怎么还不放寒假啊

Weiwen Chen 10.08

Outline

- state representation learning
- Improving Data Efficiency
- Vision and Touch
- Interactive Open-Ended

state representation learning

state representation learning

ABSTRACT

Scaling end-to-end reinforcement learning to control real robots from vision presents a series of challenges, in particular in terms of sample efficiency. Against end-to-end learning, state representation learning can help learn a compact, efficient and relevant representation of states that speeds up policy learning, reducing the number of samples needed, and that is easier to interpret. We evaluate several state representation learning methods on goal based robotics tasks and propose a new unsupervised model that stacks representations and combines strengths of several of these approaches. This method encodes all the relevant features, performs on par or better than end-to-end learning with better sample efficiency, and is robust to hyper-parameters change.

in our case, they do not encode the position of the target since the agent cannot act on it.

Since learning to extract the robot position is not enough to solve goal-based tasks, we need to add extra objective functions in order to encode the position of the target object. In this section, we consider two of them: minimizing a reconstruction error (auto-encoder model) or a reward prediction loss.

- *Auto-encoder:* Thanks to their reconstruction objective, auto-encoders compress information in their latent space. Auto-encoders tend to encode only aspects of the environment that are salient in the input image. This means they are not task-specific: relevant elements can be ignored and distractors (unnecessary information) can be encoded into the state representation. They usually need more dimensions that apparently required to encode a scene (e.g. in our experiments, it requires more than 10 dimensions to encode correctly a 2D position).
- *Reward prediction:* The objective of a reward prediction module leads to state representations that are specialized in a task. However, this does not constrain the state space to be disentangled or to have any particular structure.

Combining objectives makes it possible to share the strengths of each model. In our application example, the previous sections suggest that we should mix objectives to encode both robot and target positions. The simplest way to combine objectives is to minimize a weighted sum of the different loss functions, i.e. reconstruction, inverse dynamics and reward prediction losses, i.e.:

$$\mathcal{L}_{combination} = w_{reconstruction} \cdot \mathcal{L}_{reconstruction} + w_{inverse} \cdot \mathcal{L}_{inverse} + w_{reward} \cdot \mathcal{L}_{reward}$$

Each weight represents the relative importance we give to the different objectives. Because we consider each objective to be relevant, we chose the weights such that they provide gradients with similar magnitudes.

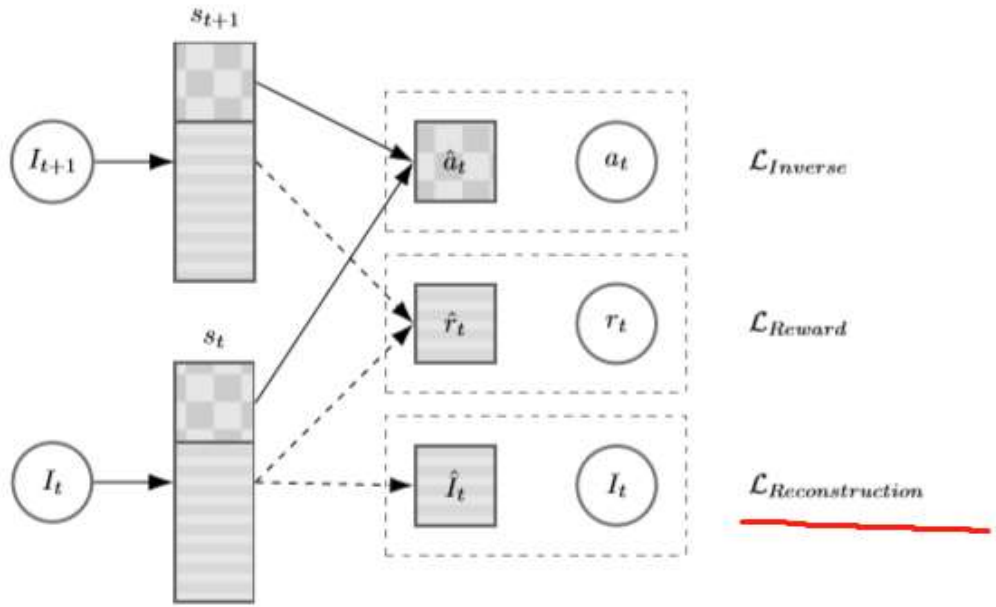


Figure 1: *SRL Splits* model: combines a reconstruction of an image I, a reward (r) prediction and an inverse dynamic models losses, using two splits of the state representation s. Arrows represent model learning and inference, dashed frames represent losses computation, rectangles are state representations, circles are real observed data, and squares are model predictions.

4 EXPERIMENTS AND RESULTS

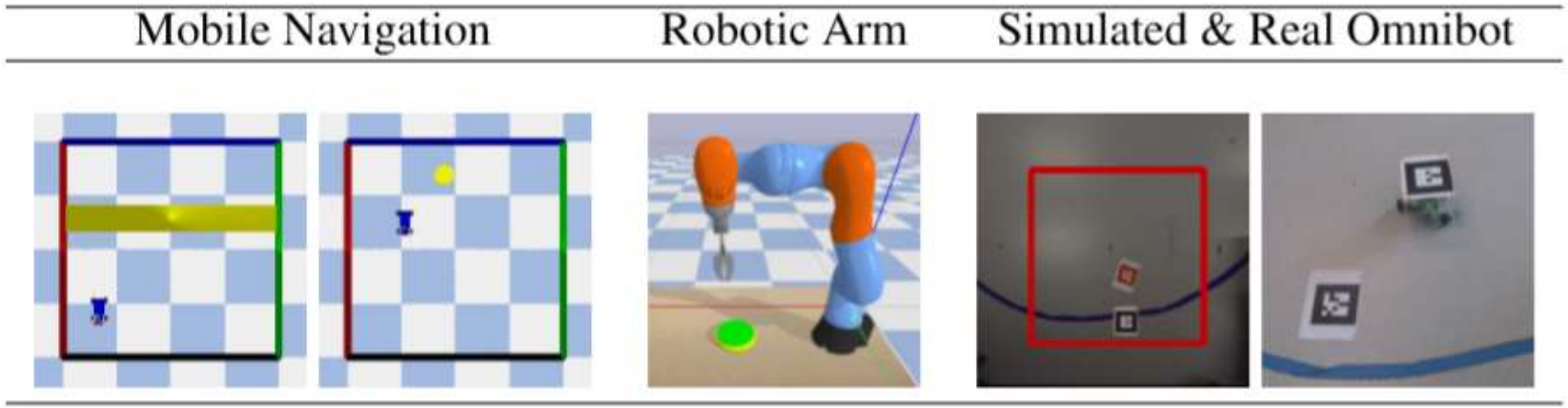


Figure 2: Environments for state representation learning from S-RL toolbox (Raffin et al., 2018) with extensions (*2D Simulated + Real Omnibot*).

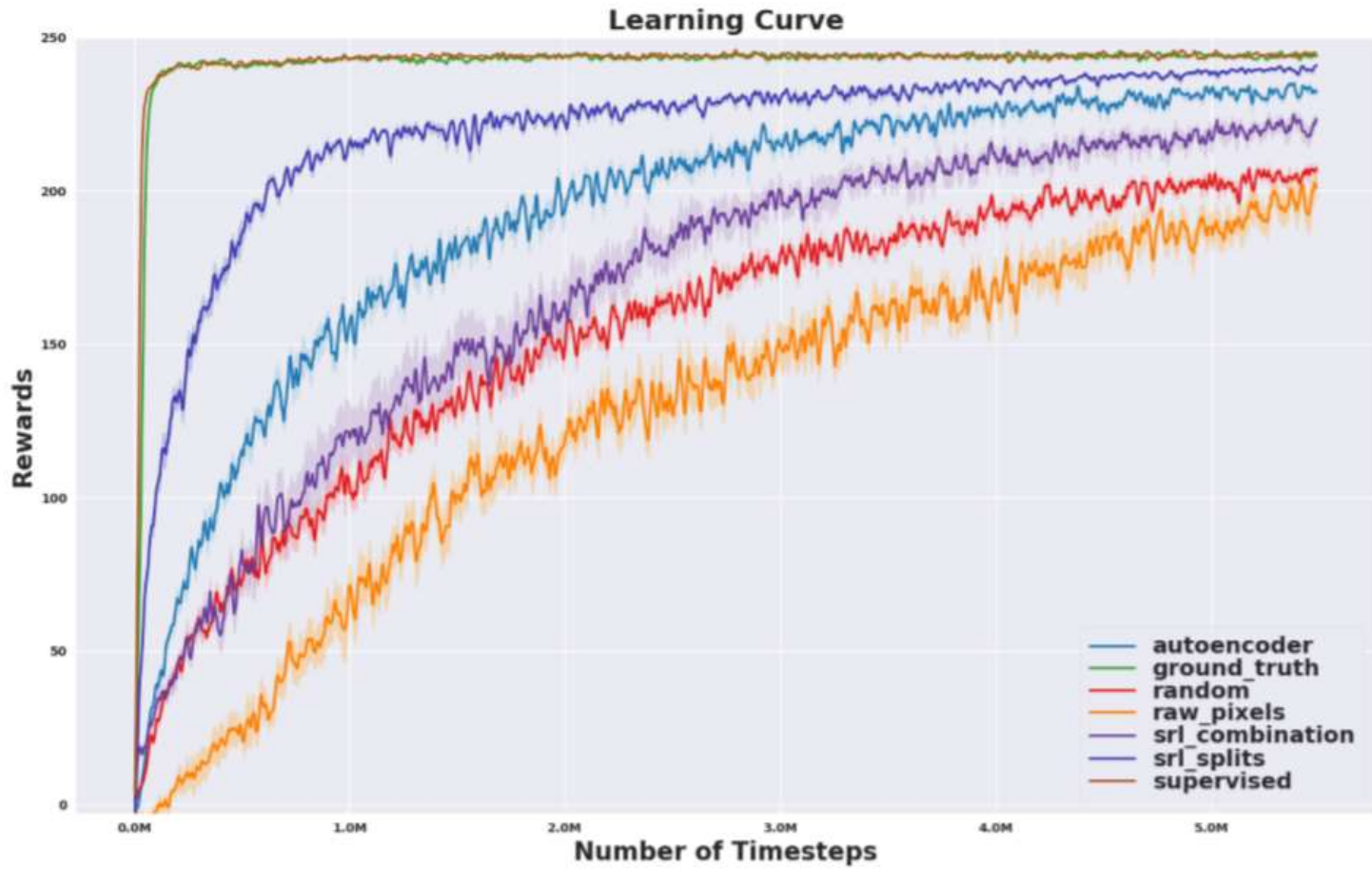


Figure 3: Performance (mean and standard error for 8 runs) for PPO algorithm for different state representations learned in Simulated OmniRobot with randomly initialized target environment.

Improving Data Efficiency

Improving Data Efficiency

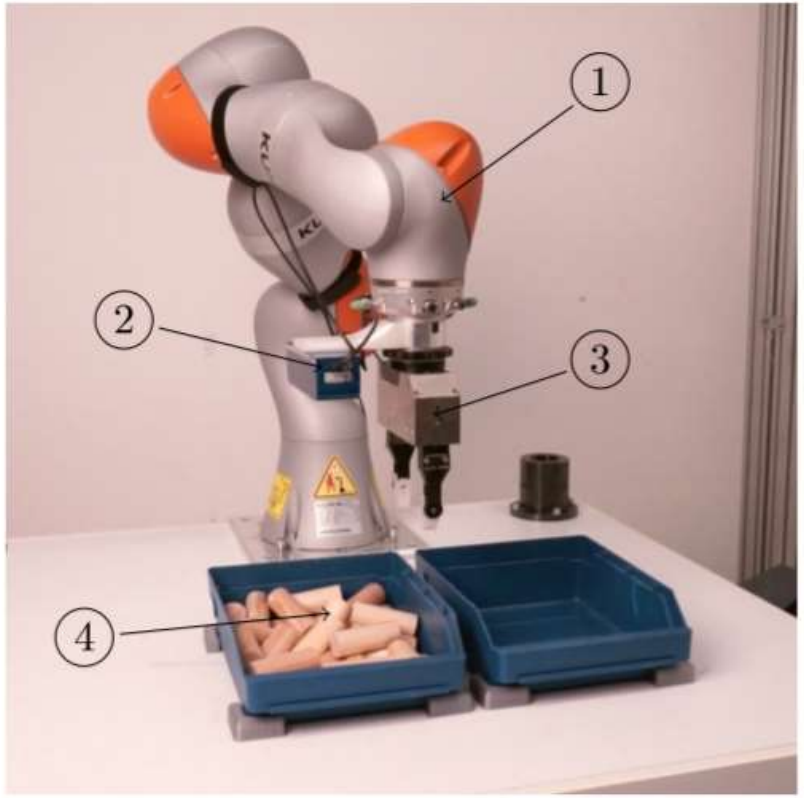
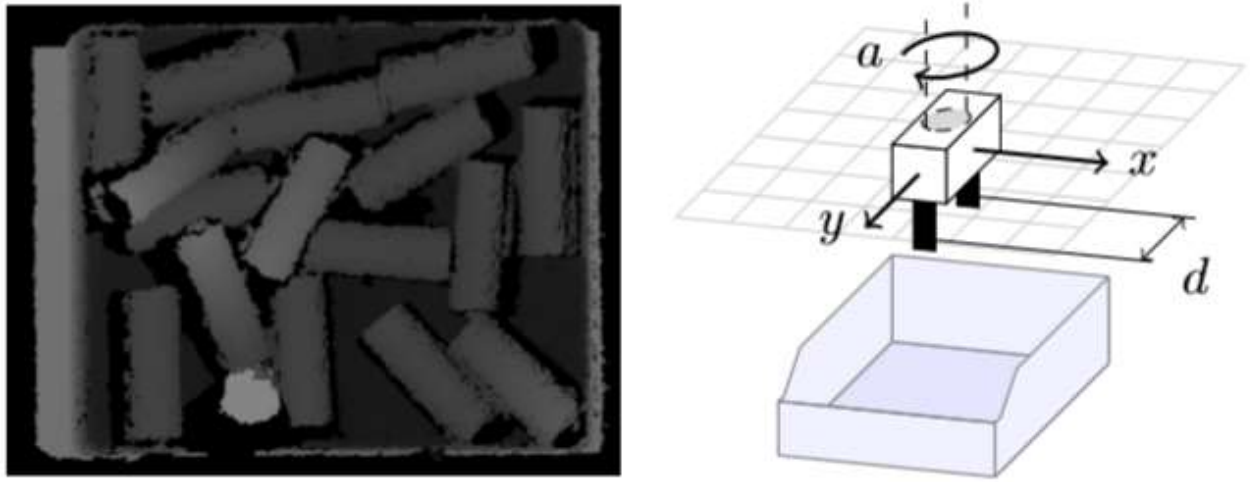


Fig. 1: Our setup consists of a KUKA LBR iiwa robot (1) with a flange-mounted stereo camera (2), a force-feedback gripper (3), and two industrial storage bins (4).

Improving Data Efficiency



(a) Input depth image (b) Output parameters

Fig. 2: Our algorithm maps an undistorted depth image (a) to four parameters (x, y, a, d) in the gripper-task space (b). The image ranges from near (white) to far (black), missing depth information are displayed black. The image plane is aligned with the x-y-plane of the task space, which is a crucial requirement for our algorithm.

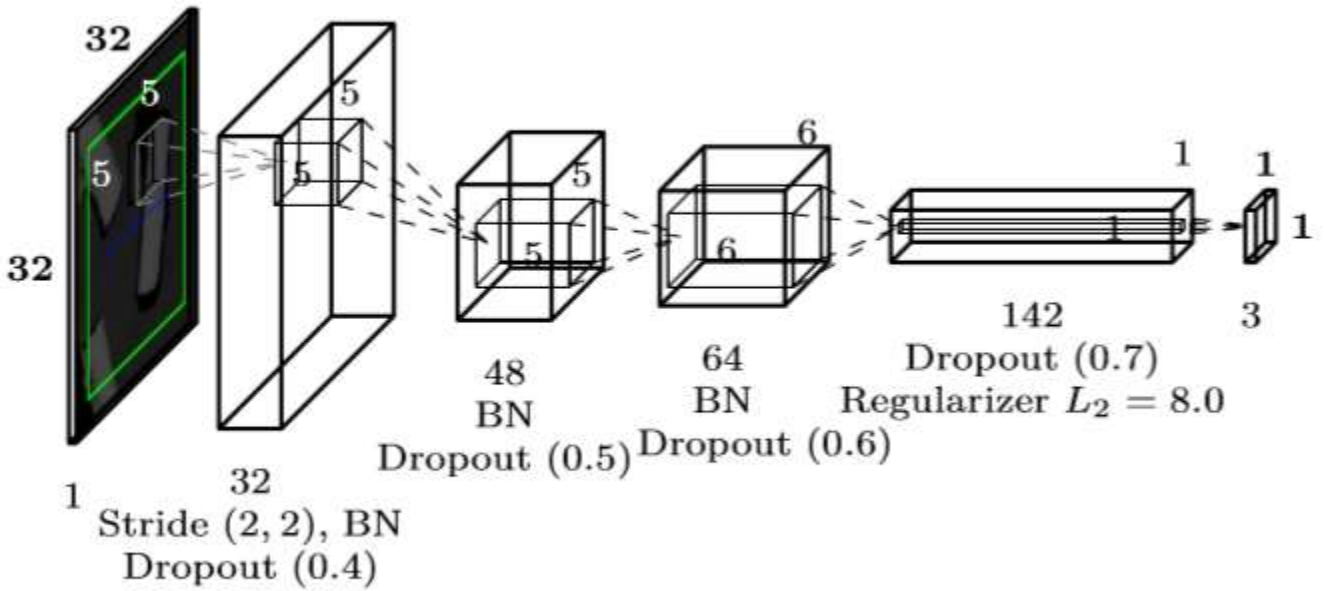
Improving Data Efficiency

A. Single-Shot Grasps

The grasping pose is defined as the position (x, y, z) and Euler angles (a, b, c) (with a around the z-axis) where the gripper is closed. Given a calculated grasping pose, the natural process of a single-shot grasp attempt is:

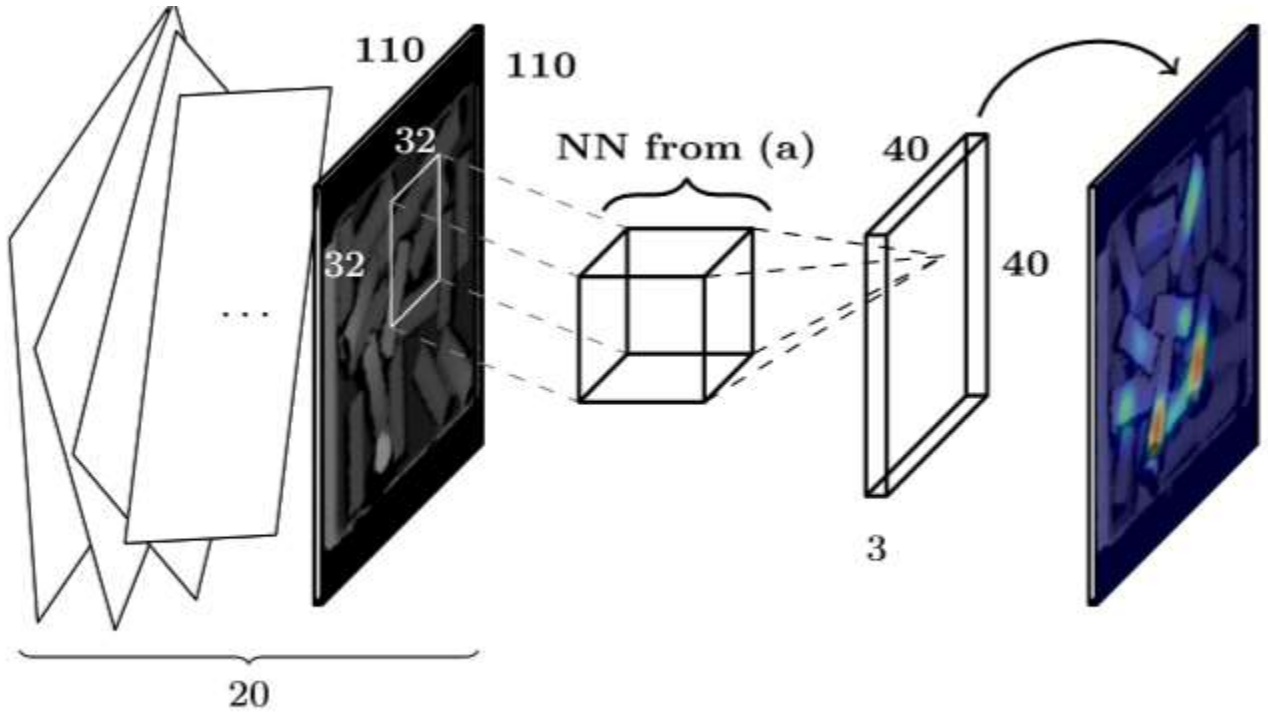
1) The gripper jaw distance is set to d.
2) The gripper approaches the grasping pose with a movement parallel to its fingers. If the joint torque sensors detect a collision, the gripper retracts a few mm and continues the grasping process.
3) The gripper closes and tries to clamp an object with a predefined force f.
4) The robot lifts the object and moves to a filing position. The reward r is set to 1 for a successful grasp, otherwise to 0.

Neural Network



(a) **Training**

Neural Network



(b) **Inference**

Grasp Space Exploration

D. Grasp Space Exploration

Random grasps result in a success rate of only 3 %. This motivates our systematic approach for a data-efficient self-supervised exploration. Therefore, learned information is used to improve learning itself. As this RL problem is simplified towards a supervised learning problem, the task of exploration resembles active learning. First of all, images of successful grasps contain more valuable information, as they are harder to collect. Secondary, a grasp attempt is valuable for learning if the predicted grasp probability differs from the measured reward, leading to grasps a given by $\arg\max_a |r - \psi(s, a)|$. To fulfill both conditions, exploration is implemented by choosing one of the following selection functions σ:

Grasp Space Exploration

choosing one of the following selection functions σ:

a) Random: A uniformly random selection function can be given as $\sigma = \hat{a} \sim P(a) = |\mathcal{A}|^{-1}$. Only this method is guaranteed to fully span the grasp space $\mathcal{A}$. Random grasp rates were measured using this method.

b) Maximum(N): A more general form of the greedy action $\sigma(\psi) = \arg\max_a \psi(s, a)$ is to choose one of the N highest-ranked poses randomly. In application, this method with $N = 1$ is first applied. If the grasp fails, a grasp attempt with $N = 5$ is repeated. This method is

Grasp Space Exploration

used for exploitation.

c) Probabilistic: By choosing a grasp according to the given and normalized probability $\sigma(\psi) = \hat{a} \sim \psi$, the overall measure of probability improves. This method increases the grasping rate but keeps exploration active and is used largely during training.

d) Uncertain: The selection function for uncertain grasps with the nearest probability to 0.5 is given by $\sigma(\psi) = \arg\min_a |0.5 - \psi(s, a)|$. Those grasps contain valuable information, because the difference between the grasp probability and the measured reward is near 0.5.

E. Weighted Retraining

Weighted retraining aims to minimize the impact of two error types. Firstly errors from defective measurements, e.g. caused by moving objects, and secondly the sampling error known in RL. In the context of a MDP, the grasping success is always probabilistic. This is in contrast to the binary value of r and reinforces the sampling error. Let w_i be the weight of a grasp attempt during training and N be the total dataset size. The NN is at first trained with $w_i = 1$. Then, the NN predicts the reward ψ and adapts the weights according to

$$w_i = \frac{N\left(1 - |r_i - \psi(s_i, a_i)|\right)}{\sum_j \left(1 - |r_j - \psi(s_j, a_j)|\right)}.$$

The normalized difference between measured and predicted reward is used as a quality measure of the input data, and the NN is retrained with a weighted training set.

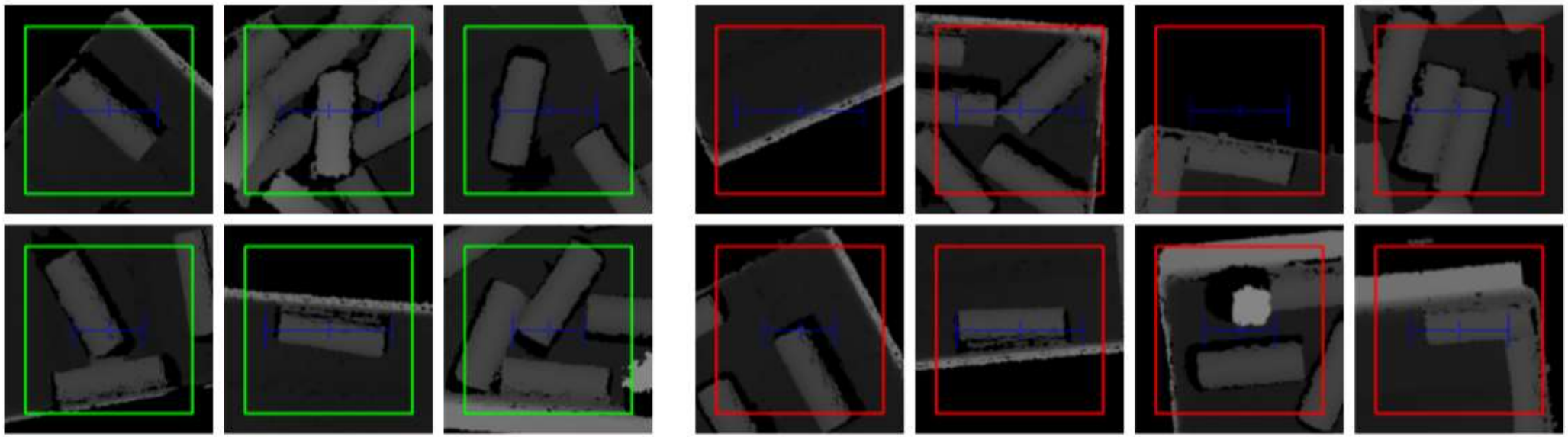


Fig. 4: Examples of cropped window depth images. Successful (left, green) and failed (right, red) grasps as well as the gripper position (blue) are marked.

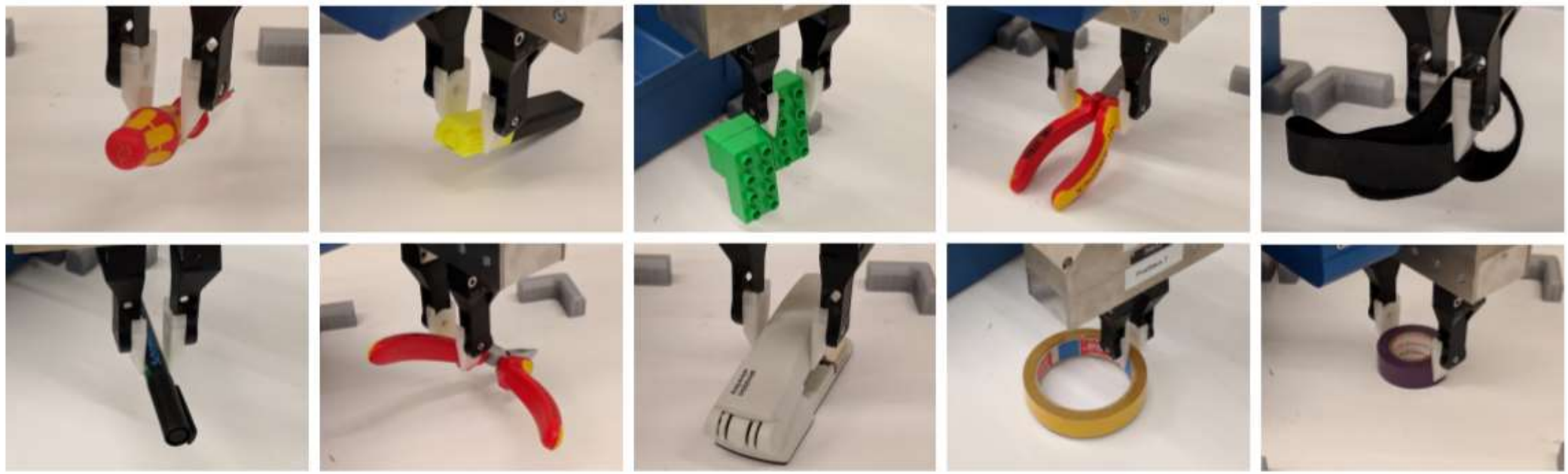


Fig. 5: Our robot generalizes to unseen object types. Although it was trained just with cylinders, grasps of similar (left) to more differing (to the right) objects work reliably.

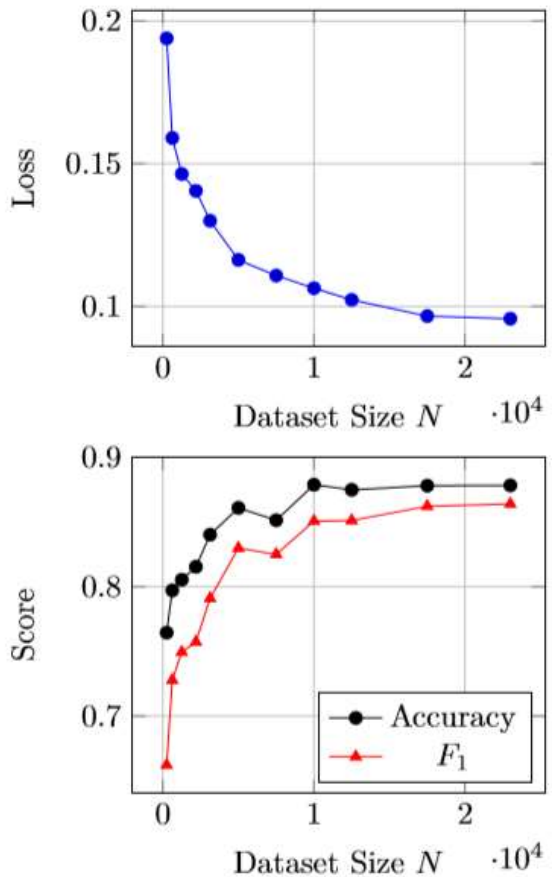


Fig. 6: Loss, accuracy and F_1-score of the test set depending on the dataset size N.

attempts. For typical bin picking scenarios, a grasp rate of $(96.6 \pm 1.0)\,\%$ was measured (Table I). Human performance is around 600 picks per hour (PPH) [7], exceeding our robot system by 40 %. As expected, the grasp rate improves with decreasing clutter and increasing grasp choices.

TABLE I: The measured grasp rate for grasping n objects without replacement out of a bin initially containing m objects. The random grasp rate is $\approx 3\,\%$.

n out of m	Grasp Rate	PPH	Number Samples
1 out of 1	$(99.8 \pm 0.2)\,\%$	367 ± 2	460
5 out of 5	$(99.3 \pm 0.7)\,\%$	418 ± 8	146
5 out of 10	$(99.5 \pm 0.5)\,\%$	428 ± 3	181
10 out of 20	$(96.6 \pm 1.0)\,\%$	425 ± 9	207

Weighted retraining of the NN improves the grasp rate (10 out of 20) by around 1 % from $(95.2 \pm 2.8)\,\%$ to $(96.6 \pm 1.0)\,\%$. The main effect of weighted retraining is the reinforcement of reliable grasps. As a side effect, this decreases the probability of making the same grasp mistake again, leading to a reduced uncertainty.

C. Generalization

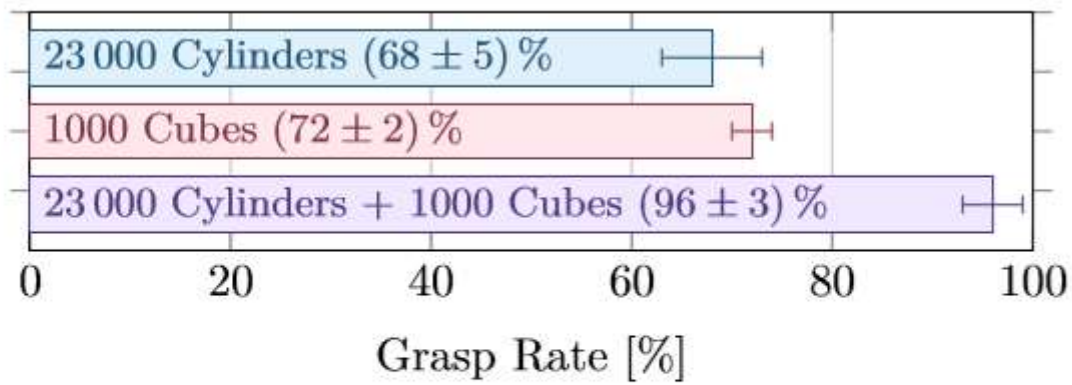
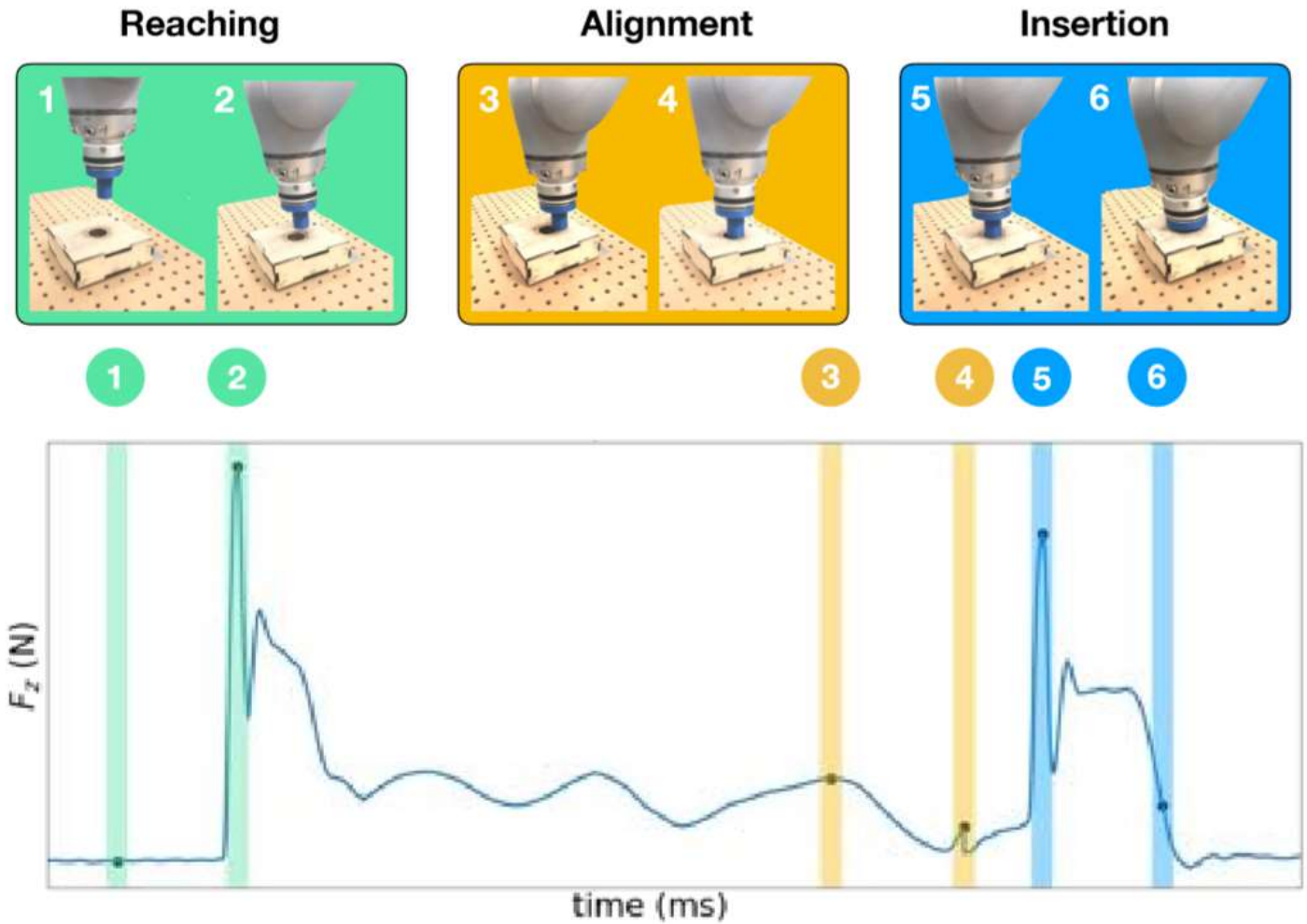


Fig. 7: The grasp rate of cubes (10 out of 15 objects), depending on the trained object type (label) and their corresponding dataset sizes N.

Vision and Touch

***Abstract*— Contact-rich manipulation tasks in unstructured environments often require both haptic and visual feedback. However, it is non-trivial to manually design a robot controller that combines modalities with very different characteristics. While deep reinforcement learning has shown success in learning control policies for high-dimensional inputs, these algorithms are generally intractable to deploy on real robots due to sample complexity. We use self-supervision to learn a compact and multimodal representation of our sensory inputs, which can then be used to improve the sample efficiency of our policy learning. We evaluate our method on a peg insertion task, generalizing over different geometry, configurations, and clearances, while being robust to external perturbations. We present results in simulation and on a real robot.**


Reaching
Alignment
Insertion
1
2
3
4
5
6
1
2
3
4
5
6
F_z (N)
time (ms)

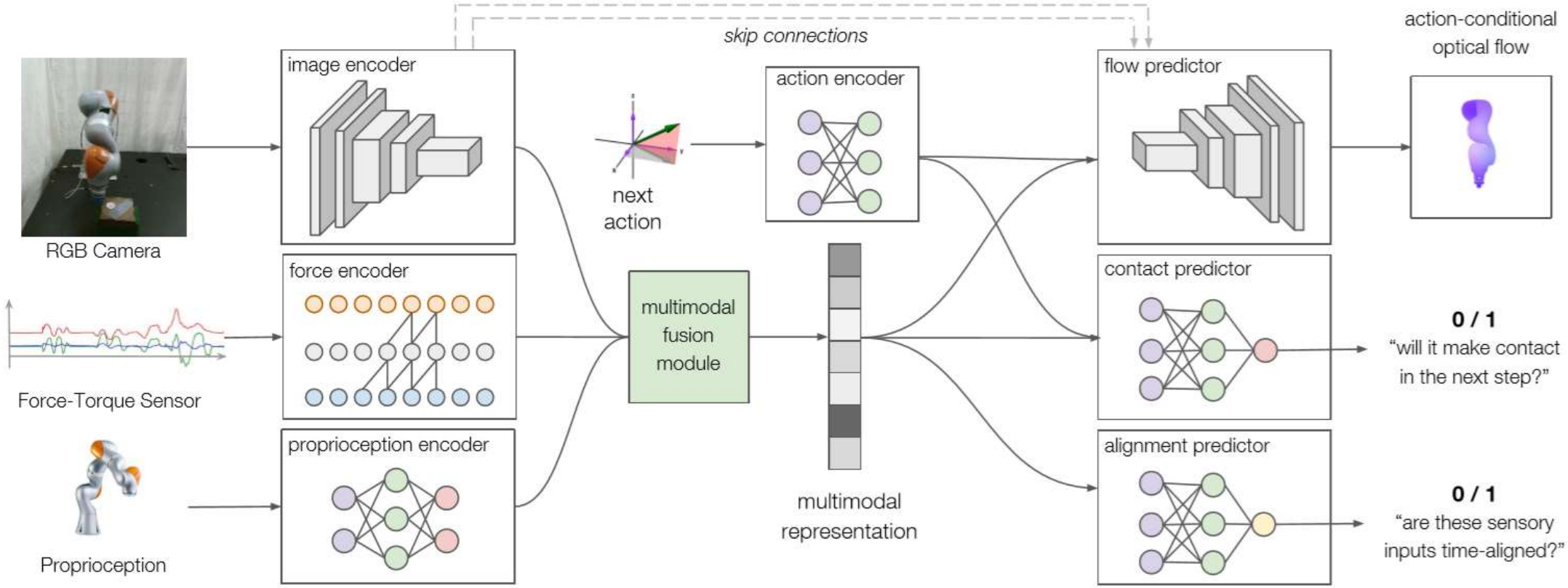


Fig. 2: Neural network architecture for multimodal representation learning with self-supervision. The network takes data from three different sensors as input: RGB images, F/T readings over a 32ms window, and end-effector position and velocity. It encodes and fuses this data into a multimodal representation based on which controllers for contact-rich manipulation can be learned. This representation learning network is trained end-to-end through self-supervision.

A. *Modality Encoders*

Our model encodes three types of sensory data available to the robot: RGB images from a fixed camera, haptic feedback from a wrist-mounted force-torque (F/T) sensor, and proprioceptive data from the joint encoders of the robot arm. The heterogeneous nature of this data requires domain-specific encoders to capture the unique characteristics of each modality. For visual feedback, we use a 6-layer *convolutional neural network* (CNN) similar to FlowNet [22] to encode $128 \times 128 \times 3$ RGB images. We add a fully-connected layer to transform the final activation maps into a 128-d feature vector. For haptic feedback, we take the last 32 readings from the six-axis F/T sensor as a 32×6 time series and perform 5-layer causal convolutions [39] with stride 2 to transform the force readings into a 64-d feature vector. For proprioception, we encode the current position and velocity of the end-effector with a 2-layer *multilayer perceptron* (MLP) to produce a 32-d feature vector. The resulting three

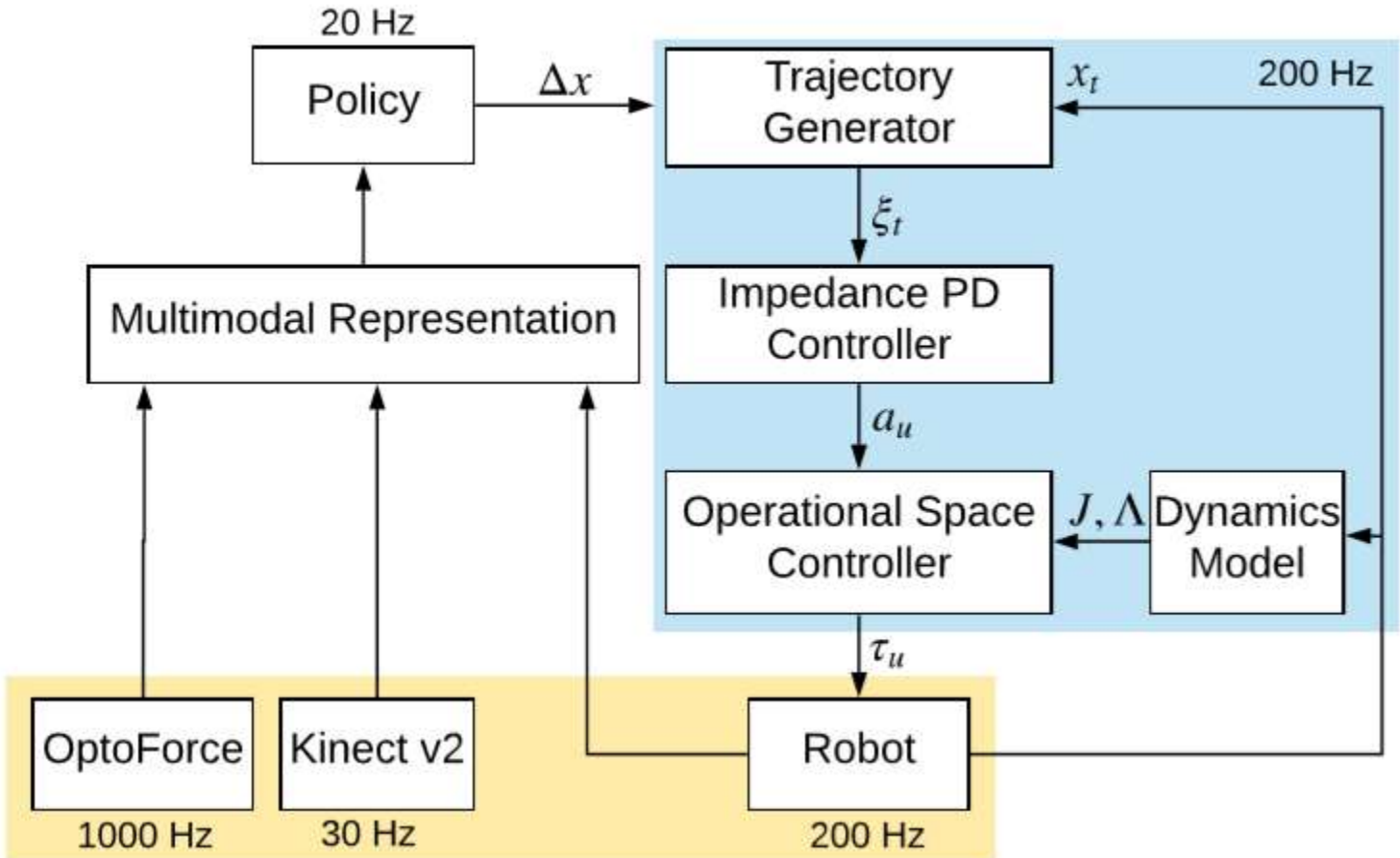
We model the manipulation task as a finite-horizo
counted Markov Decision Process (MDP) $\mathcal{M}$, with
space $\mathcal{S}$, an action space $\mathcal{A}$, state transition dynami
$\mathcal{S} \times \mathcal{A} \to \mathcal{S}$, an initial state distribution ρ_0, a reward fu
$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, horizon T, and discount factor $\gamma \in (0,$
determine the optimal stochastic policy $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A}$
want to maximize the expected discounted reward

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T-1} \gamma r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

We represent the policy by a neural network with parameters θ_π that are learned as described in Sec. V. $\mathcal{S}$ is defined

Policy Learning. Modeling contact interactions and multi-contact planning still result in complex optimization problems [42, 43, 52] that remain sensitive to inaccurate actuation and state estimation. We formulate contact-rich manipulation as a model-free reinforcement learning problem to investigate its performance when relying on multimodal feedback and when acting under uncertainty in geometry, clearance and configuration. By choosing model-free, we also eliminate the need for an accurate dynamics model, which is typically difficult to obtain in the presence of rich contacts. Specifically, we choose trust-region policy optimization (TRPO) [46]. TRPO imposes a bound of KL-divergence for each policy update by solving a constrained optimization problem, which

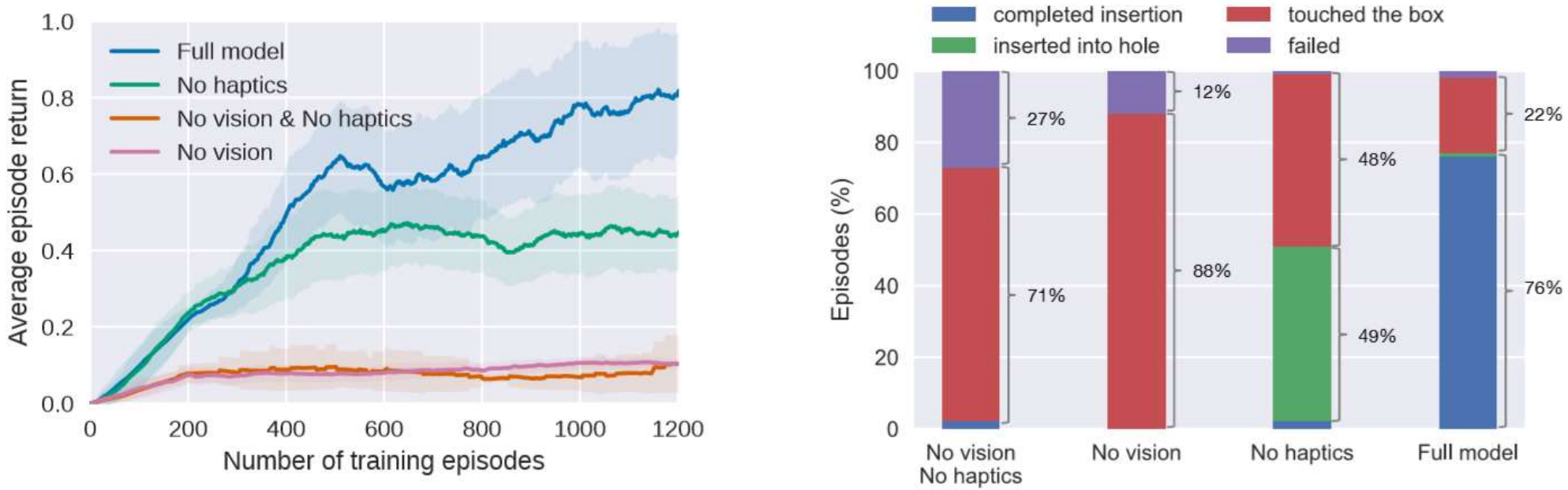
nal optical flow with endpoint
ver all pixels [22], and both
ne alignment prediction with
cross-entropy loss. During training, we minimize a sum of the three losses end-to-end with stochastic gradient descent on a dataset of rolled-out trajectories. Once trained, this network produces a 128-d feature vector that compactly represents multimodal data. This vector from the input to


20 Hz
Policy
Δx
Trajectory Generator
x_t
200 Hz
ξ_t
Impedance PD Controller
Multimodal Representation
a_u
Operational Space Controller
J, Λ
Dynamics Model
τ_u
OptoForce
Kinect v2
Robot
1000 Hz
30 Hz
200 Hz

Reward Design. We use the following staged reward function to guide the reinforcement learning algorithm through the different sub-tasks, simplifying the challenge of exploration and improving learning efficiency:

$$r(\mathbf{s}) = \begin{cases} c_r - \frac{c_r}{2}(\tanh \lambda \|\mathbf{s}\| + \tanh \lambda \|\mathbf{s}_{xy}\|) & \text{(reaching)} \\ 2 - c_a \|\mathbf{s}_{xy}\|_2 & \text{if } \|\mathbf{s}_{xy}\|_2 \leq \varepsilon_1 \quad \text{(alignment)} \\ 4 - 2(\frac{s_z}{h_d - \varepsilon_2}) & \text{if } s_z < 0 \quad \text{(insertion)} \\ 10 & \text{if } h_d - |s_z| \leq \varepsilon_2 \quad \text{(completion)}, \end{cases}$$

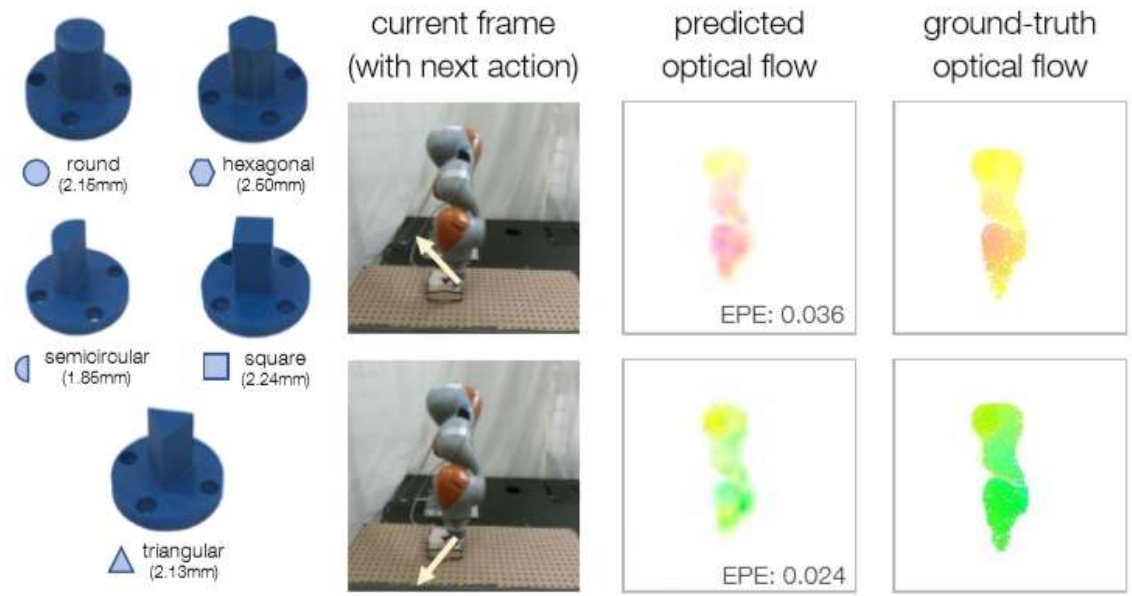
where $\mathbf{s} = (s_x, s_y, s_z)$ and $\mathbf{s}_{xy} = (s_x, s_y)$ use the peg's current position, λ is a constant factor to scale the input to the tanh function. The target peg position is $(0, 0, -h_d)$ with h_d as the height of the hole, and c_r and c_a are constant scale factors.



(a) Training curves of reinforcement learning

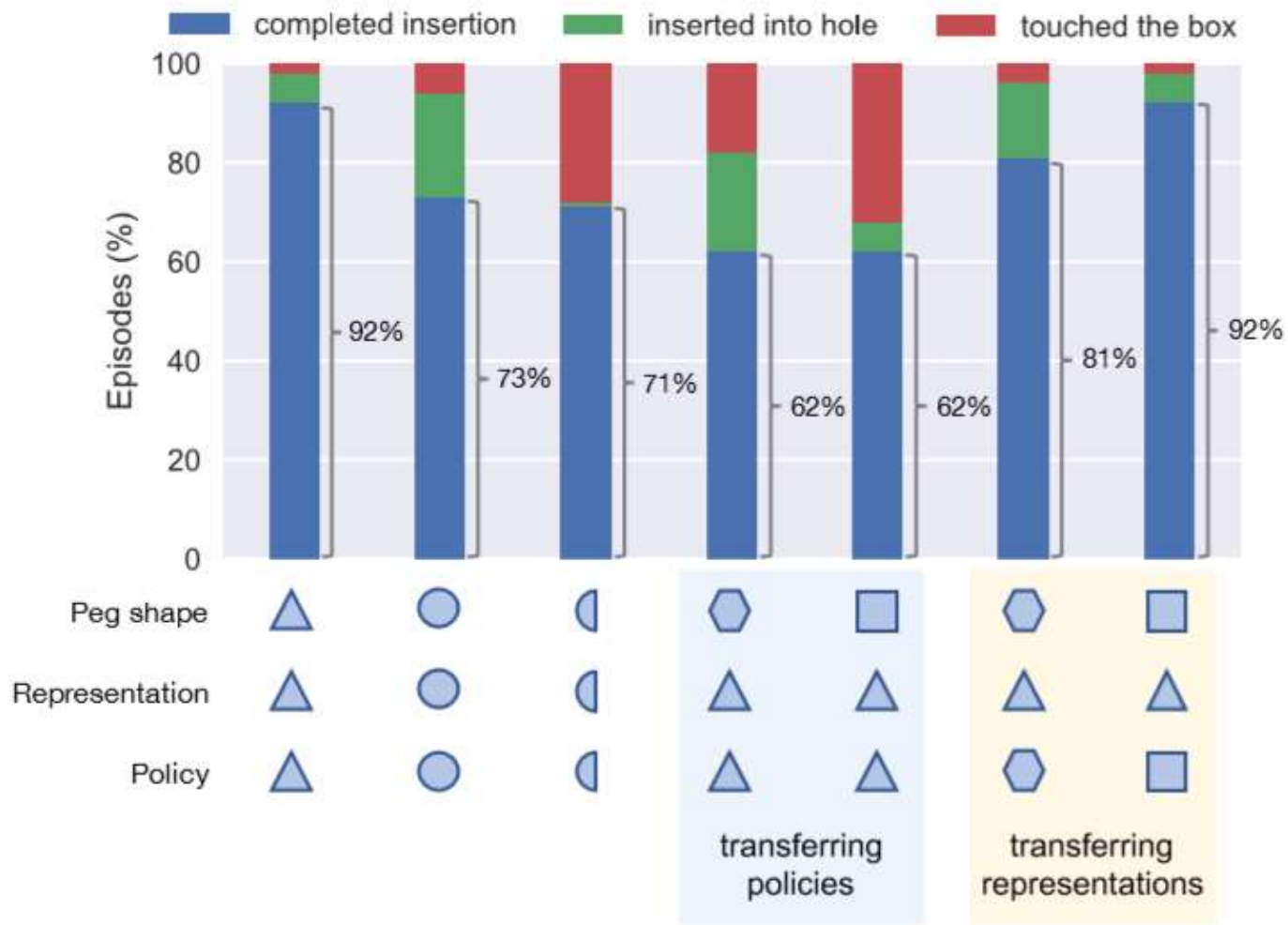
(b) Policy evaluation statistics

Fig. 4: Simulated Peg Insertion: Ablative study of representations trained on different combinations of sensory modalities. We compare our full model, trained with a combination of visual and haptic feedback and proprioception, with baselines that are trained without vision, or haptics, or either. (b) The graph shows partial task completion rates with different feedback modalities, and we note that both the visual and haptic modalities play an integral role for contact-rich tasks.



(a) Peg variations (b) Optical flow prediction examples

Fig. 5: (a) 3D printed pegs used in the real robot experiments and their box clearances. (b) Qualitative predictions: We visualize examples of optical flow predictions from our representation model (using color scheme in [22]). The model predicts different flow maps on the same image conditioned on different next actions indicated by projected arrows.


completed insertion
inserted into hole
touched the box
Episodes (%)
100
80
60
40
20
0
92%
73%
71%
62%
62%
81%
92%
Peg shape
Representation
Policy
transferring policies
transferring representations

Interactive Open-Ended

Interactive Open-Ended

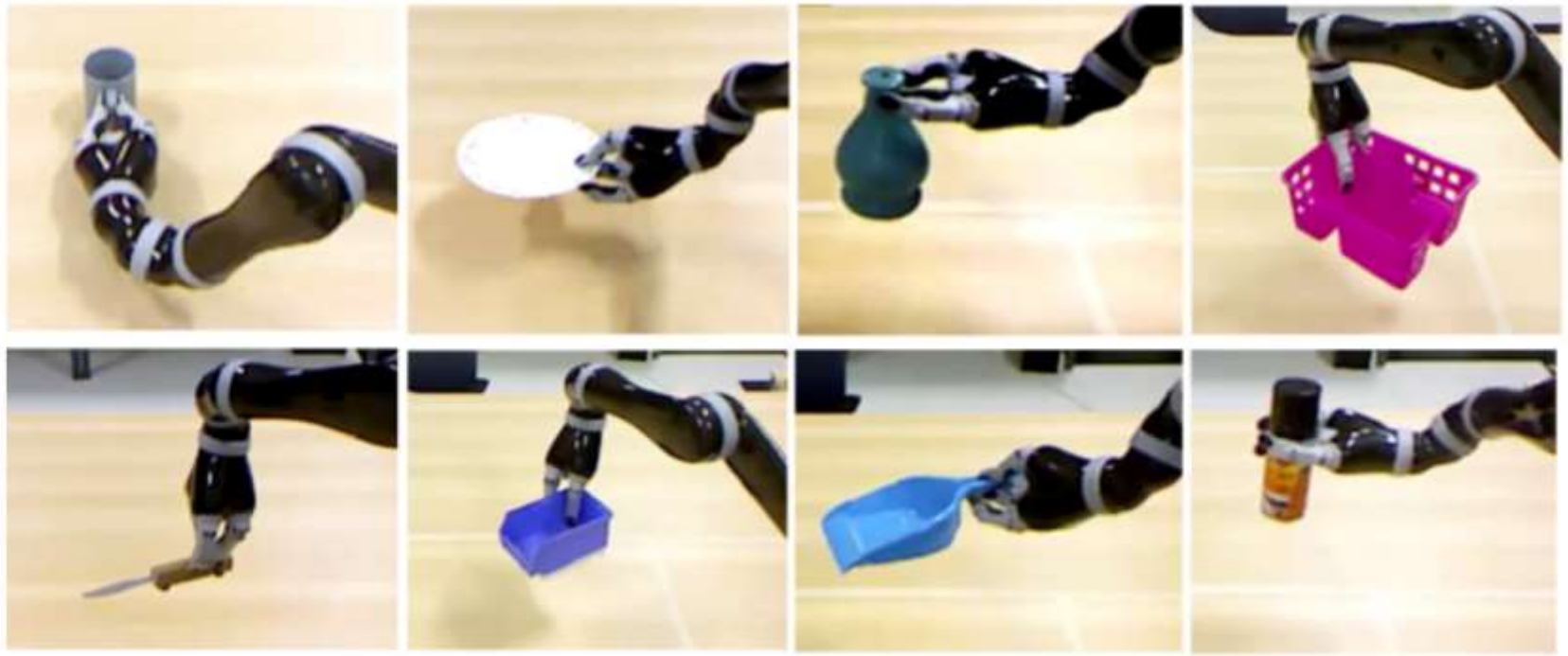


Fig. 1. Eight examples of affordance detection results: given the partial point cloud of an object, we simultaneously detect the object category label, pose, and its grasp affordance.

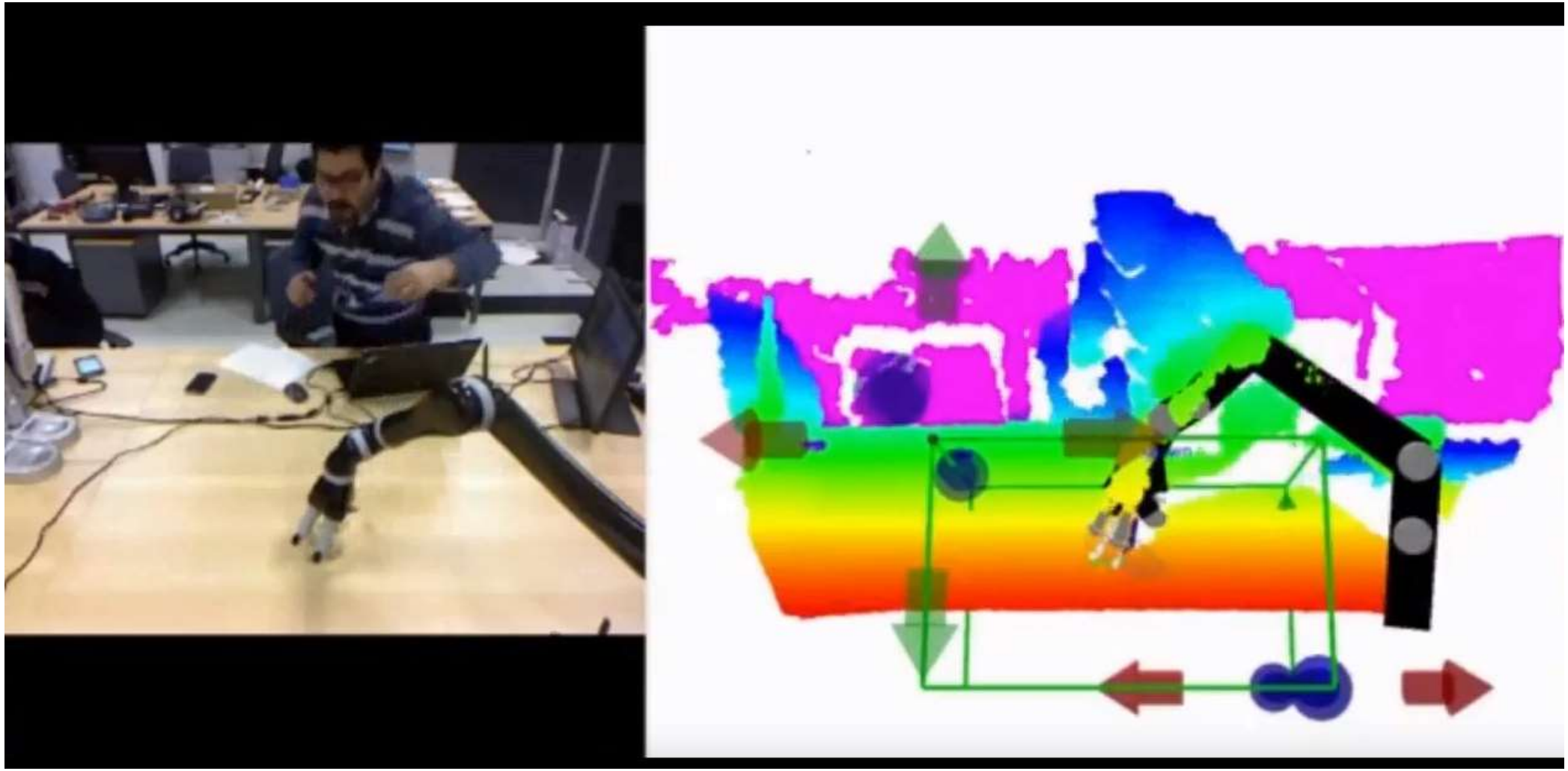
of interaction, the interface allows the user to perform the following actions:

- **Select:** point to the target object or select its *TrackID* from a menu.
- **Teach-category:** teach the object category or the affordance category of the selected object (each stable pose of an object on the table may map to a different affordance category).
- **Ask-category:** inquire the category or the affordance of the target object, which the agent will predict based on previously learned knowledge.
- **Correct-category:** if the agent could not recognize a given object or its affordance correctly, the user can teach the correct one.
- **Teach-grasp:** using kenesthetic teaching, teach a grasp configuration of the robotic arm to grasp the target object.
- **Grasp:** command the robot to grasp the target object.

Interactive Open-Ended

Fig. 3. Kinesthetic teaching: (*left*) the teacher interacts with the robot by moving the robot's gripper to a proper position; (*right*) Then, the teacher demonstrates a feasible grasp for the Pentomino object to the robot.

Interactive Open-Ended



Interactive Open-Ended

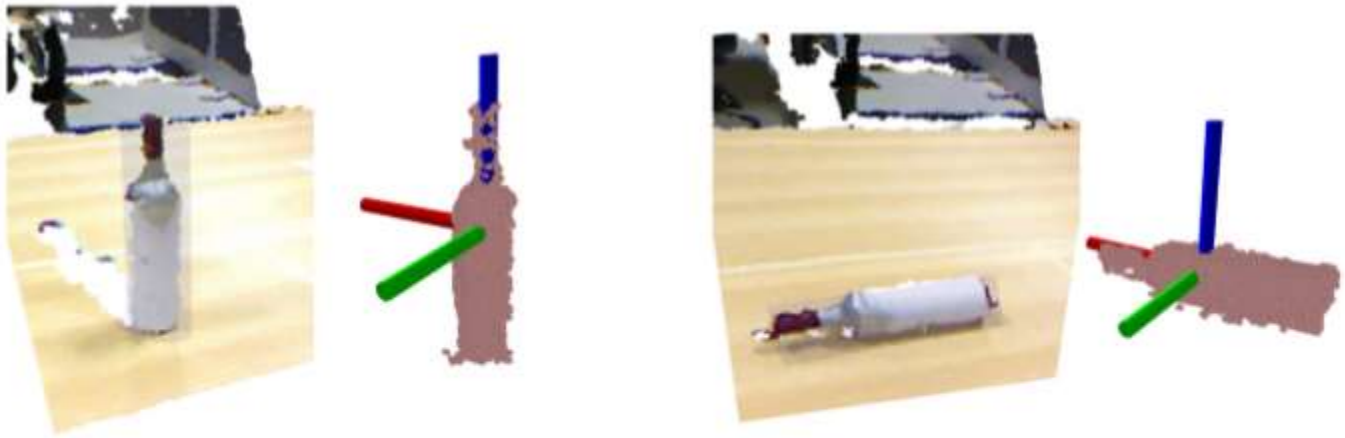


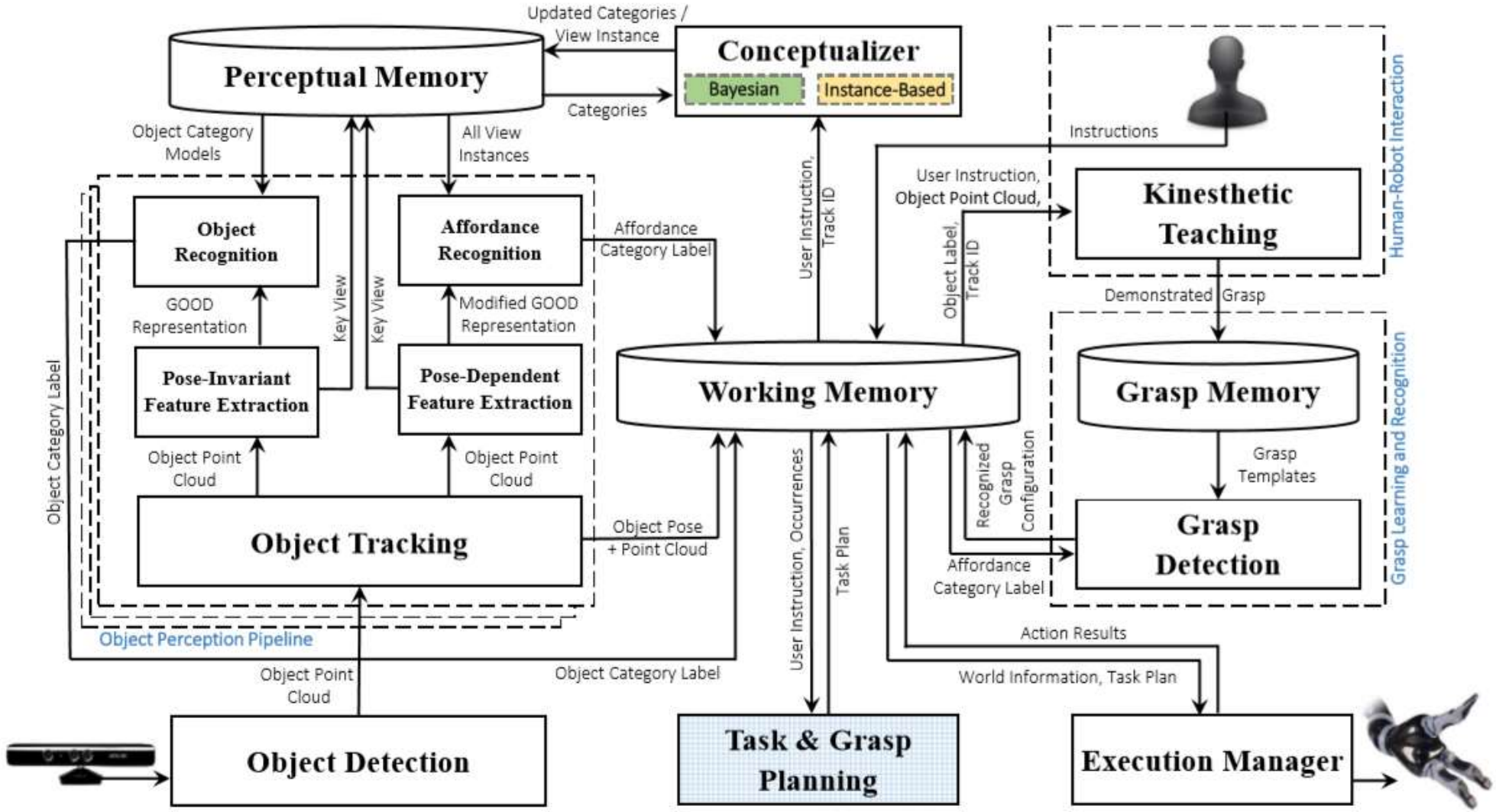
Fig. 4. Constructing Local Reference Frames (LRF) for the bottle object in two different situations. The red, green and blue lines represent the unambiguous **X**, **Y**, **Z** axes respectively.

Interactive Open-Ended

for affordance recognition.

2) Object Category Learning and Recognition: Given an input object point cloud, the *Pose-Invariant Feature Extraction* module computes the Global Orthographic Object Descriptor (GOOD) [31] to represent the object view. GOOD is formed by concatenating the three orthographic projections of the object view in a unique and repeatable local reference frame [31]. For category learning, an open-ended formulation of the Naive Bayes approach is adopted [7]. Therefore,

Interactive Open-Ended



Interactive Open-Ended

TABLE I

SUMMARY OF OPEN-ENDED EVALUATIONS.

Approaches	#QCI	ALC	AIC	GCA	APA
BoW(#)[7]	1811.60	47.40	14.78	0.69	0.75
LDA [37]	900.20	31.00	12.25	0.68	0.76
Local-LDA(*)[11]	1359.50	**49.00**	10.01	0.75	0.78
Our Work(*)	**1249.10**	**49.00**	**8.46**	**0.79**	**0.83**

(*) Stopping condition was "*lack of data*". (#) Stopping condition was "*lack of data*" in 6 out of 10 experiments.

Interactive Open-Ended

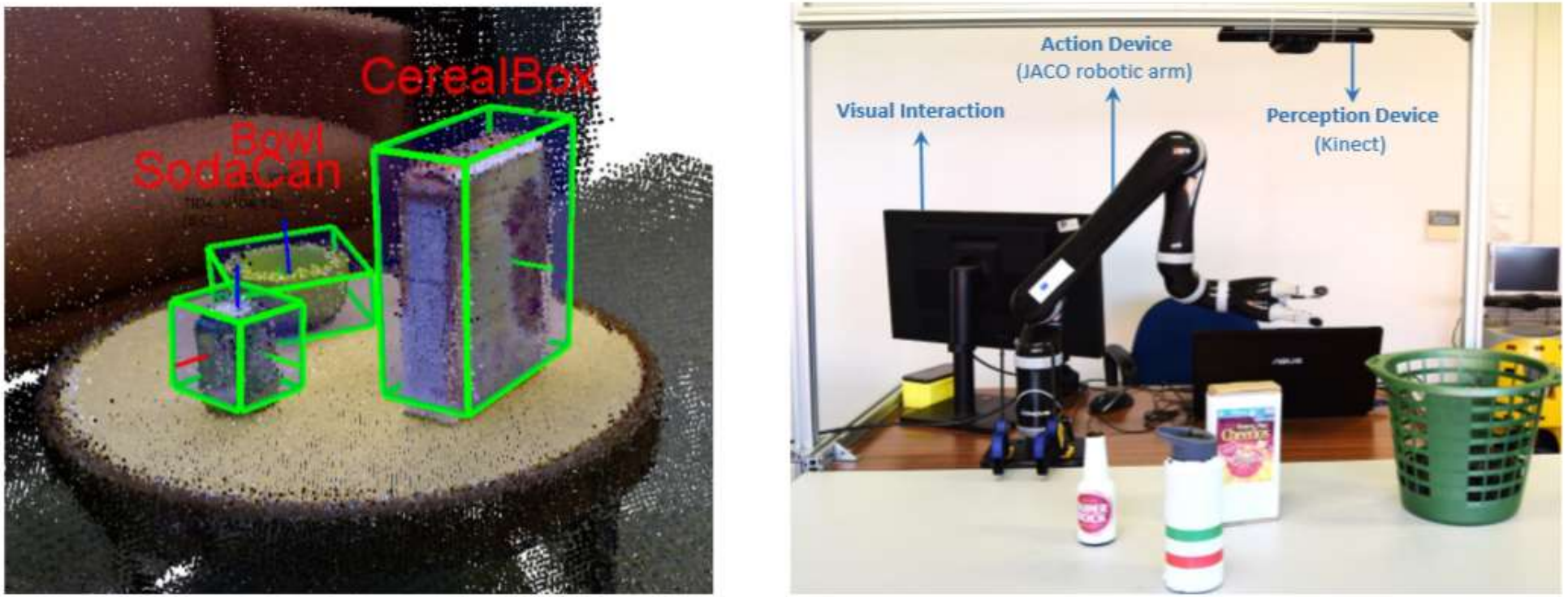


Fig. 7. System demonstrations: (*left*) using Washington RGB-D scene dataset [30]; (*right*) real-world robotic application.

Thank you

- End