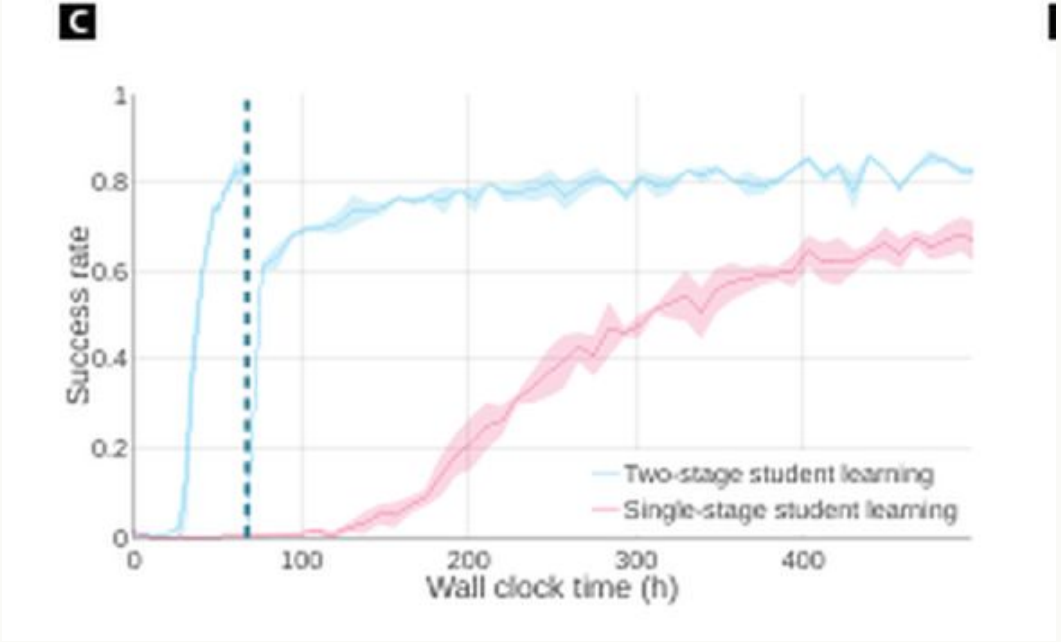


Fig. 7C — success rate vs. wall-clock time

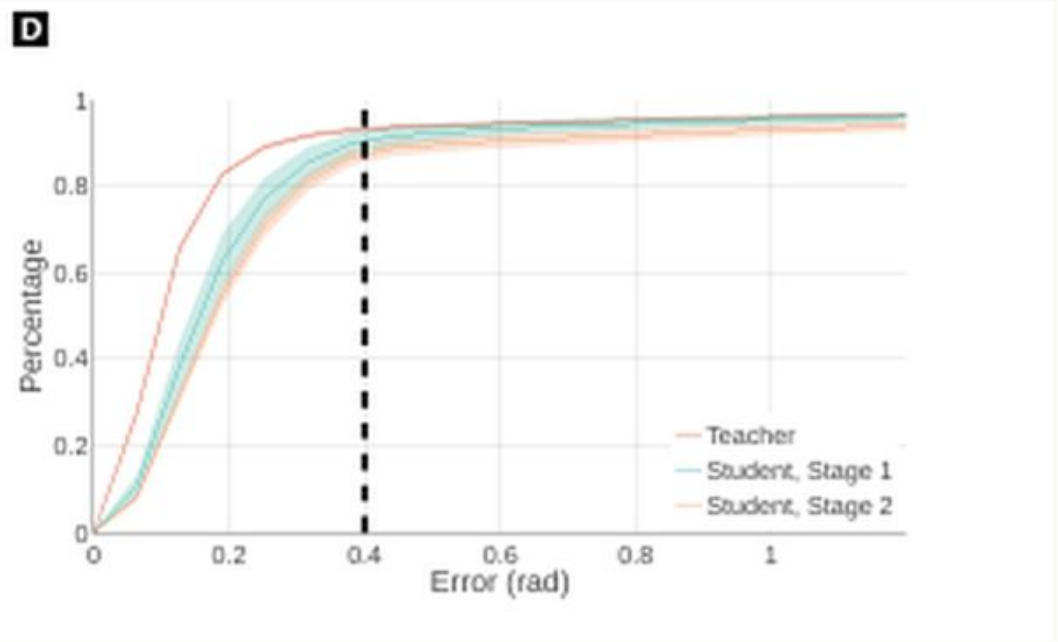


Fig. 7D — error ECDF: teacher vs. students

- Two-stage: ~0.8 success at **~70 h**; single-stage (render from scratch): **~0.65 after 500 h**
- Dip at the dashed line = input switches from unoccluded to occluded; recovers quickly. Students ≈ teacher

Reducing the sim-to-real gap

Dynamics gap — robot dynamics, contact model, object friction

Perception gap — sensor statistics and noise

Target	Countermeasure
Perception	Depth only (closer to sim than RGB) + noise on simulated points
Robot dynamics	System identification → small randomization
Object / environment	Cannot be identified → wide randomization
Unmodeled physics	Random external forces on the object during training
Hardware	Soft fingertips: more friction and compliance, tolerate command errors

Too much randomization → over-conservative policy. Match what you can; randomize only the rest.

System identification with parallel simulation

URDF from CAD gives accurate kinematics, but joint **damping and stiffness** must be estimated. Isaac Gym is closed-source → no equations of motion, no gradients → **black-box search**.

1

Spawn many simulated robots with different parameters λ on the GPU

2

Send the same commands to real and simulated robots

3

Score = − squared distance between joint trajectories

4

CMA-ES searches for the best λ

$$h = -\left\| q_{\text{real}}(\cdot) - q_{\lambda}(\cdot) \right\|^2, \qquad \lambda^* = \arg\max_{\lambda} \; h$$

An exact match may not exist (approximate parameterization, real-world noise) → keep small randomization

Least squares and CMA-ES → Appendix p. 53

Deployment and the stopping criterion

At deployment

- Motor joint positions + RealSense depth → one point cloud
- A head on the policy features predicts $|\Delta\theta_t|$

Stop when both hold

$$\Delta\theta_t^{\text{pred}} < \bar{\theta}$$

$$\|a_t\| < \bar{a}$$

“Almost there” and “barely moving”

- **Why both?** Distance alone could stop while the object is swinging through the target
- **Why stop at all?** Tool use needs the object held still in the target pose
- **Weak spot:** |Δθ| prediction is less accurate when the true distance is below 0.4 rad

Details → Appendix p. 54

03

Results

On a table, in the air, with daily objects, and against prior work

Experimental setup

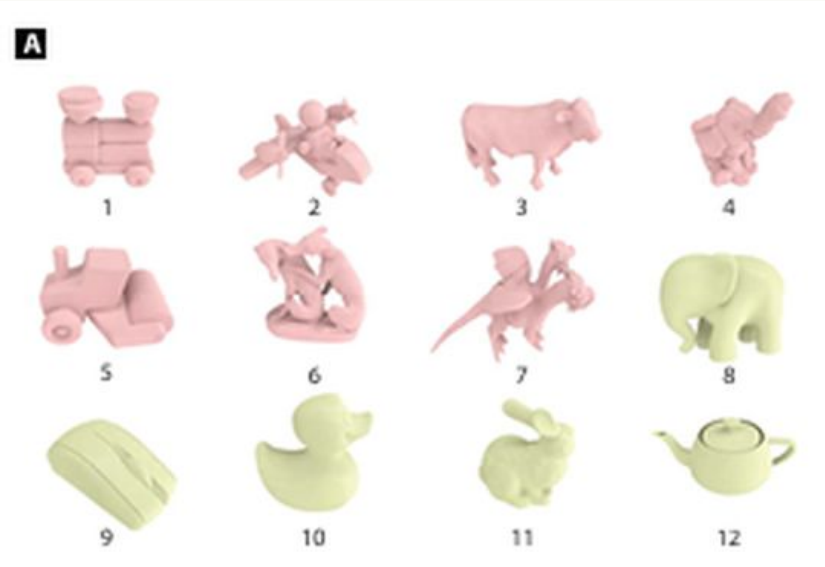


Fig. 2A — pink 1–7: train set; yellow 8–12: test set

- **Train:** 150 objects (dataset B); **test:** 12 ContactDB objects (dataset S), no shape overlap
- Evaluated on 7 in-distribution + 5 out-of-distribution (OOD) objects
- **3D-printed** → same shape as in simulation
- 20 trials per object, random initial and target orientation
- Error from **OptiTrack** (evaluation only), measured when the controller **stops**
- Success: ≤ 0.4 rad (23°) or ≤ 0.8 rad (46°)

How orientation error is measured → Appendix p. 54

Q1. How well does it reorient on a table?

Success rate	Rigid Train	Rigid Test	Soft Train	Soft Test	Sim Train	Sim Test
≤ 0.4 rad (23°)	81%	45%	79%	55%	96%	85%
≤ 0.8 rad (46°)	95%	75%	98%	86%	98%	87%
Median error CI (rad)	0.20–0.27	0.29–0.46	0.21–0.28	0.33–0.42	0.12–0.13	0.15–0.18

Table 1 (retyped). Real hand on table; Train = 7 objects from B, Test = 5 objects from S. 95% BCa bootstrap CIs in the paper.

Train objects: **~80%** within 0.4 rad

Soft fingertips: OOD **45% → 55%**

At 0.8 rad, sim–real gap shrinks → **precision is the hard part**

Q1. Reading the error distributions

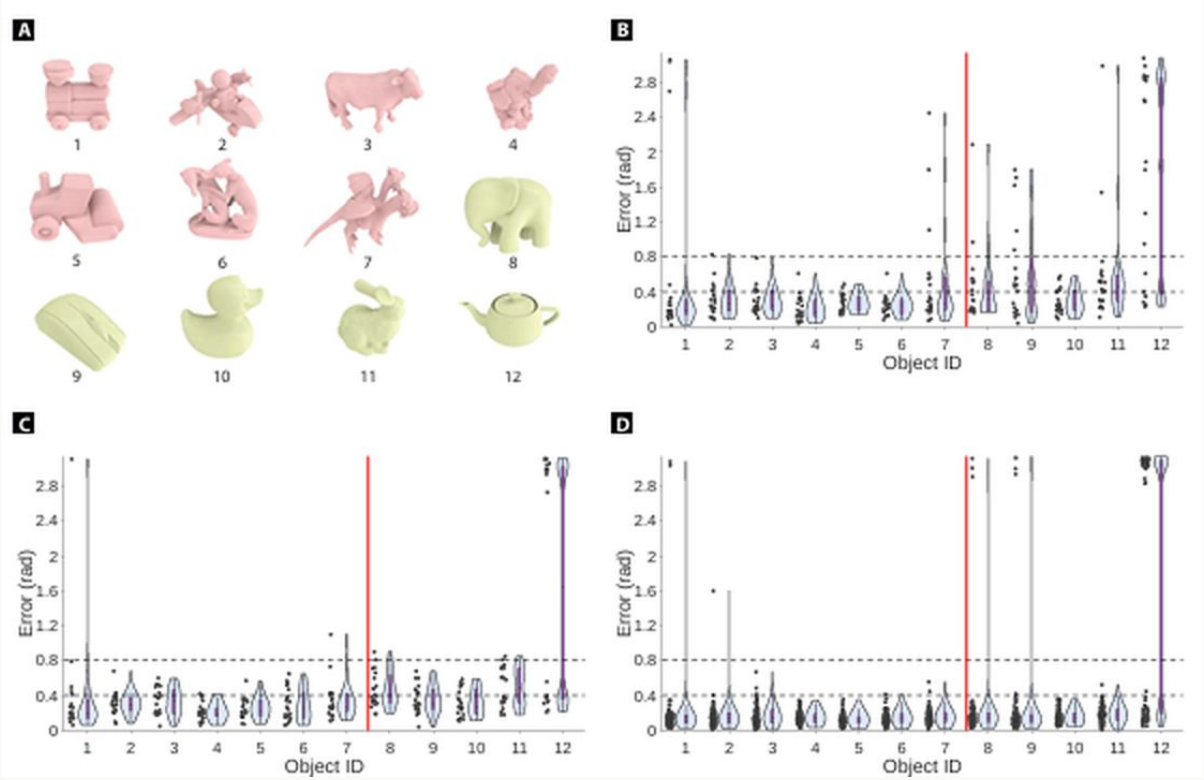


Fig. 2B real rigid · 2C real soft · 2D simulation

- Wide part = frequent error; box = 25–75%, bar = median
- Dashed lines: 0.4 and 0.8 rad. Red line: train | test
- Sim errors lower, but **same trend** as real
- **#12 teapot**: nearly symmetric in a point cloud → errors near π (180°), a **bi-modal** distribution
- High-error runs mostly from **wrong stopping**, not wrong actions (authors' hypothesis)

Q2. Do soft fingertips help?

Rigid, hemispherical

- Near point contact → small action errors amplified
- Low friction → slips
- Failures: overshoot, finger slipping

45% OOD

Soft elastomer

- Higher friction; deforms on contact (compliance)
- Larger contact area absorbs small errors
- Smoother, less aggressive motion

55% OOD

Train objects barely change (81% → 79%). Trained in a **rigid-body** simulator, yet no loss with soft tips — observed, not explained.

Q3. Is it robust to the table surface?

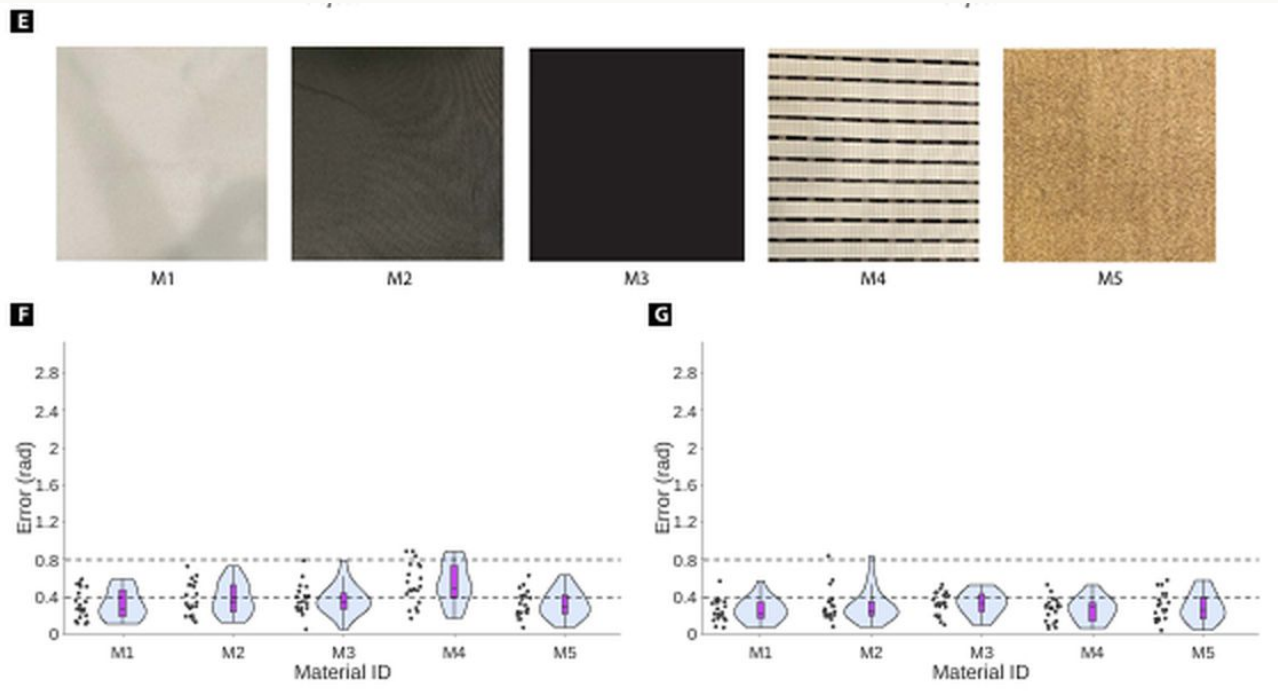


Fig. 2E materials · 2F object #5 (train) · 2G object #10 (test)

ID	Material
M1	Rough cloth
M2	Smooth cloth
M3	Slippery acrylic
M4	Perforated bath mat
M5	Uneven door mat

Similar error on all five → **robust** to surface dynamics

Q4. In the air: why four fingers?

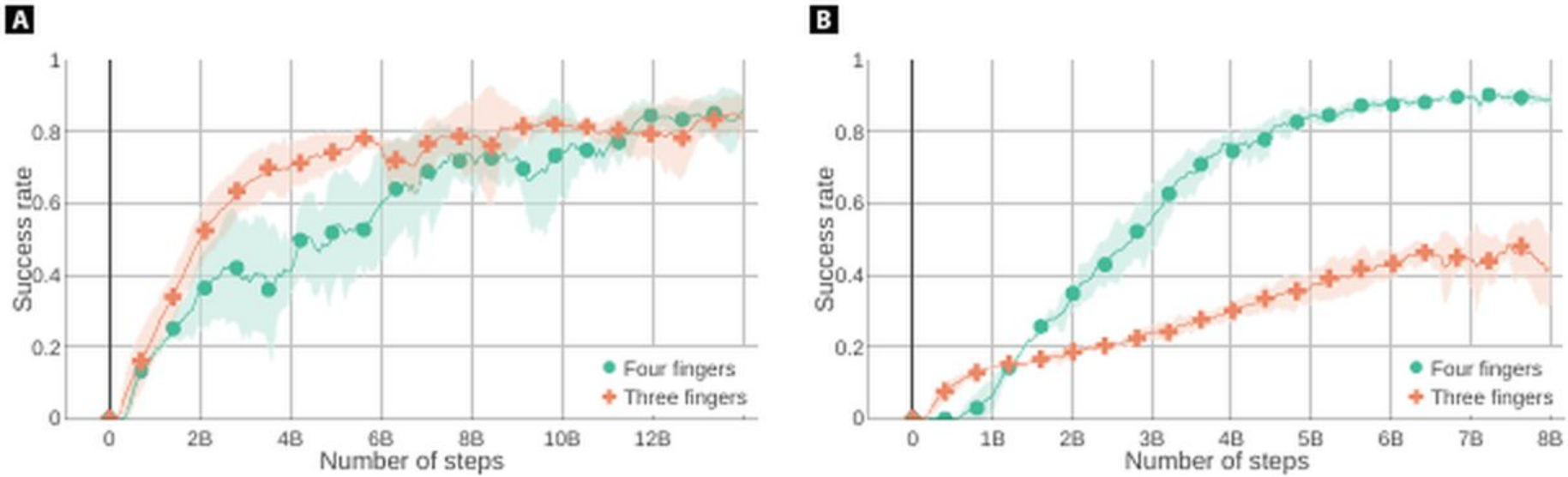


Fig. 4A with a table (left) · 4B incentivized to lift (right). x-axis: training steps (B = billion)

Table-trained controller + no table → **always drops**

Lifting: 4 fingers **~0.9** vs 3 fingers **~0.45**

Hypothesis: more finger configurations work + redundancy against errors

Q5. In-air performance: error, time, drops

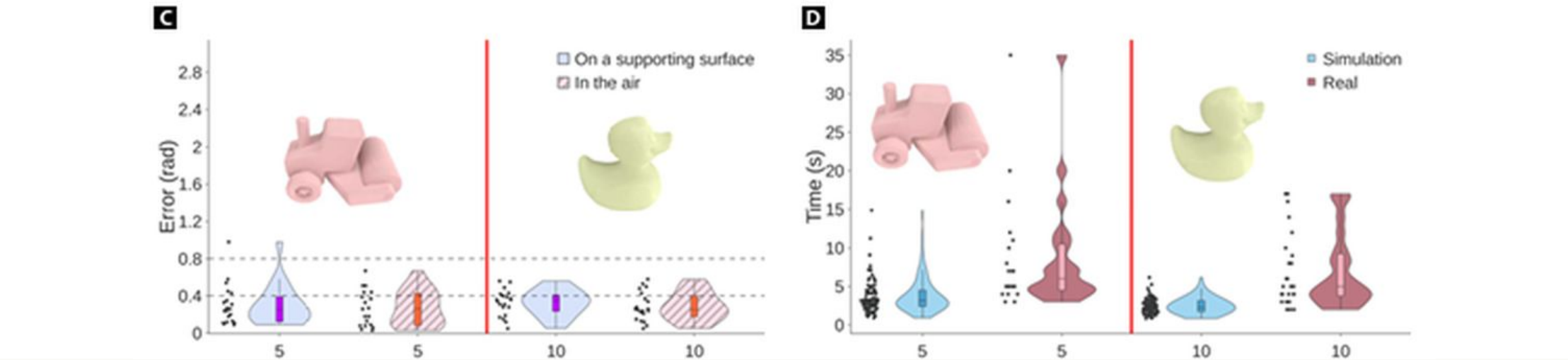


Fig. 4C error, table vs. air (20 non-drop trials) · 4D episode time, sim vs. real

Precision

Non-drop error ≈ on-table → no loss in the air

Speed

Median **< 7 s** in full SO(3); real slower than sim

Drop rate

Truck #5: **23%** · Duck #10: **56%** (sim ~20%)

Q6. One controller for every scenario?

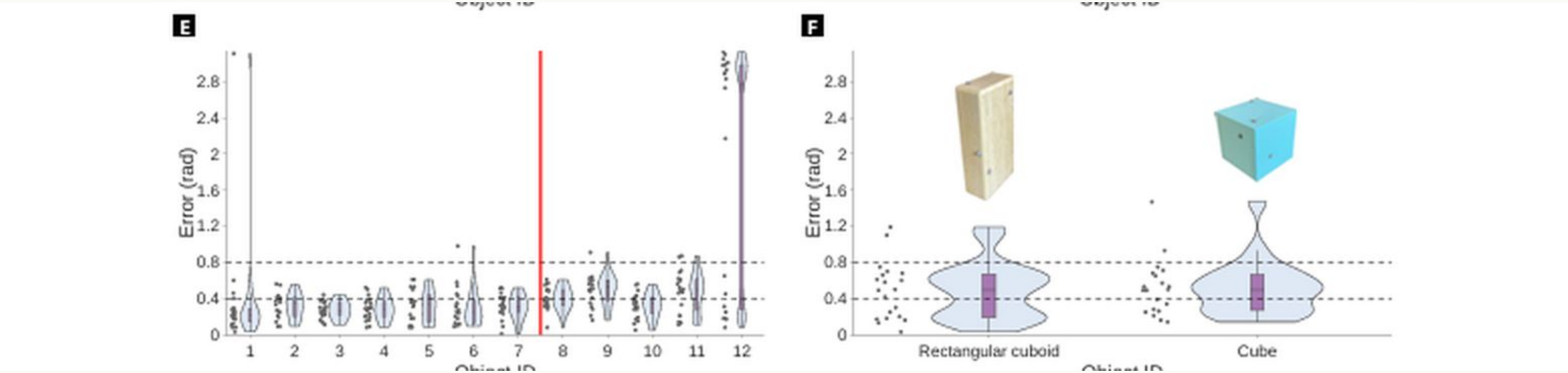


Fig. 4E on-table with reward r2 · 4F symmetric objects (never trained)

Recovery: drops onto a table → picks it up and continues

No trade-off: r2 keeps on-table error similar

Symmetric objects: some generalization

Not random: same start + goal → similar motion

Q7. Does it work on everyday objects?

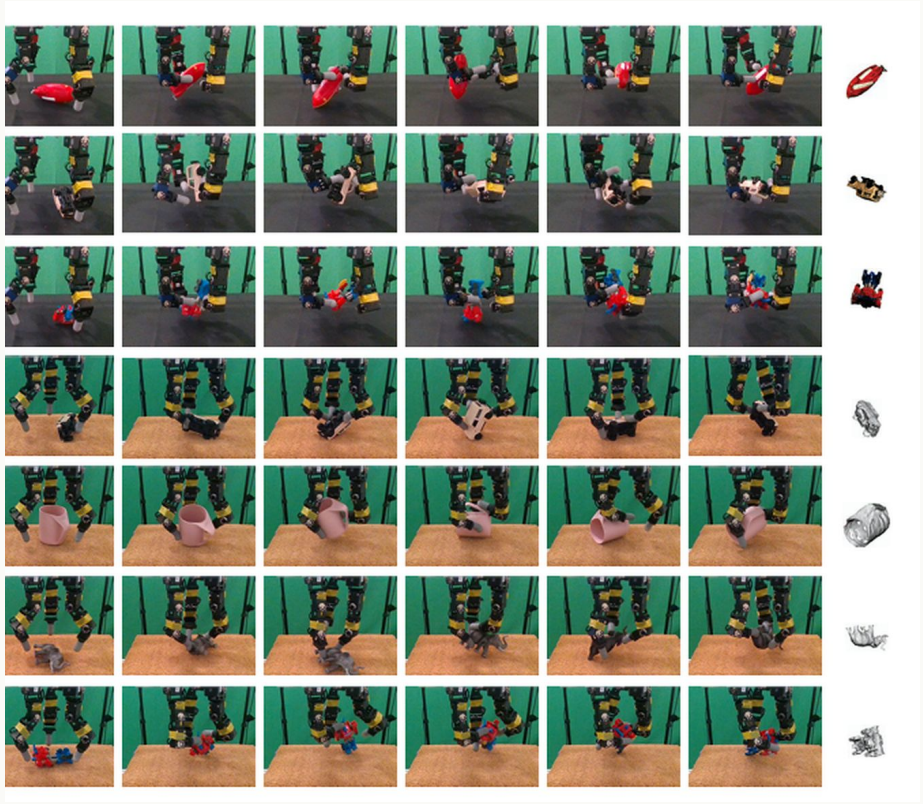


Fig. 5 — real objects, not 3D-printed. Right: scanned targets

- Toys, a mug and other rigid household objects
- Varying materials and non-uniform mass
- No CAD model → goal scanned with a free iPad app (**Scaniverse**)
- Imperfect scans → **noisy goal**, still works to a degree

Caveat: qualitative only — no mocap markers or ground-truth CAD for these objects

Q8. How does it compare with prior work?

A strictly fair comparison is not possible — fewer assumptions, different hardware and sensing. Closest work on the duck:

	Duck reorientation (23)	This work
Success	60% (3/5) within 0.1 rad, real	71% within 0.1 rad, sim, perfect stopping assumed
Speed	70 s+, quasi-static	~7 s, dynamic
Rotation	Two axes	Full SO(3)
Perception	Object-specific tracker (< 2° error)	Generic point cloud

vs. OpenAI: single cube, upward hand, 3 RGB cameras + mocap, success = passing the goal. This work must **stop and hold** — harder, and needed for tool use.

04

Discussion

What we learn, and what is still missing

Insights

Shape may matter less than expected

A shape-blind, memoryless teacher works in the real world, not only in simulation

Shape features could still improve precision and generalization

Over-parameterized hardware

More fingers = more variables, yet **easier** for deep RL — like over-parameterized networks

A conjecture; evidence is Fig. 4B only

Choices that transfer

Two-stage student, point-cloud inputs, sparse CNN, sys-ID, soft tips, more fingers

Apply to vision-based rigid-body manipulation broadly

Limitations and follow-ups

Limitation	Why (authors' hypothesis)	Follow-up
Dropping	12 Hz vision is too slow to catch a slip in time	Faster networks or more compute
Dropping	No contact, force or slip signal; contacts inferred from occluded vision	Add touch sensors
Imprecision	$\|\Delta\theta\|$ prediction poor below 0.4 rad → wrong stopping	Bigger object set; RGB + depth; vision + touch
Sim-to-real	A gap remains (duck: 20% sim vs 56% real drops)	Further study
Hardware	D'Claw finger design and count not optimal; soft tips made by hand, wear out	Computational hand design

Conclusion

1. A **shape-blind teacher** + a fast **two-stage point-cloud student** → one controller for new, complex objects in full SO(3)

2. **Sim-to-real works** for dynamic, contact-rich, downward-facing manipulation with one depth camera and <$5k open hardware

3. The contribution is **scope, not peak accuracy**: precision and dropping remain open — touch, faster control, better hands

Thank you

Q&A

Chen et al., “Visual Dexterity: In-Hand Reorientation of Novel and Complex Object Shapes,” Science Robotics, 2023

A

Appendix – Math details

Math ① RL objective and the discount factor

OBJECTIVE (SLIDE 16)

$$\pi_E^* = \arg\max_{\pi_E} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right]$$

Maximize the expected discounted return; the expectation averages over objects, initial poses and randomized physics.

WHY γ < 1 KEEPS THE RETURN FINITE

$$\left|\sum_{t=0}^{T-1} \gamma^t r_t\right| \leq r_{\max} \sum_{t=0}^{T-1} \gamma^t = r_{\max} \frac{1-\gamma^T}{1-\gamma} \leq \frac{r_{\max}}{1-\gamma}$$

Geometric series: the return is bounded by r□ax / (1 − γ), so about 1 / (1 − γ) future steps matter.

$$\gamma = 0.99 \Rightarrow \frac{1}{1-\gamma} = 100 \text{ steps}$$ Illustrative value; the paper does not report γ.

Math ② EMA is a weighted average of past actions

EMA (SLIDE 16)

$$\bar{a}_t = \alpha\, a_t + (1-\alpha)\, \bar{a}_{t-1}$$

UNROLL THE RECURSION

$$\bar{a}_t = \alpha \sum_{k=0}^{t-1} (1-\alpha)^k\, a_{t-k} + (1-\alpha)^t\, \bar{a}_0$$

WEIGHTS SUM TO 1 → A WEIGHTED AVERAGE

$$\alpha \sum_{k=0}^{t-1} (1-\alpha)^k + (1-\alpha)^t = \left(1-(1-\alpha)^t\right) + (1-\alpha)^t = 1$$

$$\alpha = 0.8: \quad 0.8, \quad 0.16, \quad 0.032, \quad 0.0064, \quad \ldots$$

Two steps back counts only **3.2%** → smooth, but little lag.

NEXT JOINT TARGET

$$q_{t+1}^{\text{tgt}} = q_t + \bar{a}_t$$

Relative command from the current angle → no large jumps.

Math ③ Reward r_1 in full, and why these shapes

$$r_t^1 = c_1\,\mathbf{1}(\text{success}) + \frac{c_2}{|\Delta\theta_t| + \epsilon_\theta} + c_3 \sum_{i=1}^{G} \left\| p_t^{f_i} - p_t^o \right\|_2^2$$

$$+ c_4\,|\dot{q}_t|^\top |\tau_t| + c_5\,\mathbf{1}\left(\|p_t^o\|_2^2 > \bar{p}\right), \qquad c_1, c_2 > 0, \quad c_3, c_4, c_5 < 0$$

1(·) = 1 if the condition holds, else 0

INVERSE VS. LINEAR (EQ. 2)

$$\frac{d}{dx}\frac{1}{x+\epsilon} = -\frac{1}{(x+\epsilon)^2} \qquad \text{vs.} \qquad \frac{d}{dx}(-x) = -1$$

Slope grows sharply as Δθ → 0 → strong pull near the goal. ε avoids division by zero and caps the reward at c_2 / ε.

Interpretations of the formulas; the paper does not derive them.

ENERGY TERM (EQ. 4)

$$|\dot{q}_t|^\top |\tau_t| = \sum_j |\dot{q}_{t,j}|\,|\tau_{t,j}| = \sum_j |\text{power}_j|$$

Mechanical power = torque × angular velocity; summed over time, it is energy.

Math ④ Reward r_2 and the Gaussian policy

$$r_t^2 = r_t^1 + c_6\,\mathbf{1}(\text{object touches table}) + c_7 \sum_{i=1}^{N} \mathbf{1}\left(p_{t,z}^{f_i} > \bar{p}_z\right), \quad c_6, c_7 < 0$$

z = fingertip height; above the threshold means pushing with the penultimate joint

WHY LIFTING EMERGES (SLIDE 19)

$$R_{\text{use table}} \approx R_{\text{lift}} - |c_6| \cdot n_{\text{contact}}$$

Each table contact costs $|c_6|$. A policy that lifts avoids it, so RL prefers lifting — if it can find it (4 fingers did, 3 did not).

GAUSSIAN POLICY (SLIDE 21)

$$a_t \sim \mathcal{N}(\mu, \sigma)$$

Sample around the mean μ to explore in training; Δθ head is a single regressed value.

Using μ at deployment is common practice, not stated in the paper.

Math ⑤ Synthetic point cloud: one matrix multiply per link

$$P_t^{l_j} = P^{l_j}\left(R_t^{l_j}\right)^\top + \left(p_t^{l_j}\right)^\top \qquad P_t^s = \bigcup_{j=1}^{M} P_t^{l_j}$$

P: K×3 mesh points of link j in its own frame; R, p: link rotation and position at time t

1 $x_{\text{world}} = R\,x + p$ — Rigid transform of one point x (column vector)

2 $x_{\text{world}}^\top = x^\top R^\top + p^\top$ — Transpose, using (AB)ᵀ = BᵀAᵀ — points are stored as rows

3 $P_{\text{world}} = P\,R^\top + \mathbf{1}_K\,p^\top, \qquad P \in \mathbb{R}^{K\times 3}$ — Stack K rows; add pᵀ to every row

Every link in every environment → GPU matrix ops in parallel: 7× faster than rendering

Math ⑥ System ID is a least-squares search

$$h = -\left\| q_{\text{real}}(\cdot) - q_{\lambda}(\cdot) \right\|_2^2 \qquad \lambda^* = \arg\max_{\lambda} h = \arg\min_{\lambda} \left\| q_{\text{real}} - q_{\lambda} \right\|_2^2$$

Same command sequence A(·) to the real and simulated robots; the error sums over all time steps and joints. Maximizing h = minimizing squared error.

NO GRADIENTS (CLOSED-SOURCE SIMULATOR) → CMA-ES LOOP

1

Sample candidate λ from a search distribution N(m, C)

2

Score each with h — in parallel on Isaac Gym

3

Move the mean m toward the best; reshape the covariance C

Math ⑦ Stopping rule and orientation error

STOPPING RULE (SLIDE 27)

$$\text{stop} \iff \Delta\theta_t^{\text{pred}} < \bar{\theta} \quad \text{and} \quad \|a_t\| < \bar{a}$$

$$\|a_t\| = \sqrt{\sum_{j=1}^{3G} a_{t,j}^2}$$

Distance alone could stop mid-swing; “almost there” and “barely moving” together filter that out.

ORIENTATION ERROR (SLIDE 29)

$$0.4 \text{ rad} \times \frac{180^\circ}{\pi} \approx 22.9^\circ, \qquad 0.8 \text{ rad} \approx 45.8^\circ$$

$$R_{\text{rel}} = R_{\text{goal}}\, R_{\text{cur}}^\top, \qquad \Delta\theta = \arccos\left(\frac{\text{tr}(R_{\text{rel}}) - 1}{2}\right)$$

$$\text{tr}\, R_z(\theta) = \cos\theta + \cos\theta + 1 = 1 + 2\cos\theta$$

Any rotation by θ about any axis has trace 1 + 2 cos θ (trace does not depend on the frame) → solve for θ.

Standard definition; the paper only says “distance between orientations”.