Deep Reinforcement Learning and Control

Visual Imitation Learning and Low-shot Vision-based manipulation with Transporters

Fall 2021, CMU 10-703

Instructors:
Katerina Fragkiadaki
Ruslan Salakhutdinov

11/22/2021

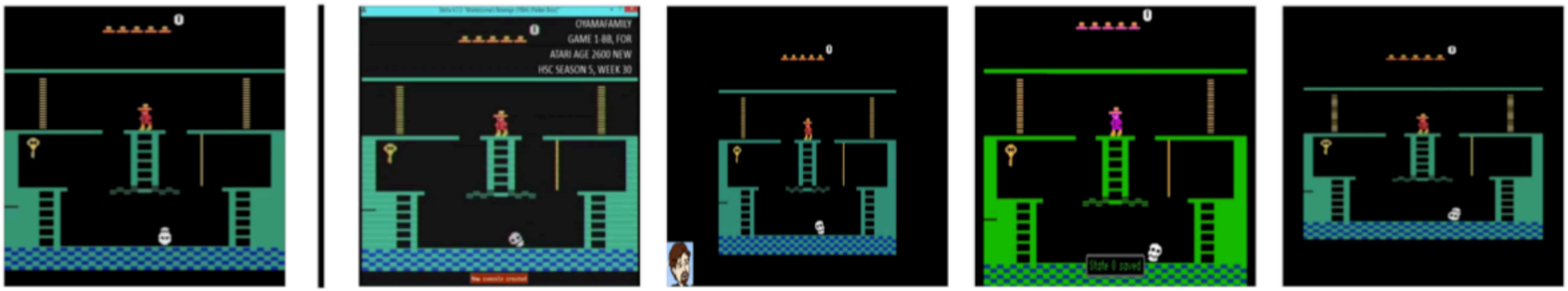
Guest Lecture by Daniel Seita

With slides from Prof. Fragkiadaki, Andy Zeng, and others.

Lecture Outline

- <u>Learning Acrobatics from Watching YouTube</u>
  - Pose Estimation
  - Motion Reconstruction
  - Motion Imitation
  - Summary and Takeaways
- Sample-Efficient Visual Imitation Learning for Robotic Manipulation
  - Manipulation via Rearranging Pixels
  - Transporter Networks
  - Goal-Conditioned Transporter Networks
  - Benchmark for Object Rearrangement
  - Summary and Takeaways

Review: Visual Imitation of Atari from YouTube



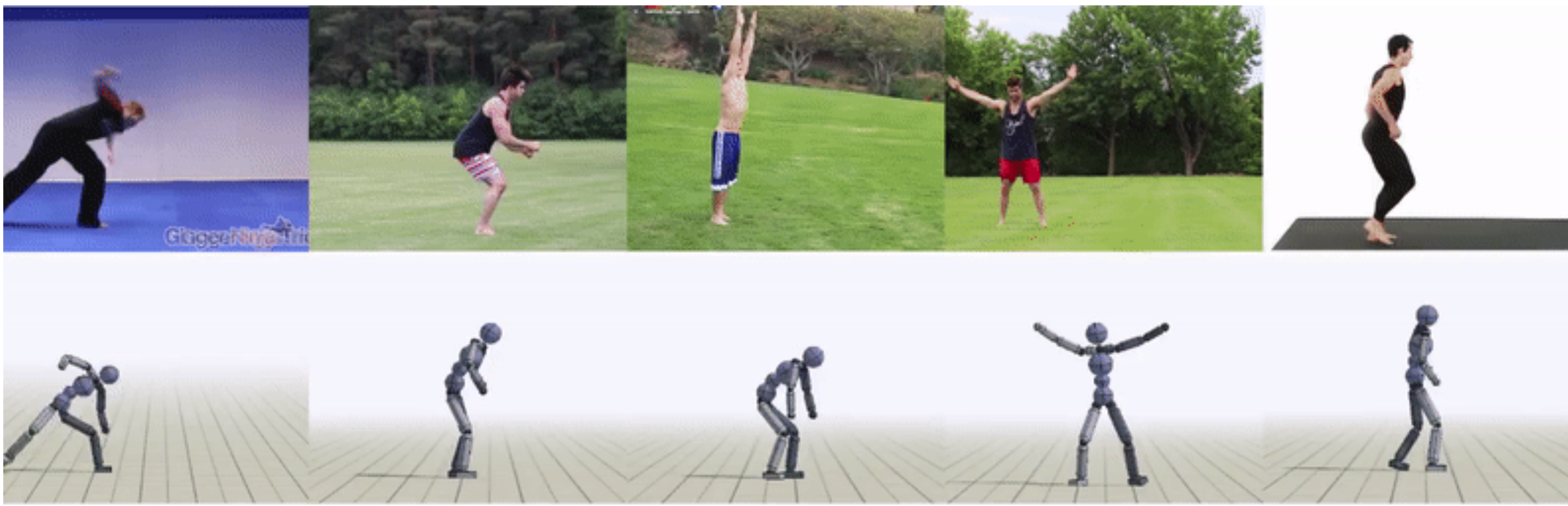
(a) ALE frame (b) Frames from different YouTube videos

- Input: video demonstrations (without rewards).
- Self-supervised visual representation learning to bridge the domain gap between YouTube video demonstrations of people playing the game, with the frames the game emits
- Given one video demo, use visual similarity encoded as frame embedding distance as imitation reward, to be added (optionally) to environment rewards.

Aytar*, Pfaff* et al., Playing Hard Exploration Games by Watching YouTube, NeurIPS 2018.

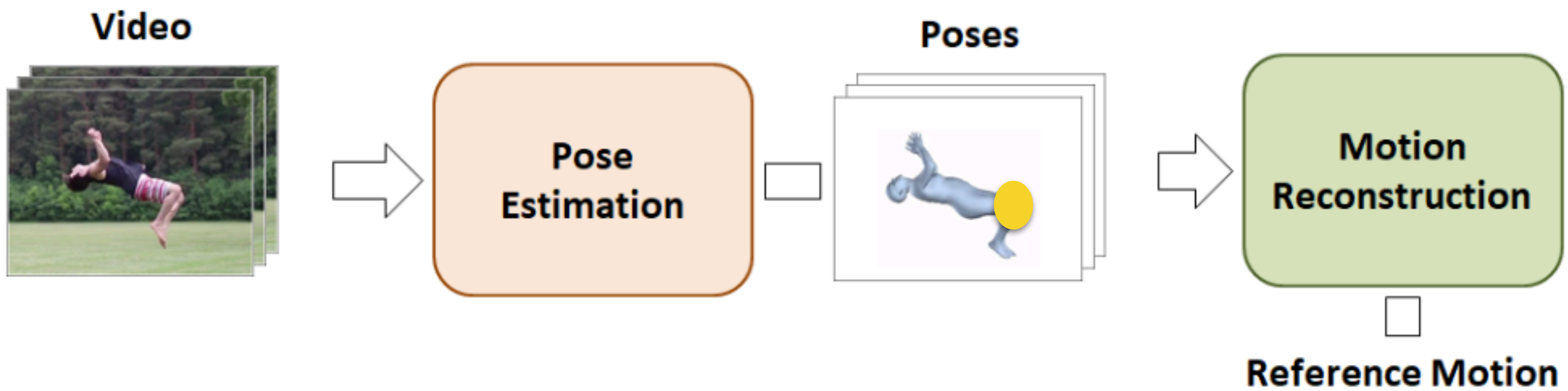
Learning acrobatics from watching YouTube

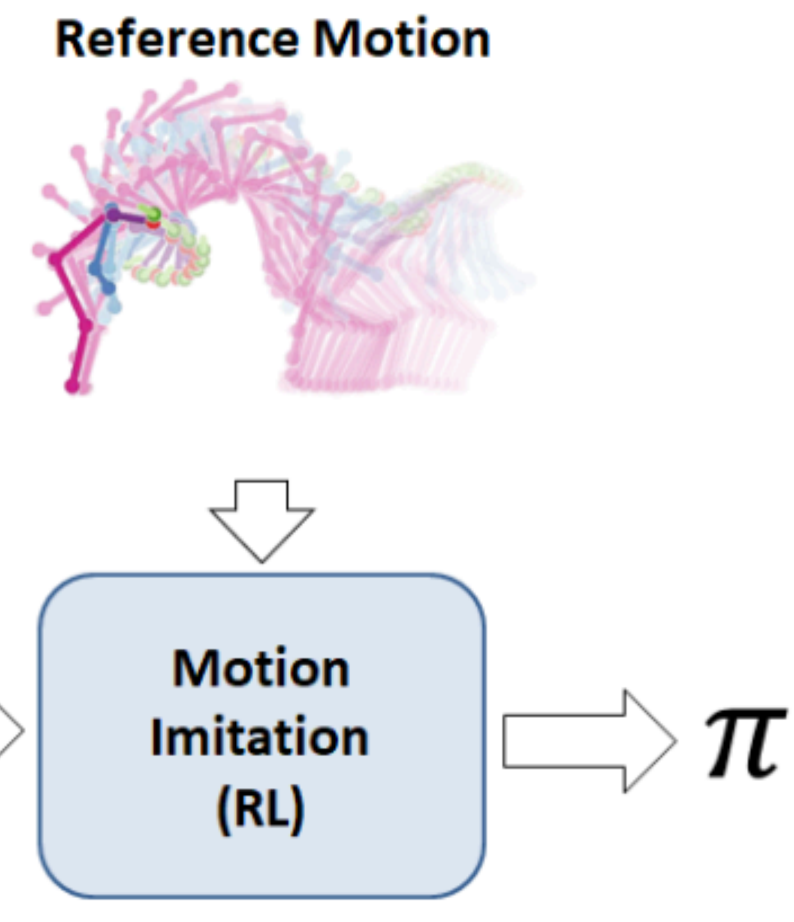
(Not by engineering the robot's actions)
Like the prior work, learn from YouTube videos.



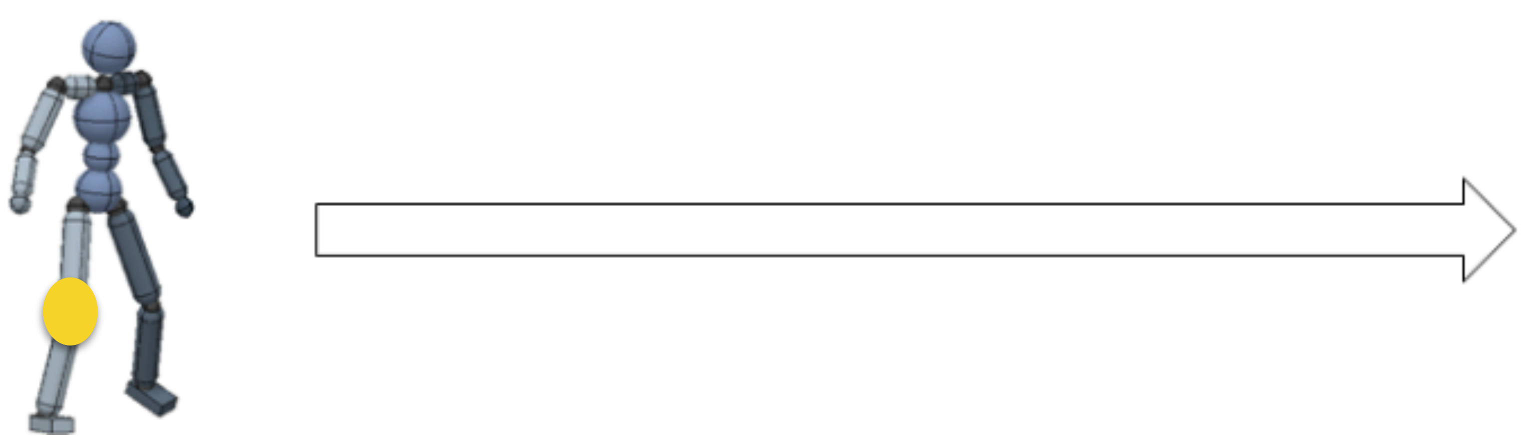
Peng et al., SFV: Reinforcement Learning of Physical Skills from Videos, SIGGRAPH Asia 2018
Blog post: https://bair.berkeley.edu/blog/2018/10/09/sfv/

Learning acrobatics from watching YouTube



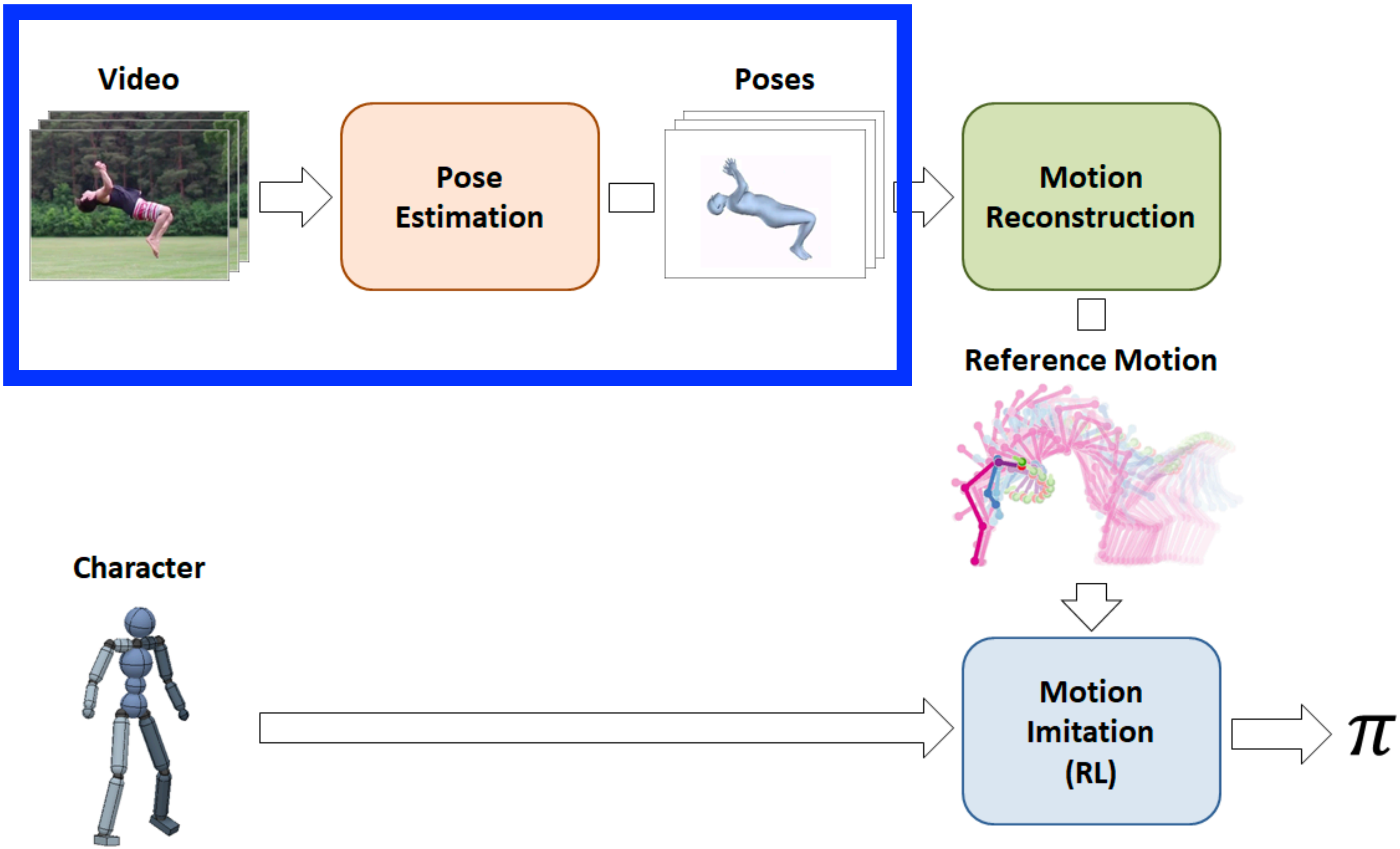


Our agent has a pre-defined mapping between its body joints and the human body joints



We'll discuss each of these 3 major components now.

The Pipeline



Approach relies on estimating 2D and 3D poses

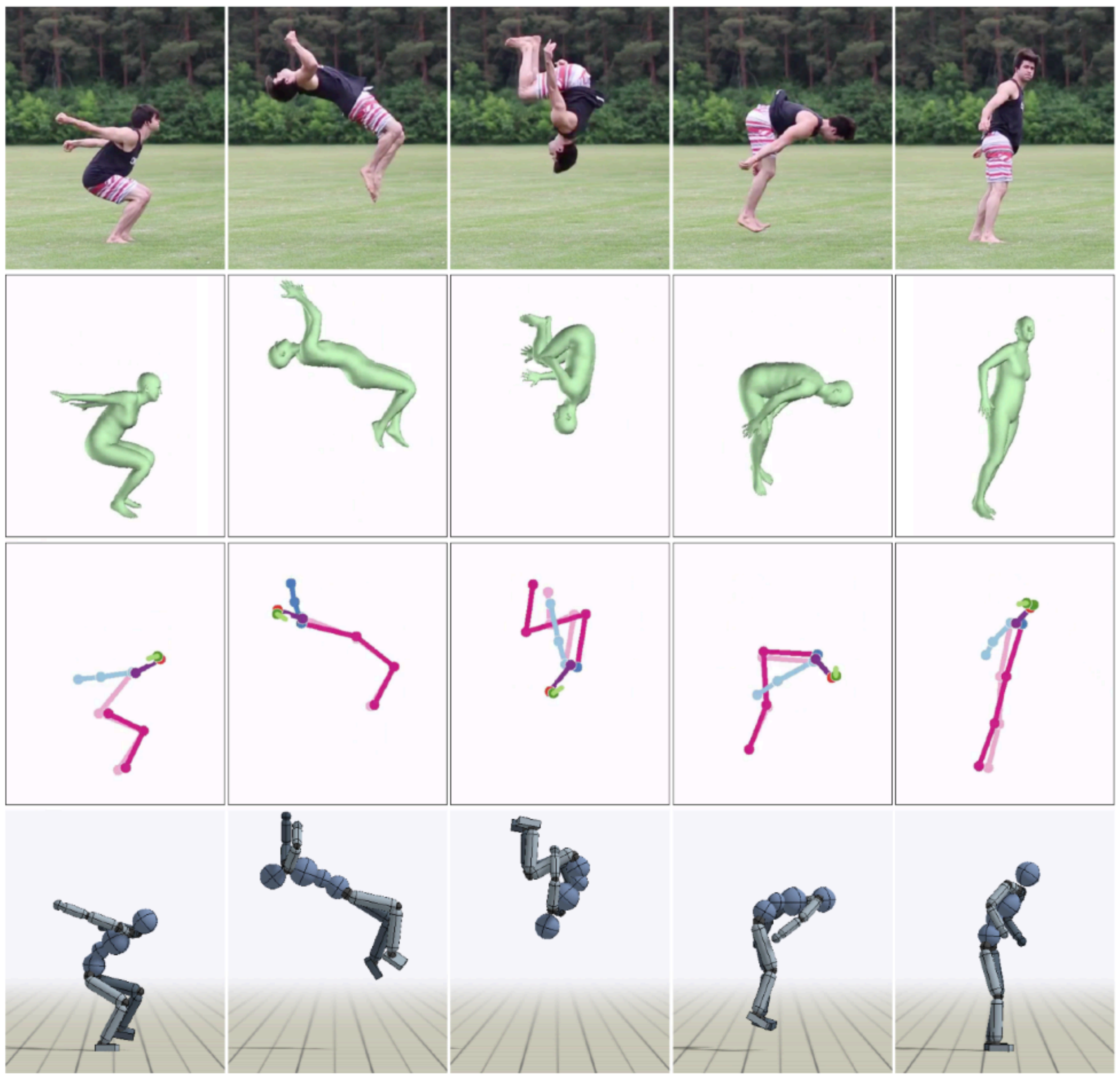
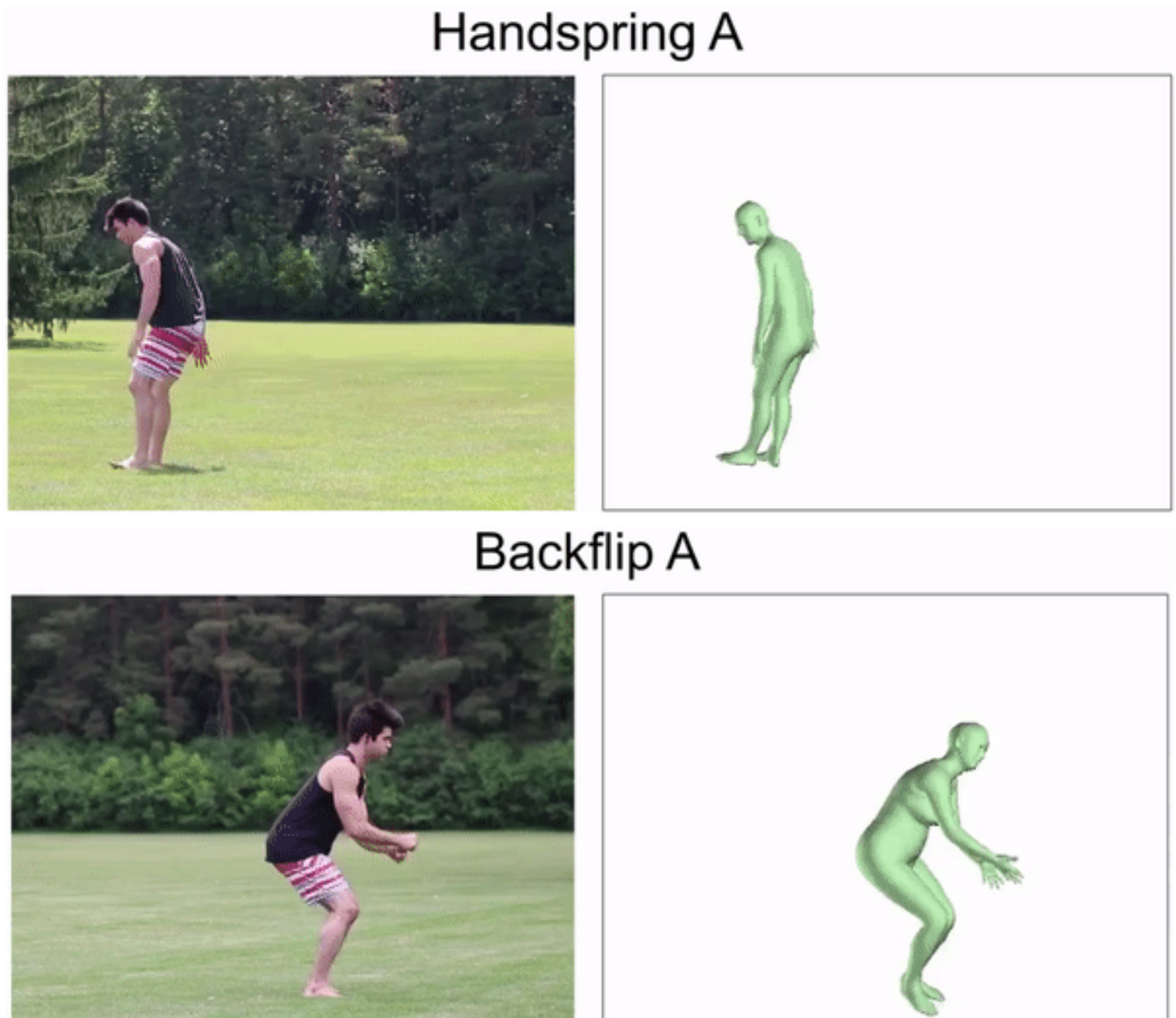


Fig. 3. Comparison of the motions generated by different stages of the pipeline for backflip A. **Top-to-Bottom:** Input video clip, 3D pose estimator, 2D pose estimator, simulated character.

- For 2D, build upon OpenPose [1], outputs 2D poses as joint coordinates in pixel space.
- For 3D, build upon Human Mesh Recovery [2], directly predict 3D pose/shape of human mesh.
- Train 2D and 3D pose estimators independently to every frame.

[1]: Wei et al., Convolutional Pose Machines, CVPR 2016.
[2]: Kanazawa et al., End-to-end Recovery of Human Shape and Pose, CVPR 2018.

Example of 3D Pose Estimates



Aside: SMPL, a 3D human shape model

(This is what "3D poses” mean in SFV.)

SMPL [M. Loper et al.]: a low-parametric model learned from aligning high-resolution 3D scans.

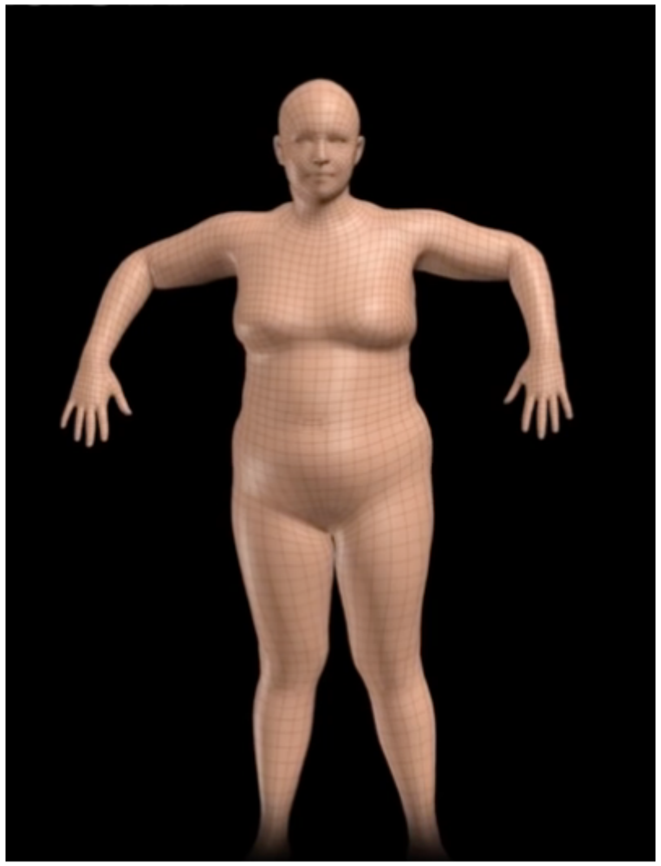
3D mesh
SMPL(θ , β)

Pose

θ

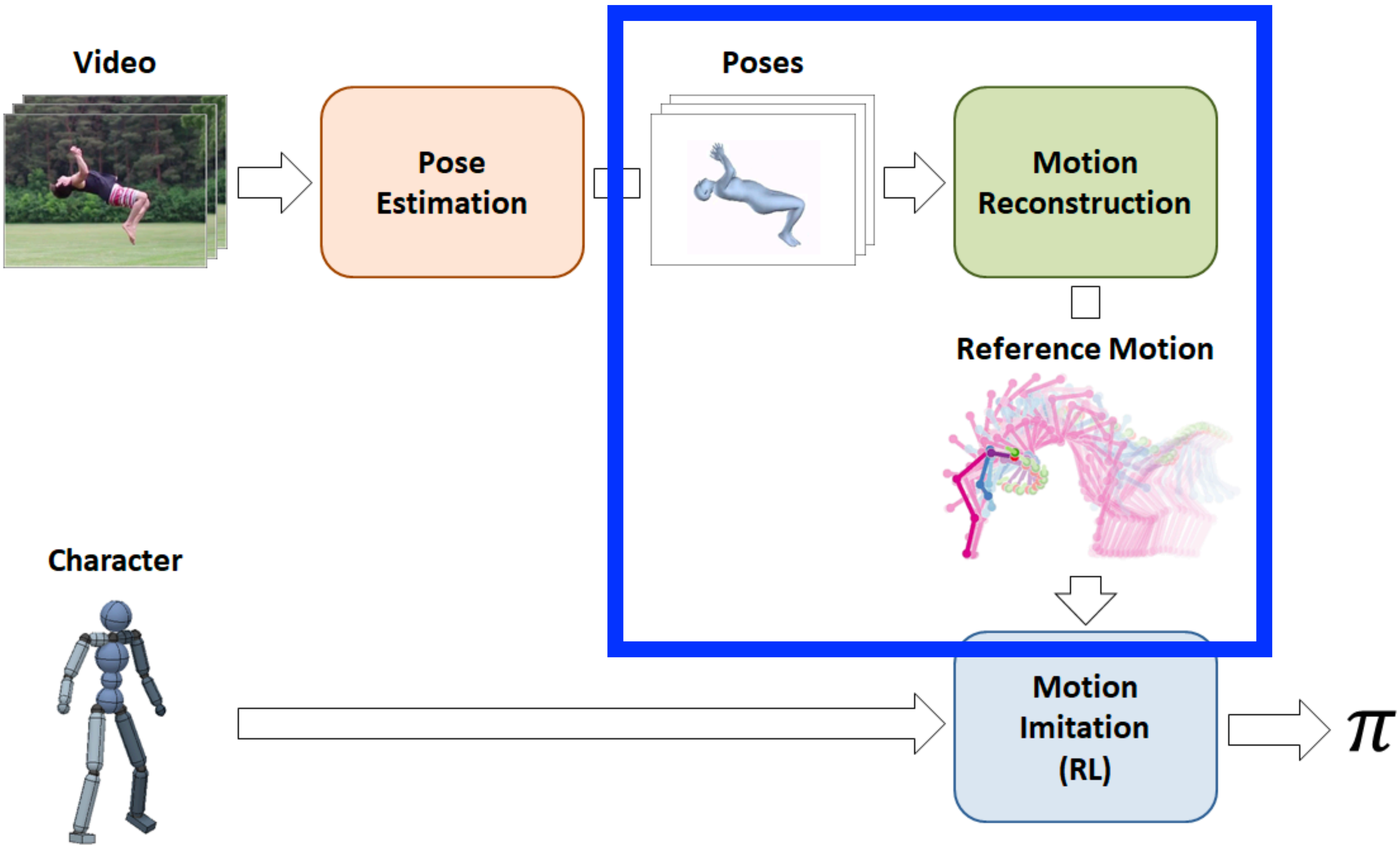
Shape

β

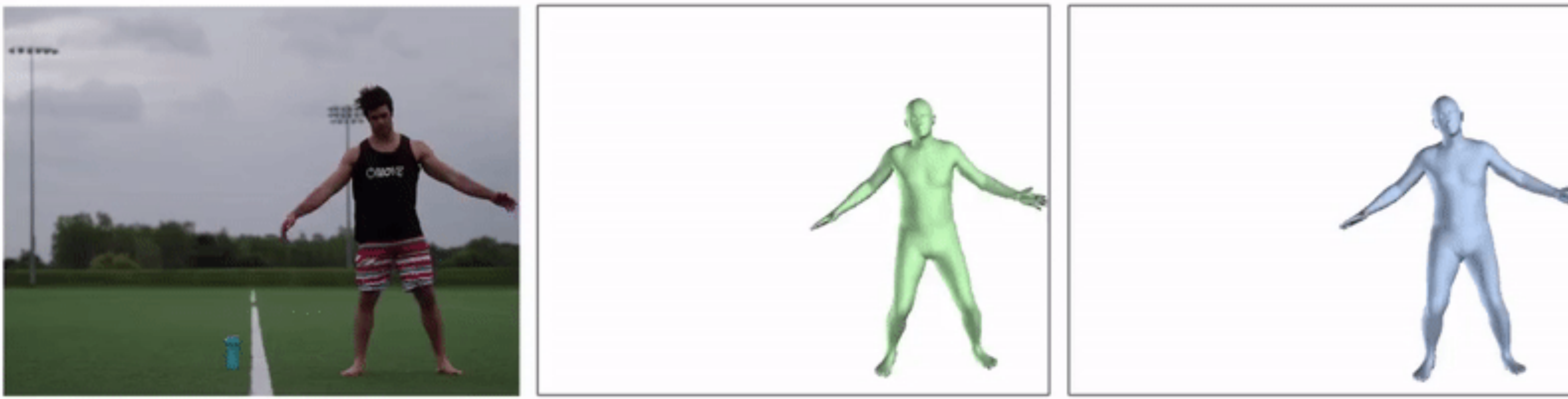


Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, Michael J. Black, SMPL: A Skinned Multi-Person Linear Model (SIGGRAPH Asia 2015)

The Pipeline



Temporal Smoothing



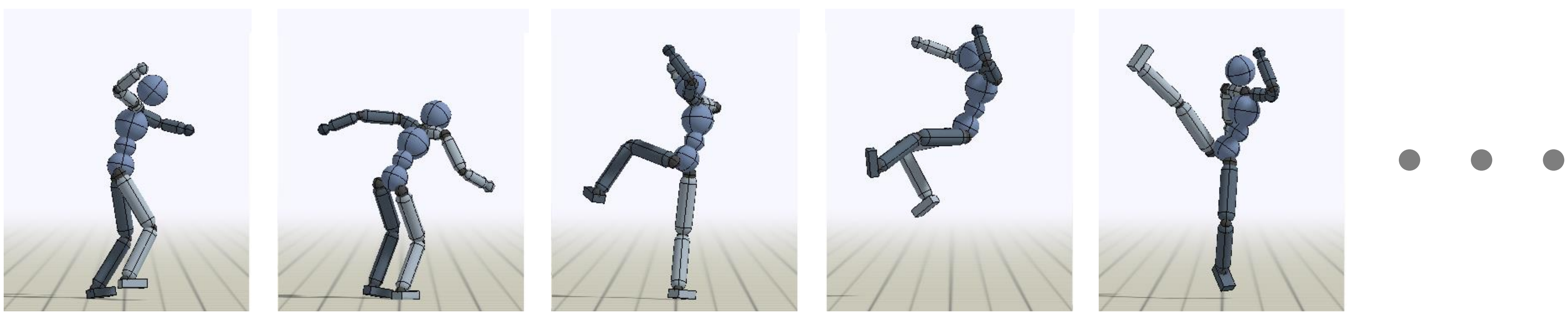
Video: Cartwheel A　　Before Reconstruction　　After Reconstruction

High-level idea:

- Consolidate / improve previously collected 2D and 3D pose estimates.
- For 3D, we have latent states at each time, can “reproject” to 3D and compare pose positions with the 2D pose estimator (this is why there’s both 2D and 3D pose estimates), and optimize over latents.
- Enforce smoothness loss of 3D joint positions among adjacent frames.

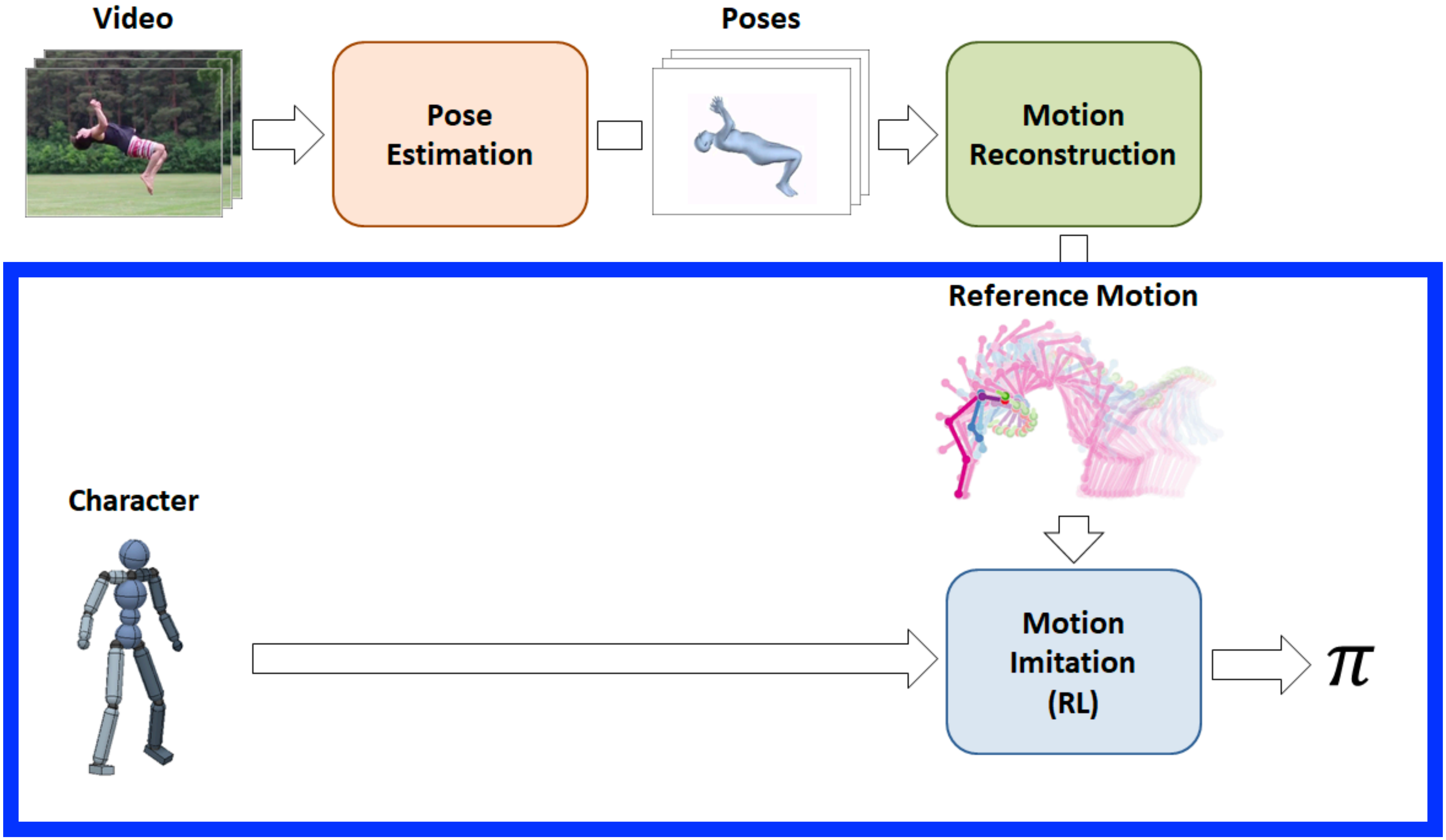
Result: A (Smooth) Reference Motion

Reference Motion



$$\{\hat{q}_0, \hat{q}_1, \ldots, \hat{q}_T\}$$

The Pipeline

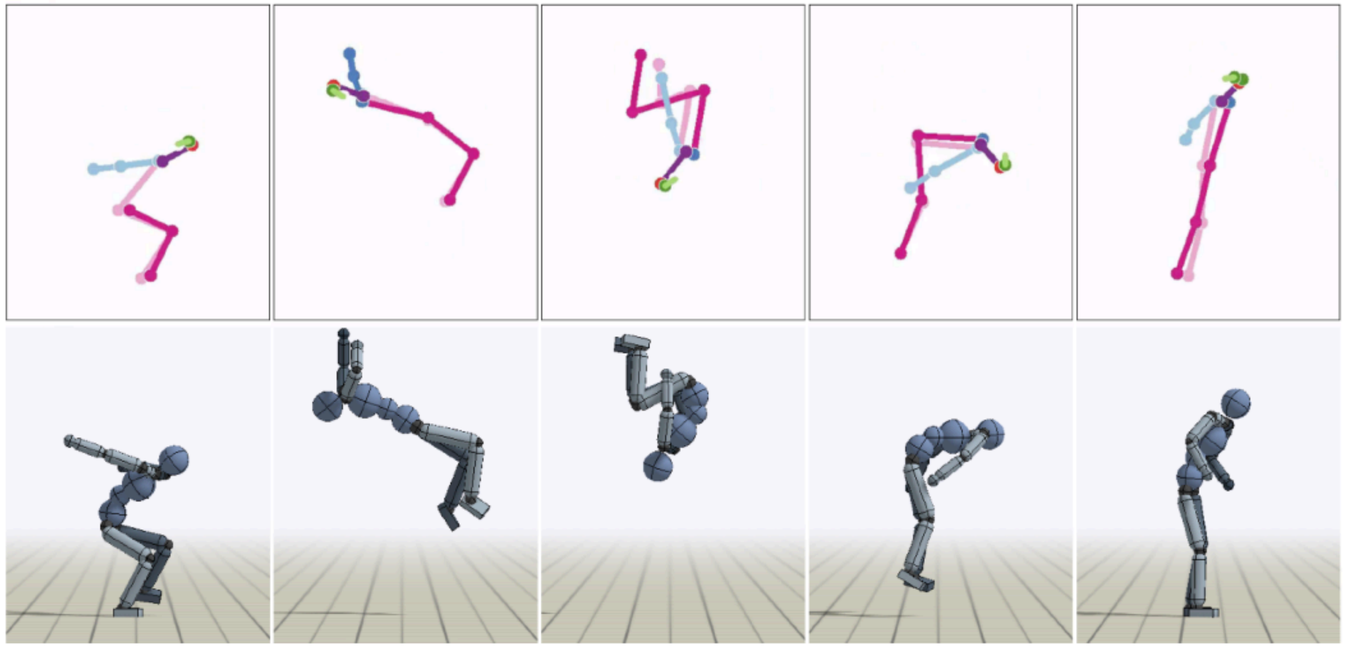


Question: why do we need RL?

- Why do we not just do behavioral cloning to imitate the reference motion sequence?

Question: why do we need RL?

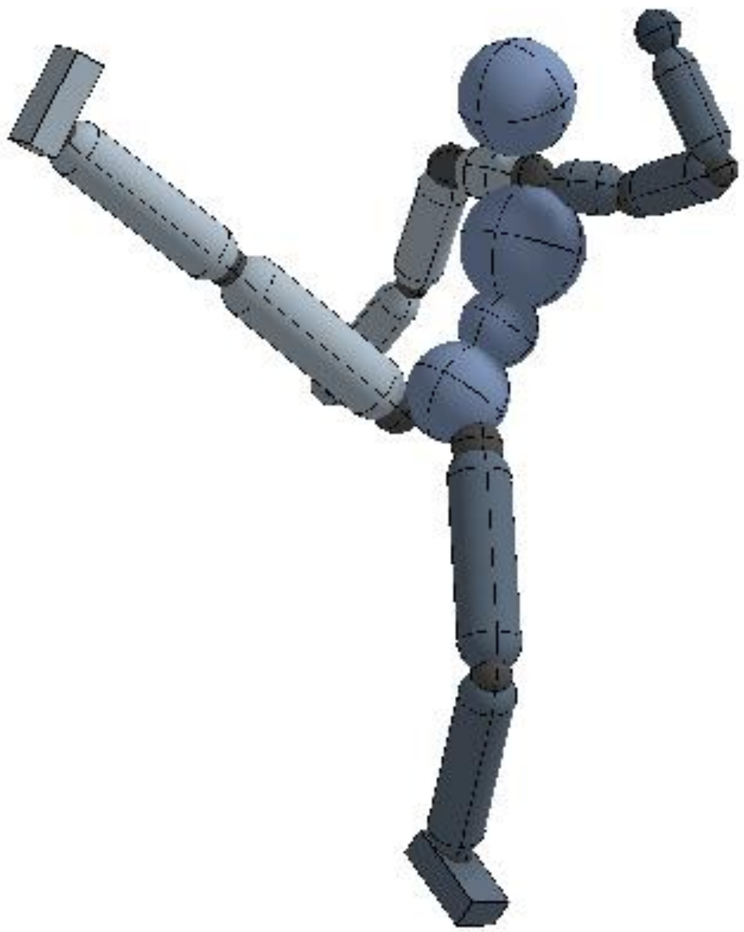
- Why do we not just do behavioral cloning to imitate the reference motion sequence?
  - The method predicts keypoints (human body parts) from video frames.
  - Cannot "copy and paste" keypoints from 2D frames into 2D frames of the simulator, because the agent controls its joints through (simulated) motor torques.
  - The agent has to learn how to control itself in the simulator, accounting for gravity and other forces. (Also, no simulator is exactly the same as reality.)
- While this might seem like a subtle / arcane technical point, it's important to clarify if RL is the right "tool" to use.



State + Action

State:

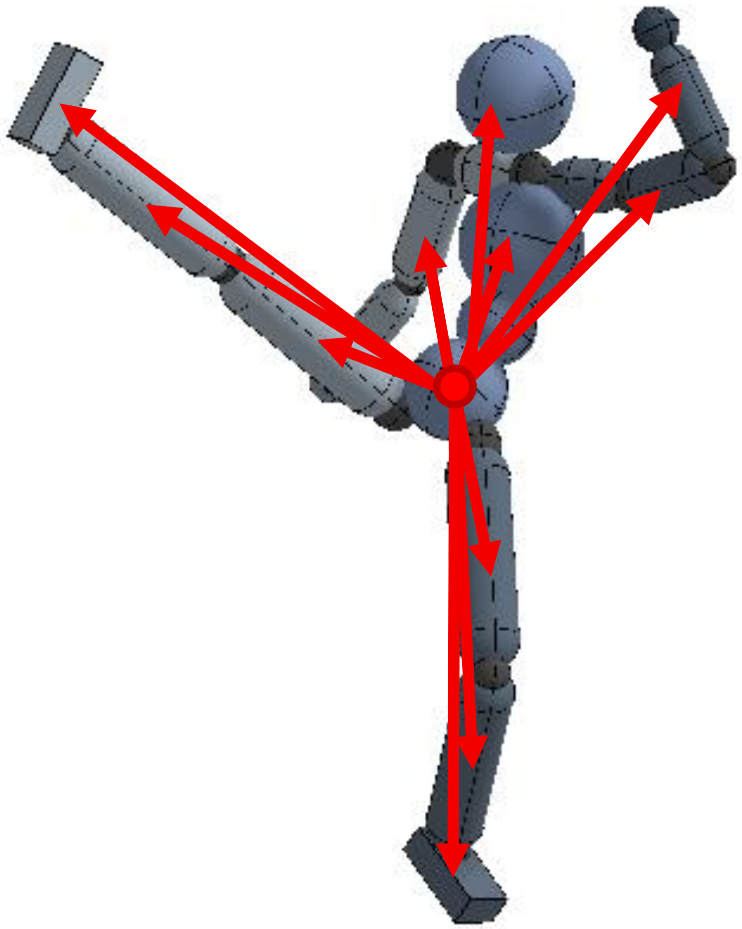
- link positions
- link velocities



State + Action

State:

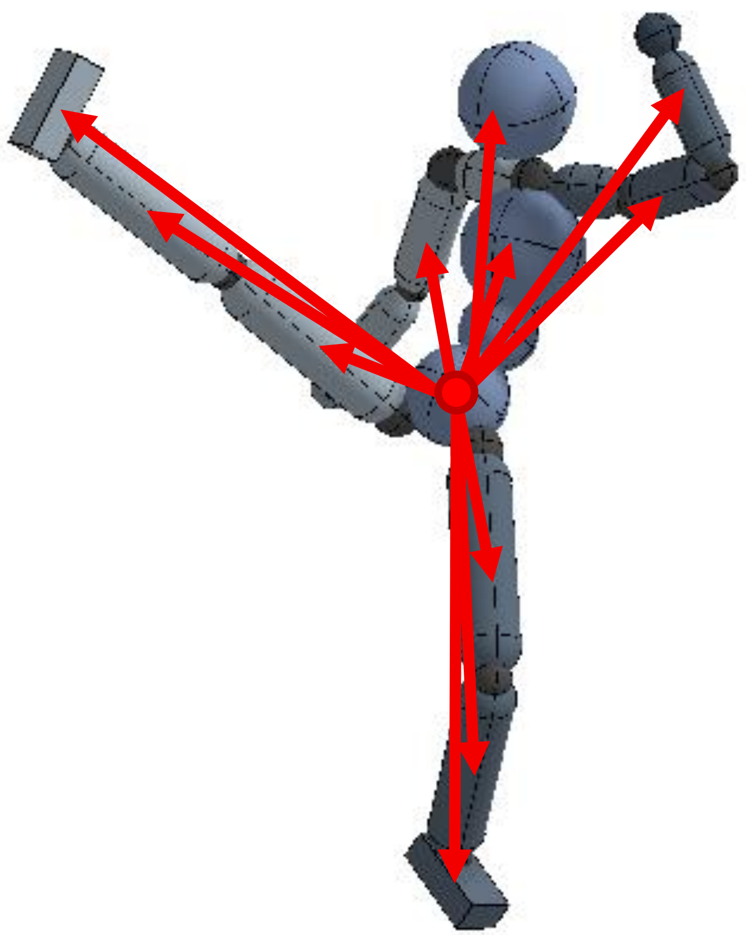
- link positions
- link velocities



State + Action

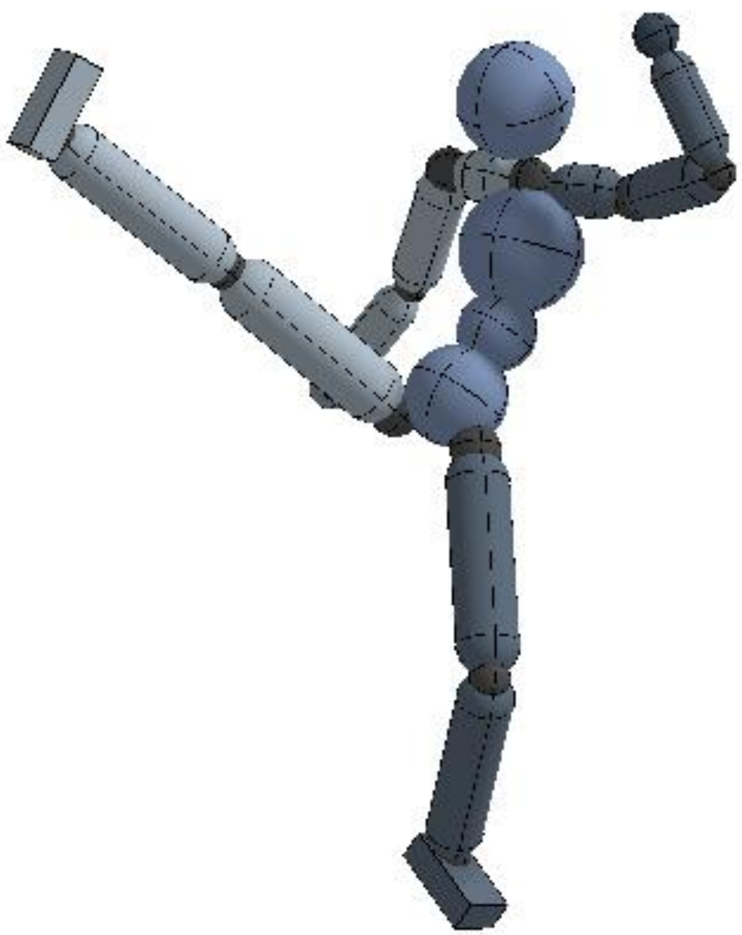
State:

- link positions
- link velocities



Action:

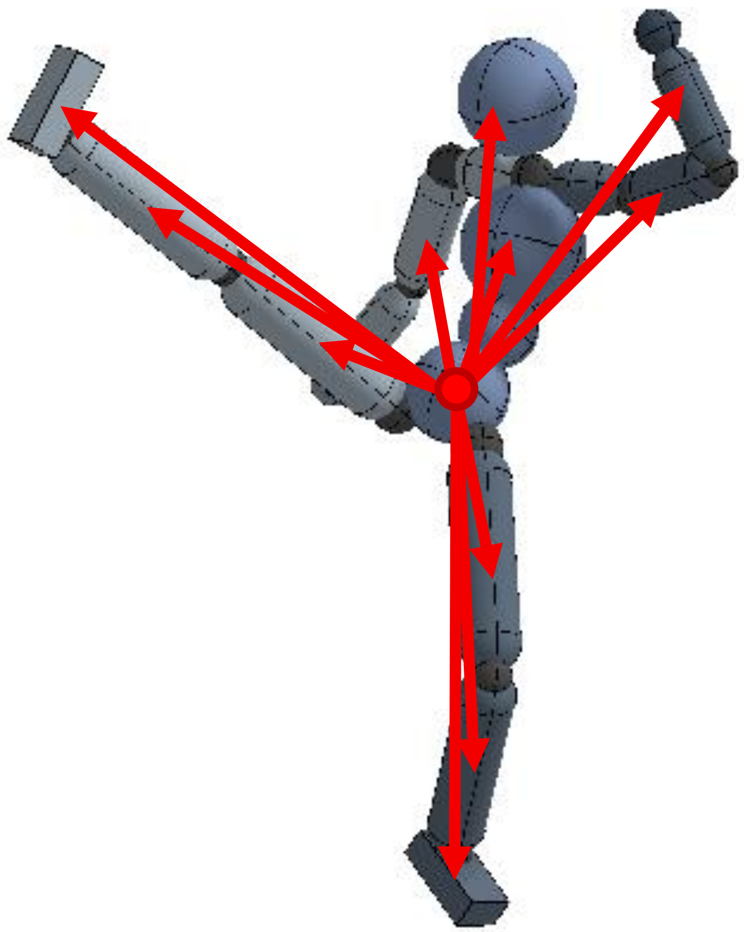
- PD targets



State + Action

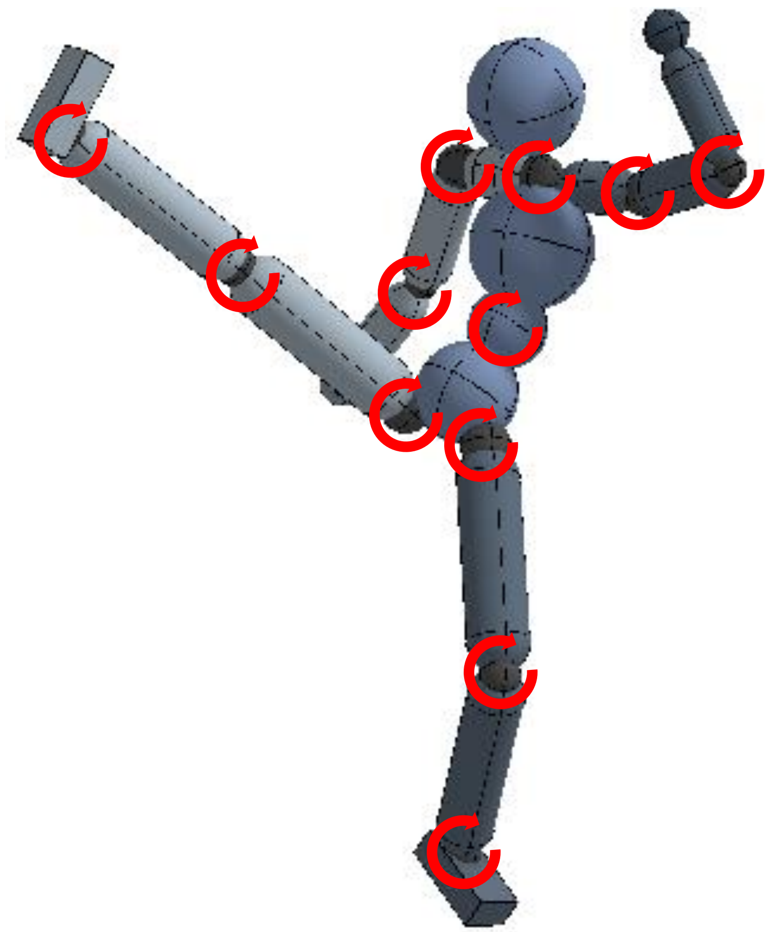
State:

- link positions
- link velocities



Action:

- PD targets

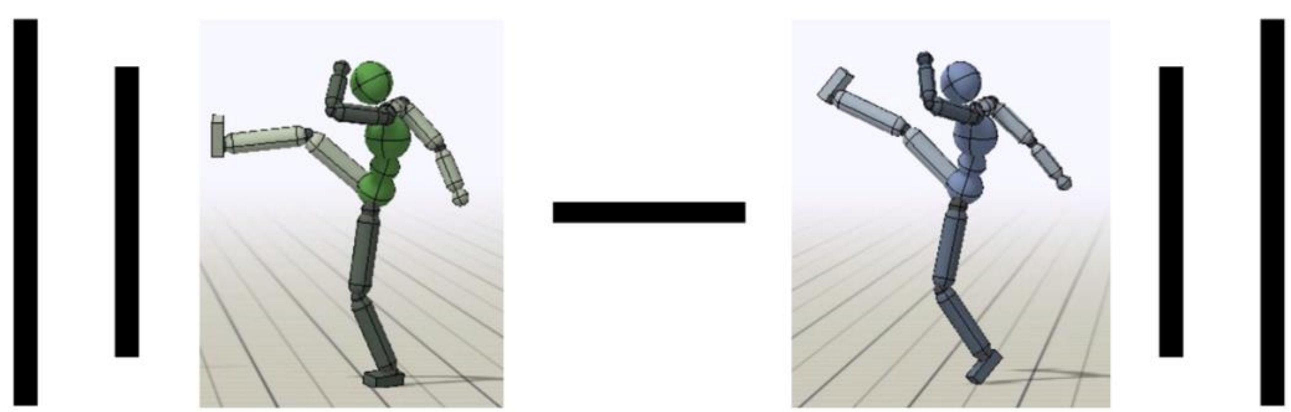


Imitation Objective

The reference trajectory, constructed from the pose estimator with temporal smoothing.

$$r_t = \exp\left(-2\ \left\| \hat{q}_t - q_t \right\|^2\right)$$

Imitation Objective



This reward considers the differences in joint orientations, joint (angular) velocities, end-effector positions, and center-of-mass.

Proximal Policy Optimization (PPO)

$$\max_{\theta} \quad J(\theta)$$

$$\text{s.t.} \quad \mathbb{E}_{s_t \sim d_\theta(s_t)}\left[KL\left(\pi_{\theta_{old}}(\cdot|s_t)\middle|\pi_\theta(\cdot|s_t)\right)\right] \leq \delta_{KL}$$

Schulman et al., Proximal Policy Optimization Algorithms, arXiv 2017.
See also: https://spinningup.openai.com/en/latest/algorithms/ppo.html

Proximal Policy Optimization (PPO)

$$\max_{\theta} \quad J(\theta)$$

$$\text{s.t.} \quad \mathbb{E}_{s_t \sim d_\theta(s_t)} \left[KL \left(\pi_{\theta_{old}}(\cdot|s_t) \middle| \pi_\theta(\cdot|s_t) \right) \right] \leq \delta_{KL}$$

Why might PPO be a reasonable option?

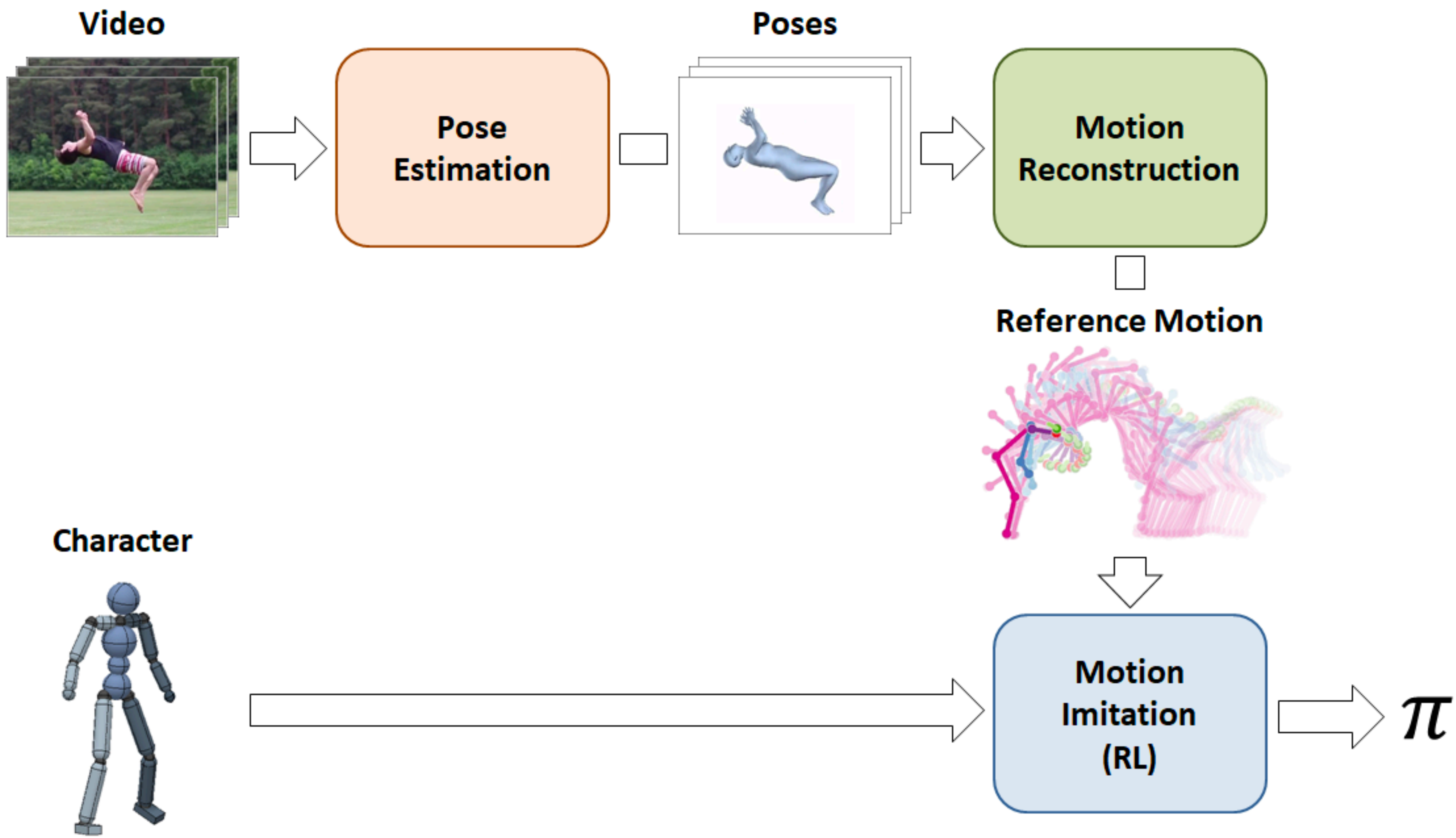
- Generally a good / easy RL algorithm, and we have dense rewards.
- In simulation, we can get on-policy samples quickly, so data collection might be less of a time bottleneck (contrast this with real-world robots).

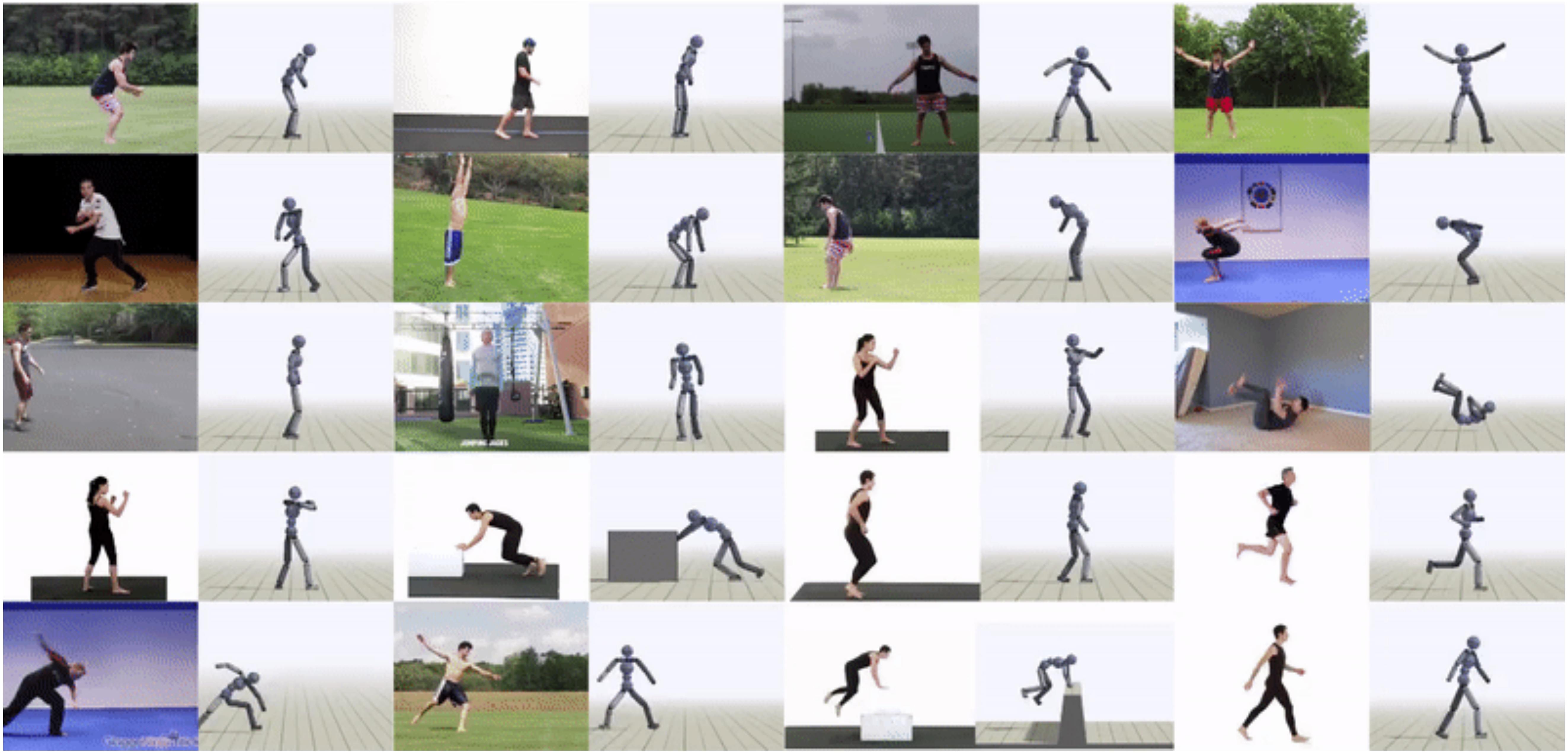
Adaptive State Initialization(ASI)

- PPO alone may still struggle, but there is another trick: in simulation, we can reset the agent to start at a variety of initial states.
- Approach: <u>learn</u> the initial state distribution!
  - Formulate as cooperative multi-agent RL.
  - First agent: the policy.
  - Second agent: proposes initial state that the policy begins each episode.
- Can still use policy gradients for this (see paper for derivation).

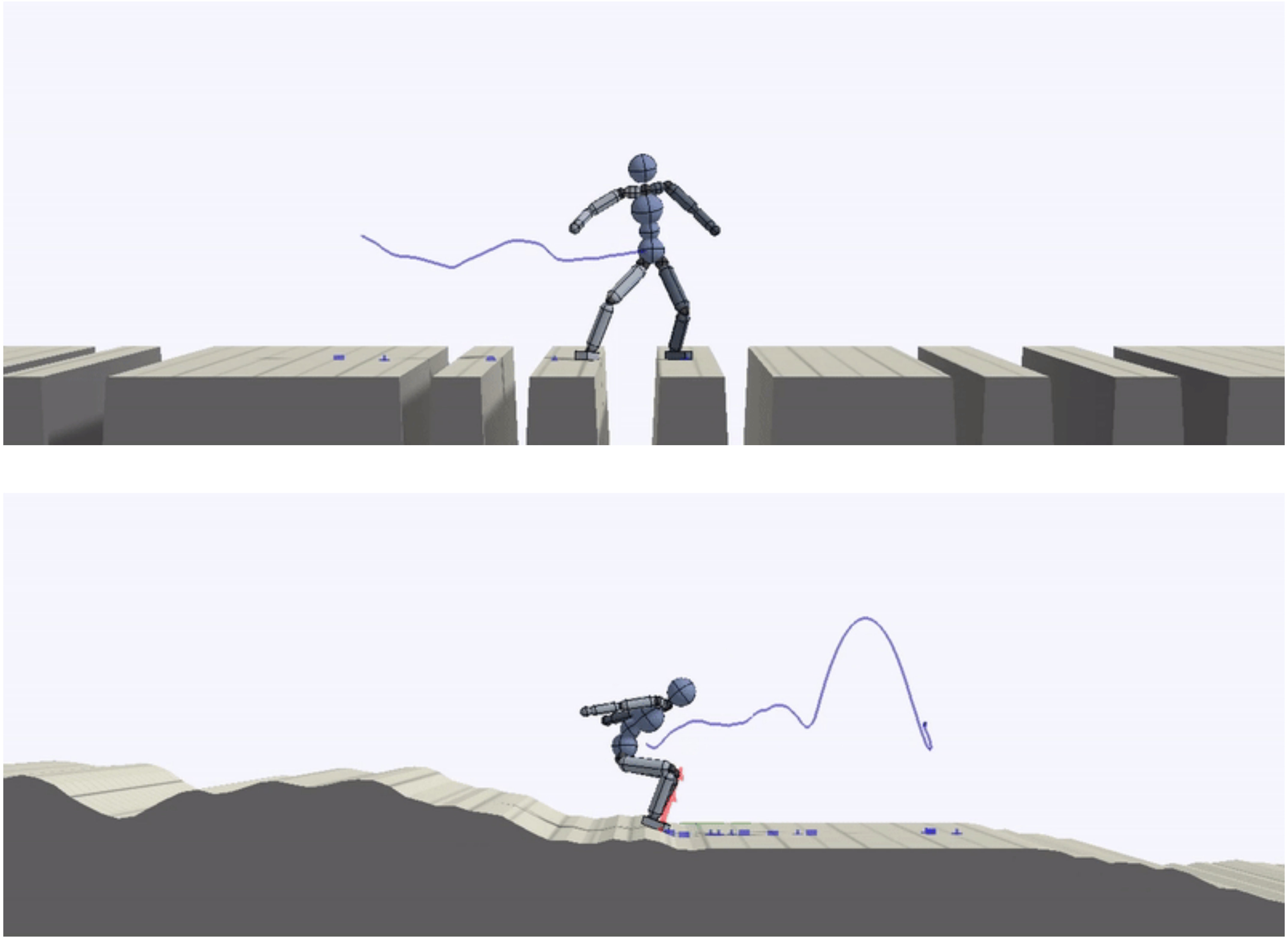
Can be interpreted as a form of "curriculum learning," with similar ideas in:
- Florensa et al., Reverse Curriculum Generation for RL, CoRL 2017.
- Florensa*, Held*, Geng*, et al., Automatic Goal Generation for RL, ICML 2018.

Recap / Summary of Approach





Adapting a skill through RL to novel environments



In order for the agent to match keypoints from the video, it must necessarily avoid falling down!

Failure modes

Video: Gangnam Style | Reference Motion | Simulation

(Notice the motion of the simulated agent's hands.)

Summary and Takeaways

- Can use knowledge of humans to use prior computer vision work to get good human pose estimates.
- We still need RL here (cannot just do BC).
- We can use RL with a dense reward from imitation.
- Combines IL and RL — the line between the techniques is blurry.
- Can result in acrobatic skills from just raw video!

Lecture Outline

- Learning Acrobatics from Watching YouTube
  - Pose Estimation
  - Motion Reconstruction
  - Motion Imitation
  - Summary and Takeaways
- Sample-Efficient Visual Imitation Learning for Robotic Manipulation
  - Manipulation via Rearranging Pixels
  - Transporter Networks
  - Goal-Conditioned Transporter Networks
  - Benchmarks for Object Rearrangement
  - Summary and Takeaways

Transporter Networks

Rearranging the Visual World for Robotic Manipulation

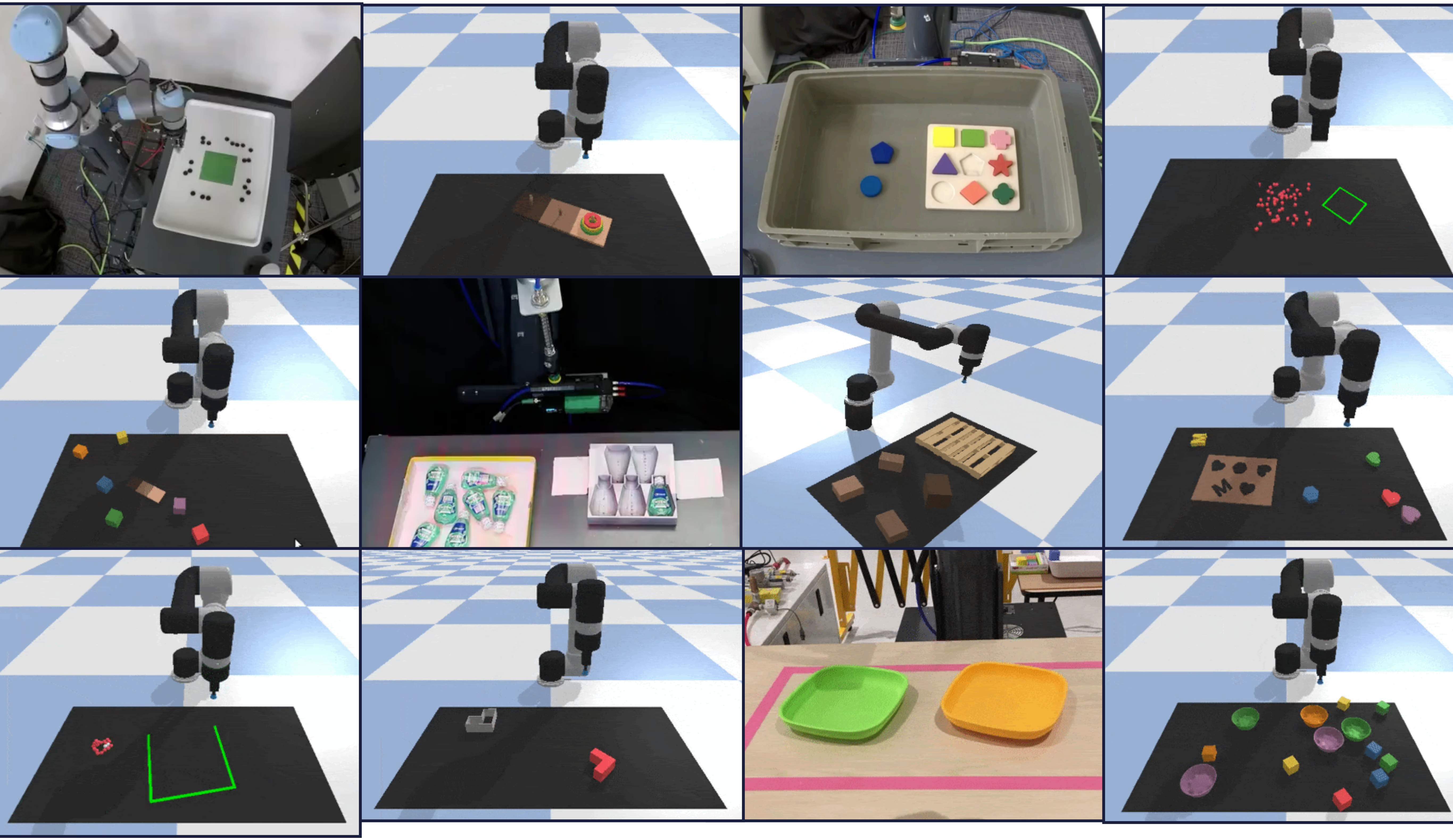
Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker,
Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin,
Dan Duong, Vikas Sindhwani, Johnny Lee

Conference on Robot Learning (CoRL) 2020
transporternets.github.io

Slides adapted from Andy Zeng et al.

Note: not to be confused with the "Transporter" introduced in "Unsupervised Learning of Object Keypoints for Perception and Control" by Kulkarni*, Gupta*, et al., NeurIPS 2019.

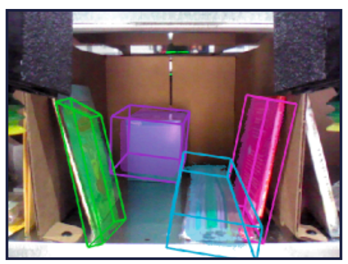
Google Research



Transporter Networks: Rearranging the Visual World for Robotic Manipulation
Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, Johnny Lee
transporternets.github.io

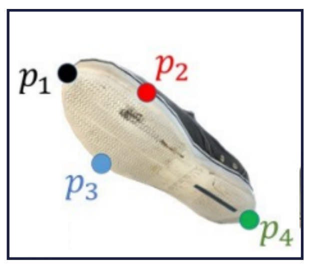
Object-Centric Representations

Poses



Zeng et al. ICRA '17

Keypoints



Manuelli et al. ISRR '19

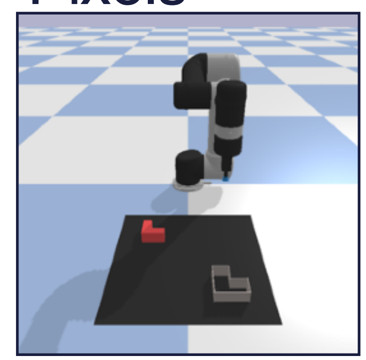
Descriptors

Florence et al. CoRL '18

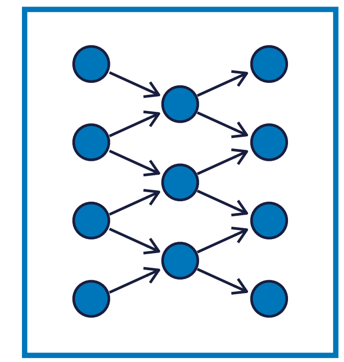
- Explicitly define "objects"
- Specialized data collection
- Unseen objects or piles?

End-to-End Models

Pixels



→



→ Actions

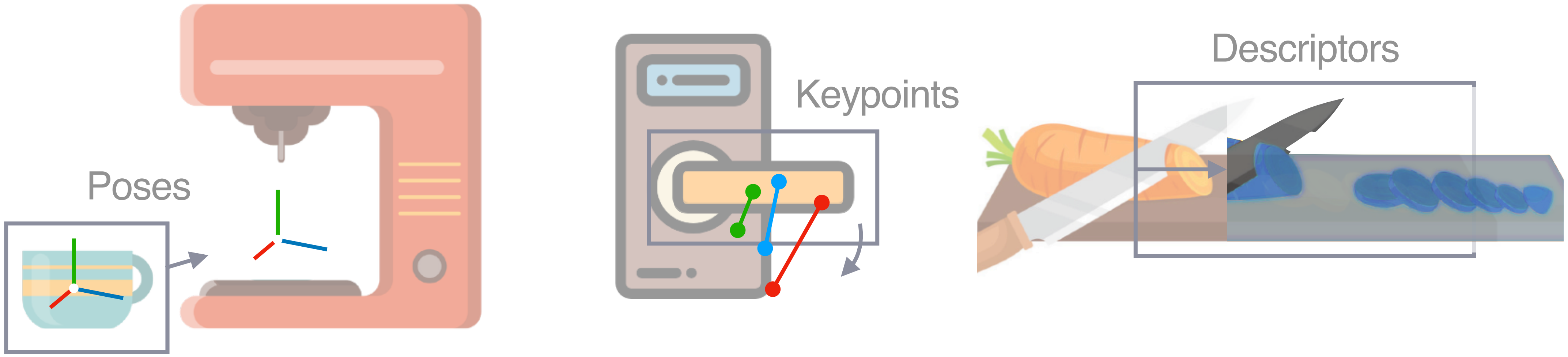
- Require massive amounts of data

+ Improves sample efficiency

Structure to improve sample efficiency without objectness?

Spatial Structure of Manipulation

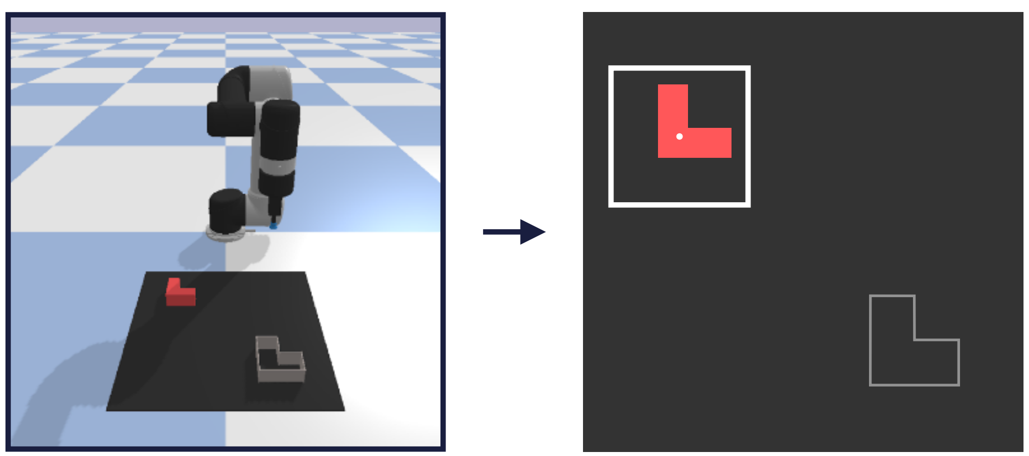
Manipulation → Rearranging objects = 3D space



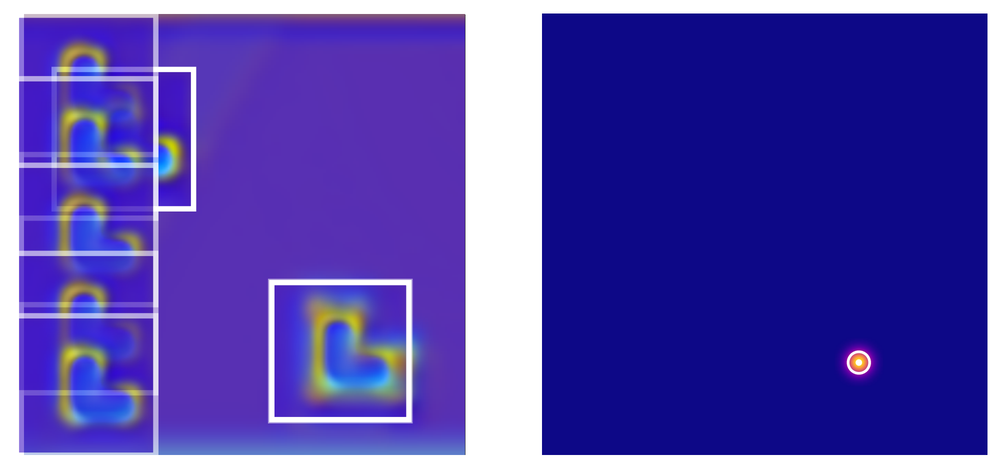
Infer displacements by rearranging visual input

Transporter Networks

1 Localize a region of interest



2 Infer its spatial displacement



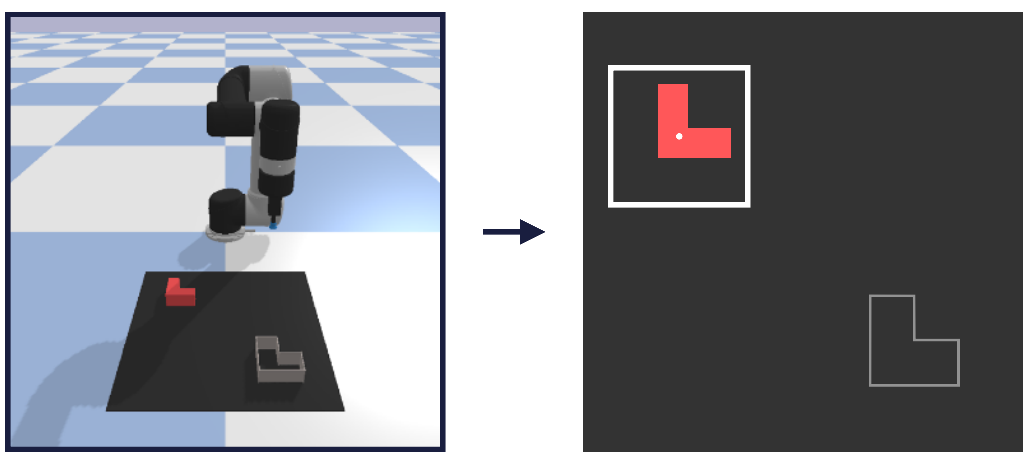
Correlation map

↑

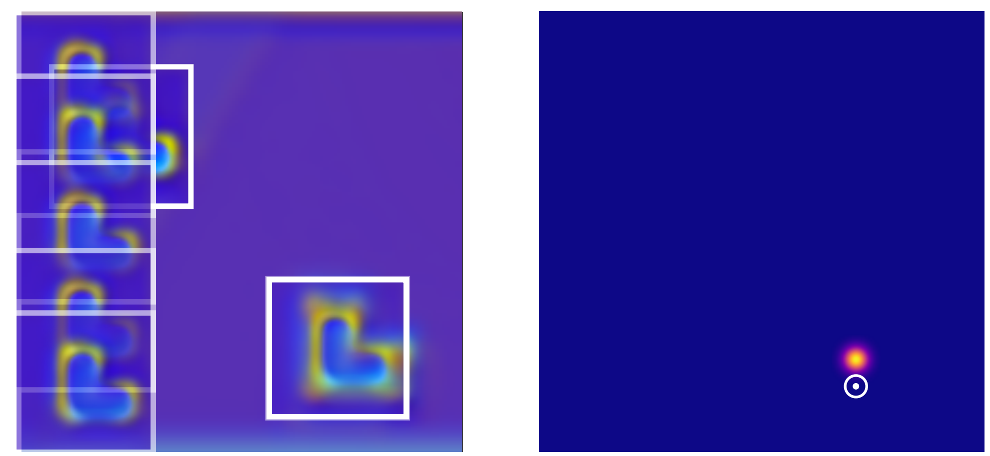
Just a convolution!

Transporter Networks

1 Localize a region of interest



2 Infer its spatial displacement



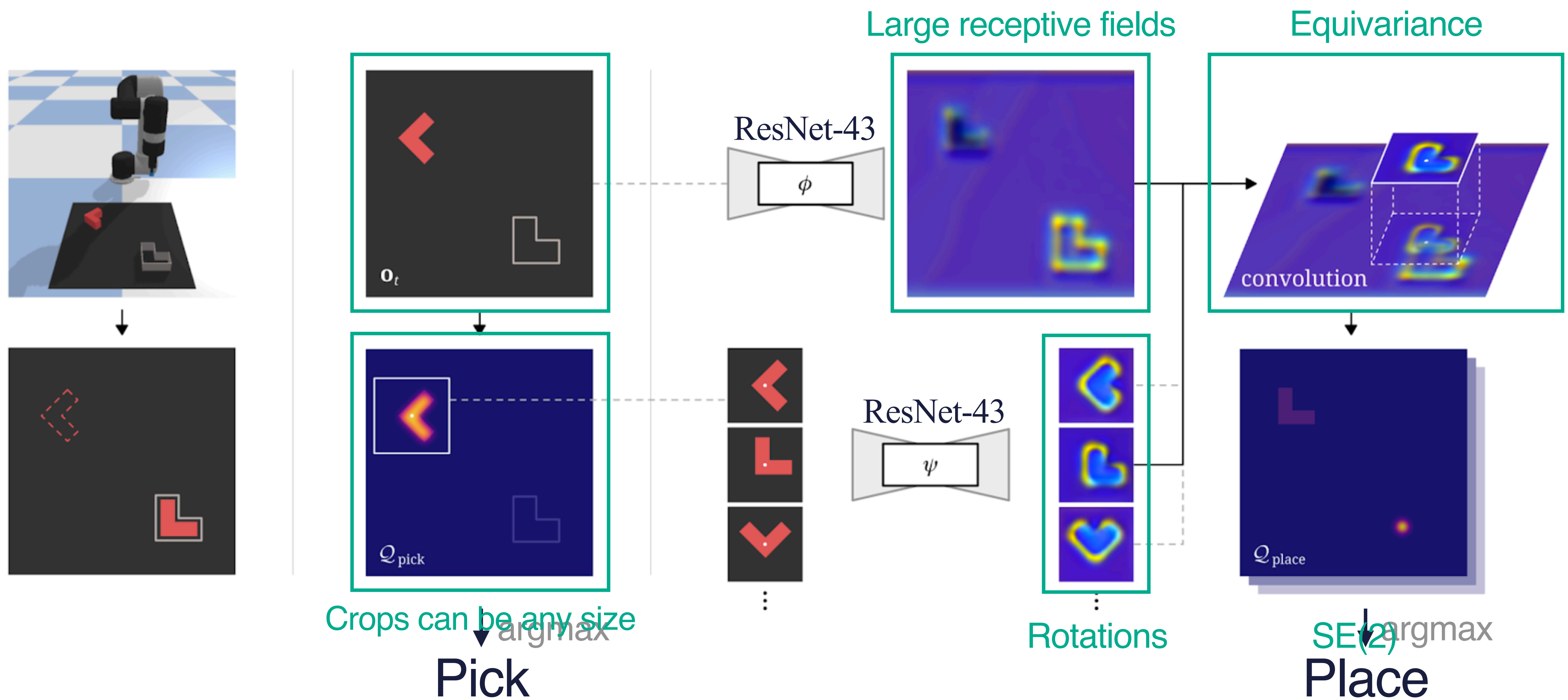
Correlation map

Just a convolution!

Intuition: why is this a reasonable approach?

- Consider a convolutional neural network on 2D images. The “transporting” operation is sliding a window of features around the images.
- Higher “dot product” means higher correlations, and more likely to match features.
- Brute force search means we can find all possible actions! Hence, action-centric.
- But how do we get the features … ?

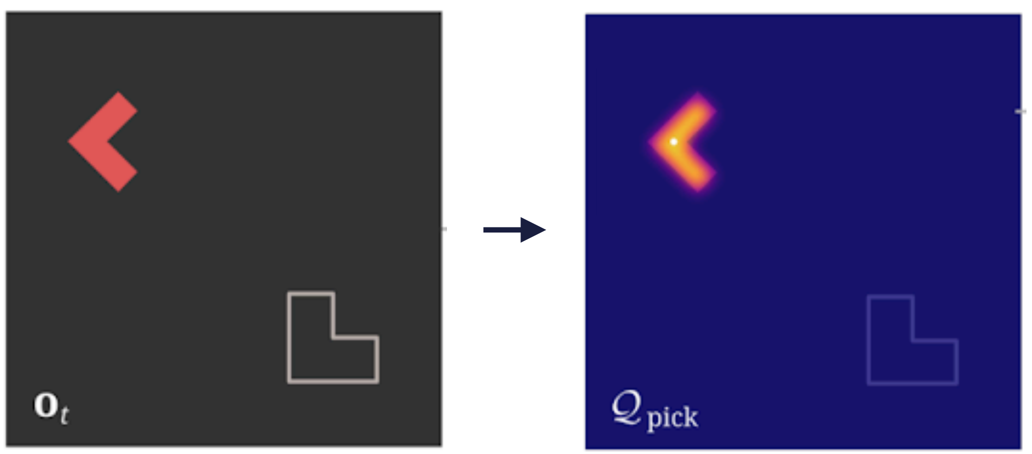
Transporter Networks



- Can compute these deep features via Fully Convolutional Neural Networks!
- Rotate the input crop, then these images are combined in the same minibatch.
- Important that the images are orthographic (not perspective), preserves spatial structure.
- Equivariance: $f_{\text{pick}}(g \circ \mathbf{o}_t) = g \circ f_{\text{pick}}(\mathbf{o}_t)$, where g is a translation

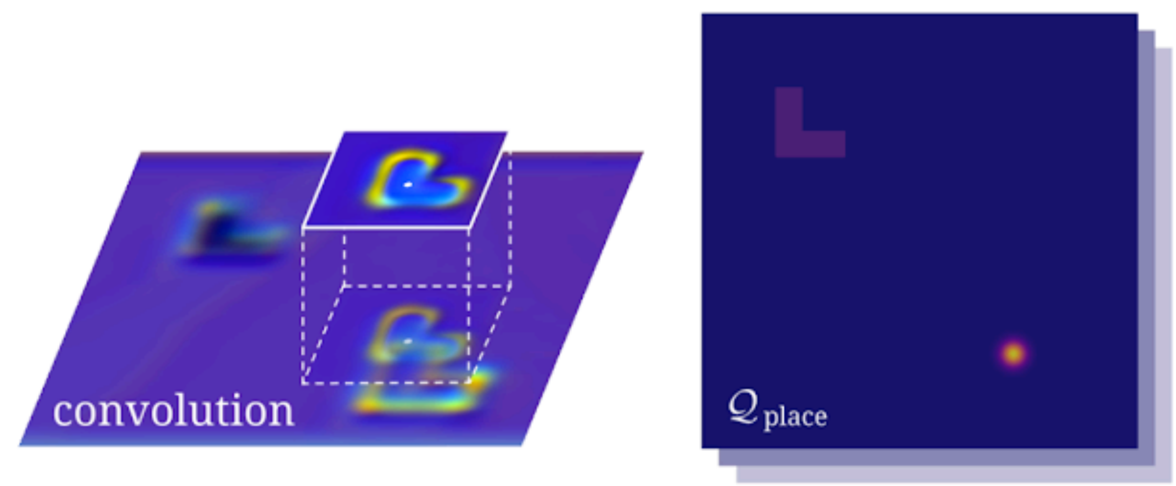
Transporter Networks for Pick and Place

1 Picking



$$f_{\text{pick}}(\mathbf{o}_t) \rightarrow \mathcal{T}_{\text{pick}}$$
$$\mathcal{T}_{\text{pick}} = \text{argmax}_{(u,v)}\ \mathcal{Q}_{\text{pick}}((u,v)|\mathbf{o}_t)$$
$$f_{\text{pick}}(g \circ \mathbf{o}_t) = g \circ f_{\text{pick}}(\mathbf{o}_t)$$

2 Pick-Conditioned Placing

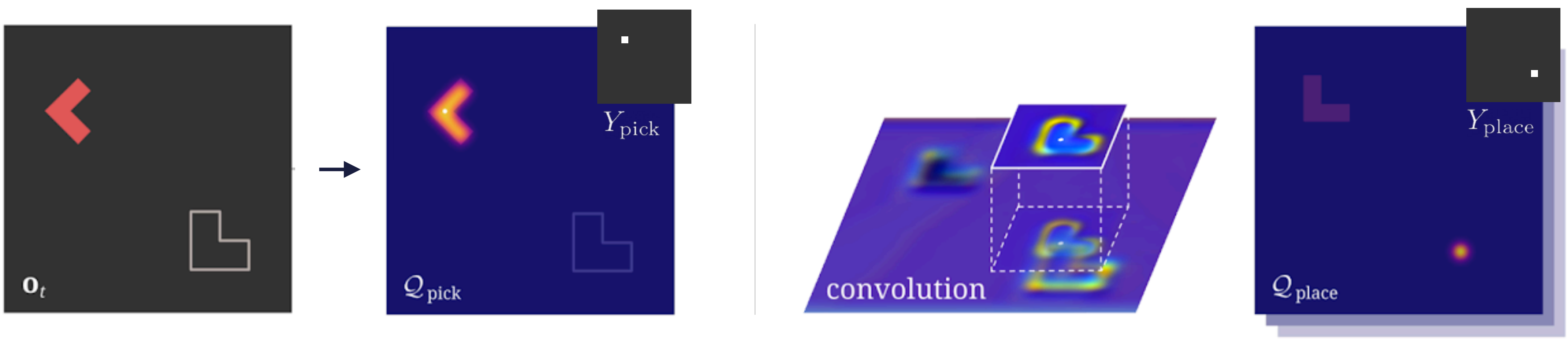


$$f_{\text{place}}(\mathbf{o}_t, \mathcal{T}_{\text{pick}}) \rightarrow \mathcal{T}_{\text{place}}$$
$$\mathcal{Q}_{\text{place}}(\tau|\mathbf{o}_t, \mathcal{T}_{\text{pick}}) = \psi(\mathbf{o}_t)[\mathcal{T}_{\text{pick}}] * \phi(\mathbf{o}_t)[\tau]$$
$$\mathcal{T}_{\text{place}} = \text{argmax}_{\{\tau_i\}}\ \mathcal{Q}_{\text{place}}(\tau_i|\mathbf{o}_t, \mathcal{T}_{\text{pick}})$$

- Heat maps show a distribution of picking and (pick-conditioned) placing points. These might be multi-modal and/or non-Gaussian!
- Uses implicit models, shown to be more robust than explicit models, see results for details and also follow-up work by Florence et al., Implicit Behavioral Cloning at CoRL 2021.
- FCNs are fast (one forward pass) and induces equivariance (helps data augmentation).

Transporter Networks for Pick and Place

① Picking ② Pick-Conditioned Placing



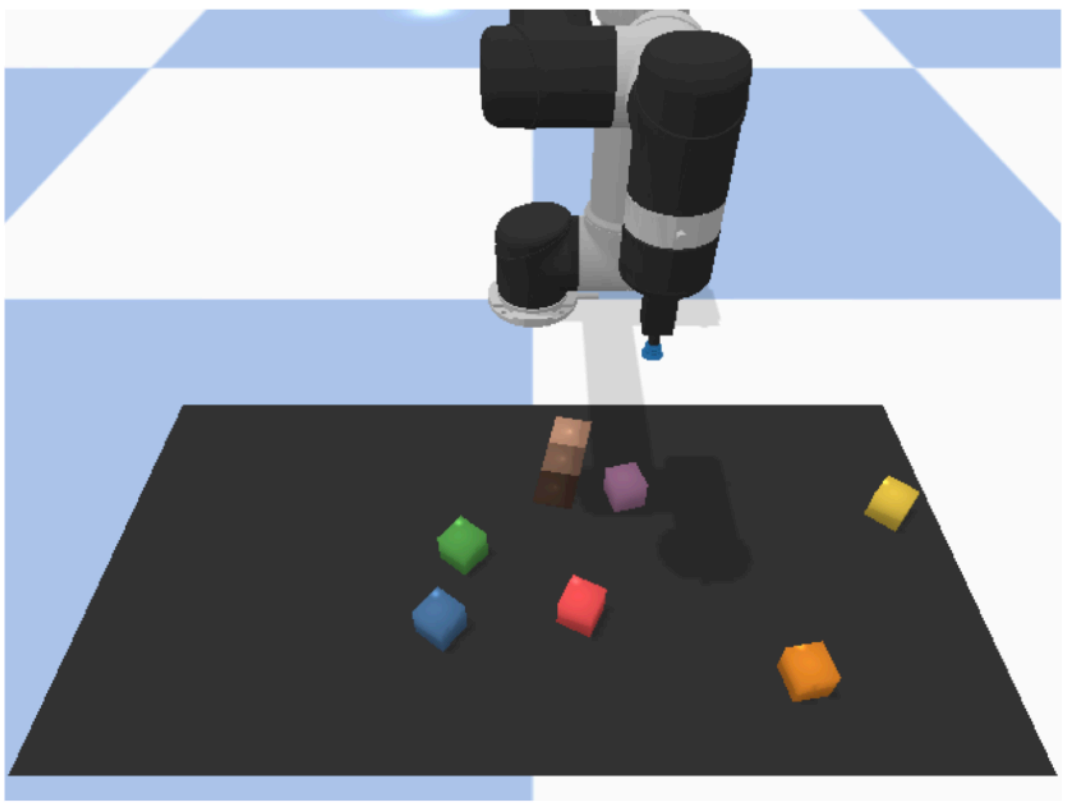
$$\mathcal{V}_{\text{pick}} \in \mathbb{R}^{H \times W} = \text{softmax}(\mathcal{Q}_{\text{pick}}((u, v)|\mathbf{o}_t))$$

$$\mathcal{V}_{\text{place}} \in \mathbb{R}^{H \times W \times k} = \text{softmax}(\mathcal{Q}_{\text{place}}(\tau|\mathbf{o}_t, \mathcal{T}_{\text{pick}}))$$

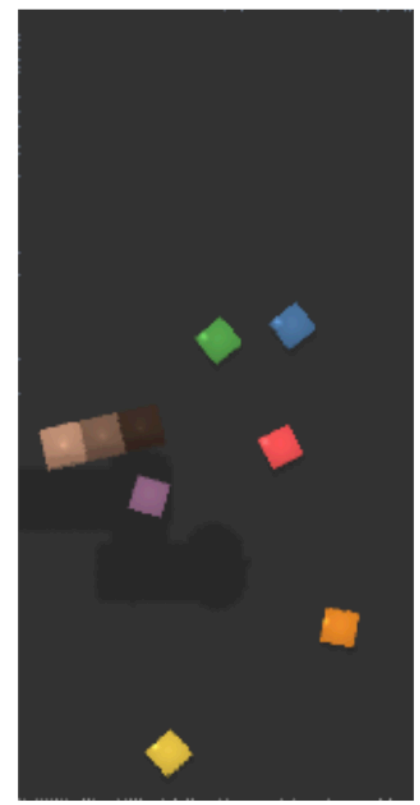
$$\mathcal{L} = -\mathbb{E}_{Y_{\text{pick}}}[\log \mathcal{V}_{\text{pick}}] - \mathbb{E}_{Y_{\text{place}}}[\log \mathcal{V}_{\text{place}}]$$

- How to train? Use behavioral cloning from pick-and-place demonstrations.
- Each pixel position (+ rotation) is a “class” in the discrete distribution of picking/placing points, and can train both networks with the standard cross-entropy loss.
- No need to have “negative samples” in training!

Data Augmentation



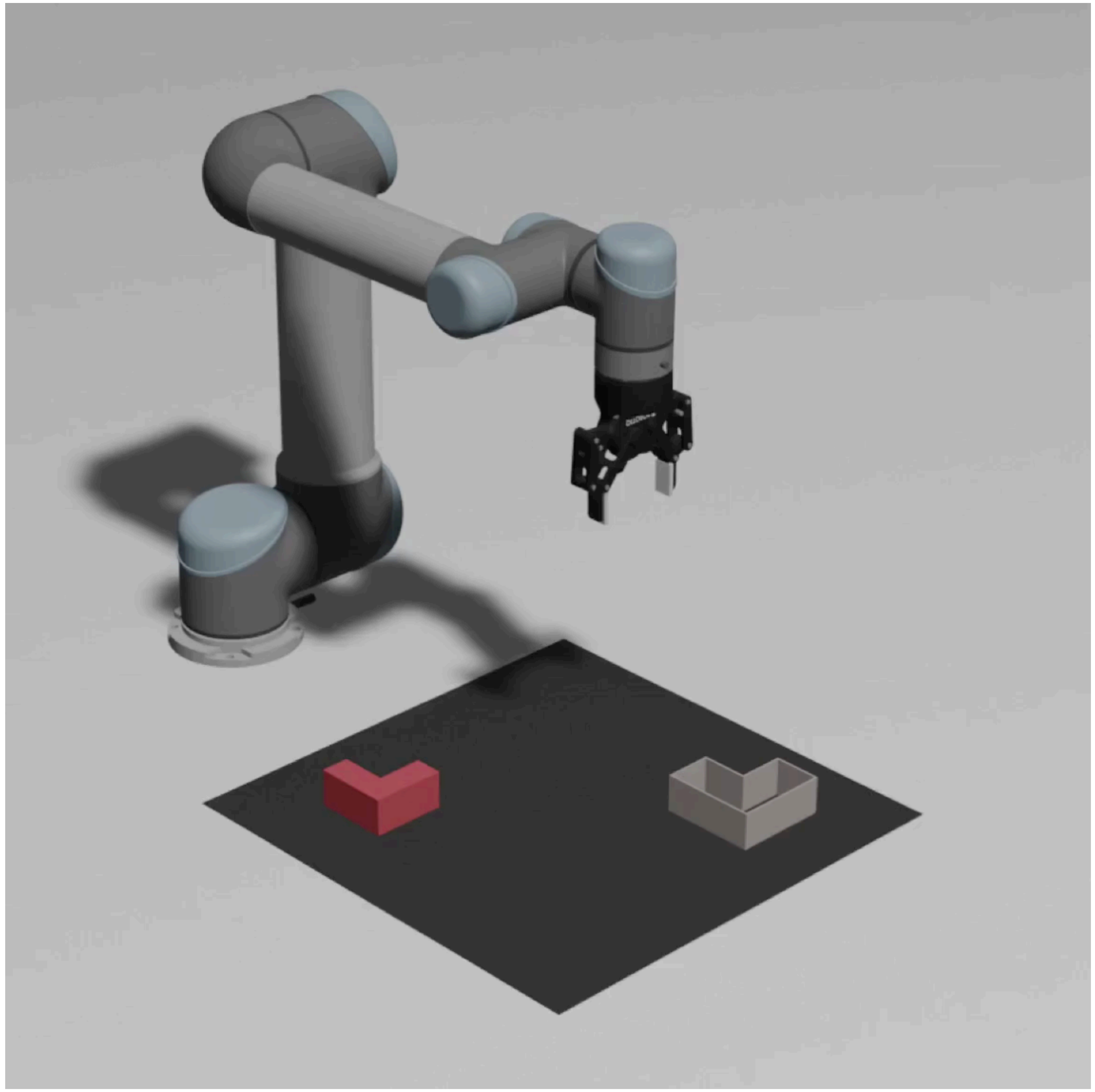
Perspective projection.



Orthographic top-down projection.

- Source: https://blog.kzakka.com/posts/representation/
- Note: while some visuals are square images to simplify presentation, in practice (in the paper) the input images are 160x320 RGB(D).

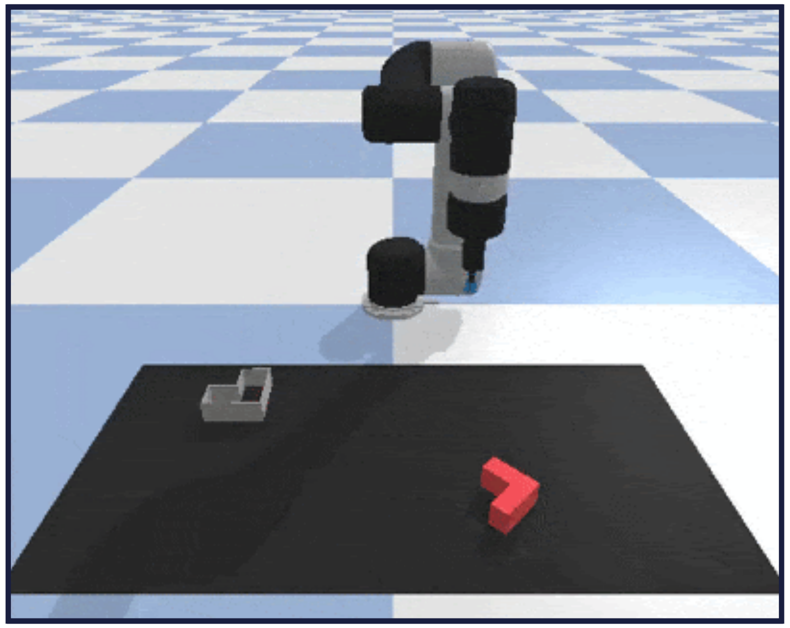
Visualization



Source: https://transporternets.github.io/

Transporter Nets for Pick and Place

Insert Block into Fixture

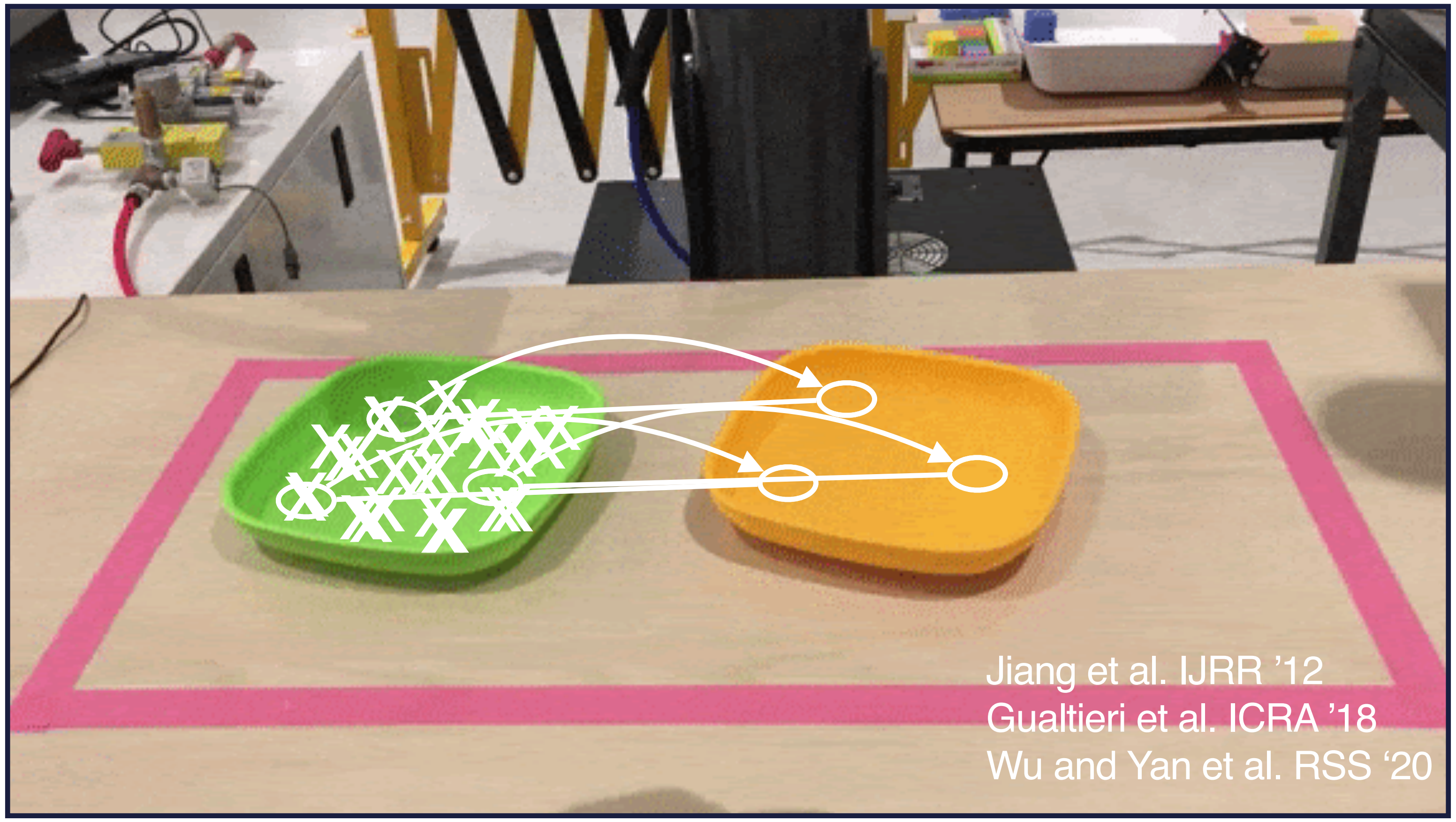


Align Box to Corner

Place Red in Green

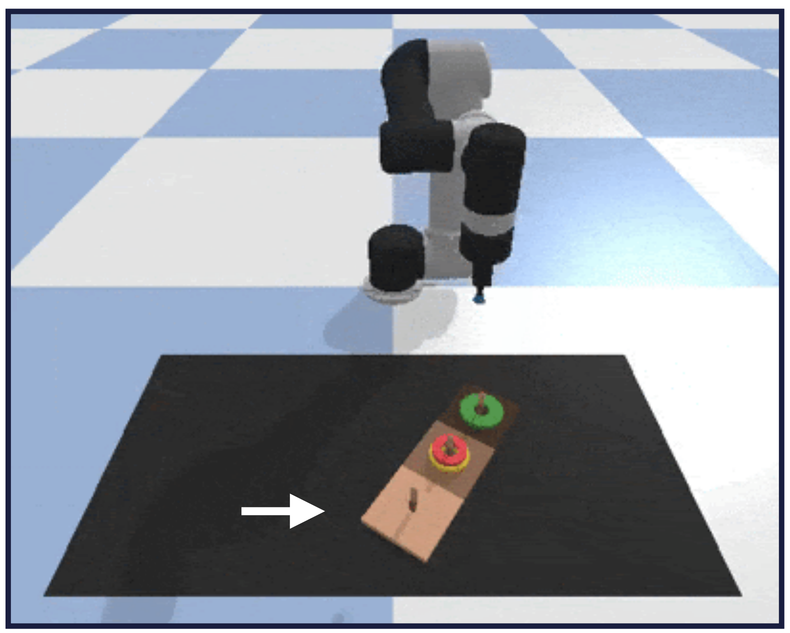
- Can do these reliably using as few as ~10 demonstrations.
- At test time, the same objects are sampled in different positions on the workspace.

Pick-Conditioned Placing

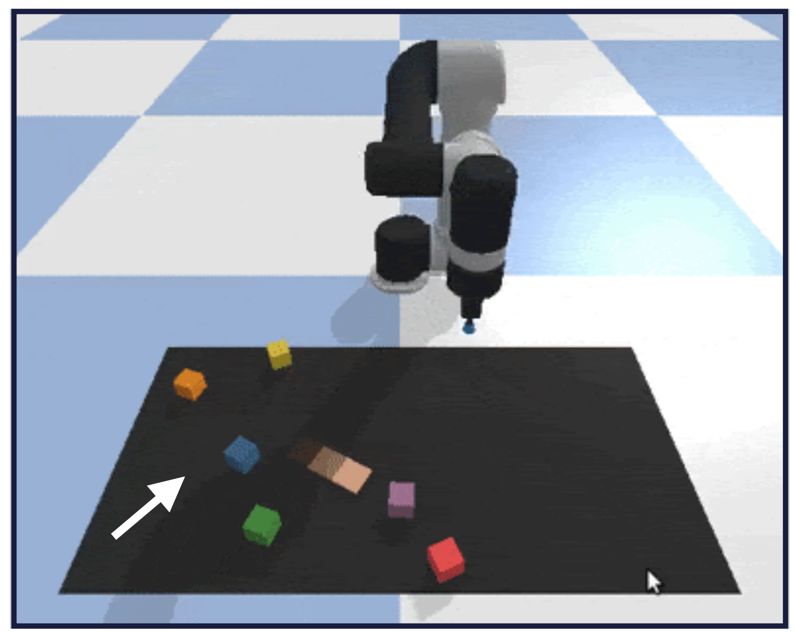


Sequential Multi-Step Tasks

Towers of Hanoi



Stacking Blocks

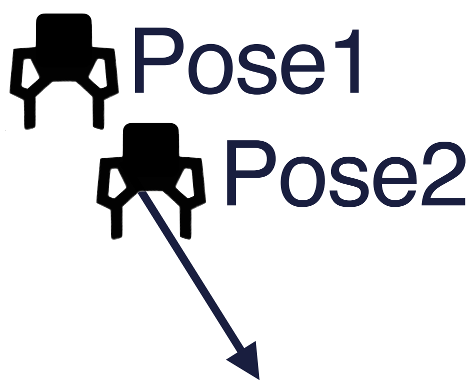


Assembling Kits

Unseen

Beyond Pick and Place

Two-Pose
Primitives



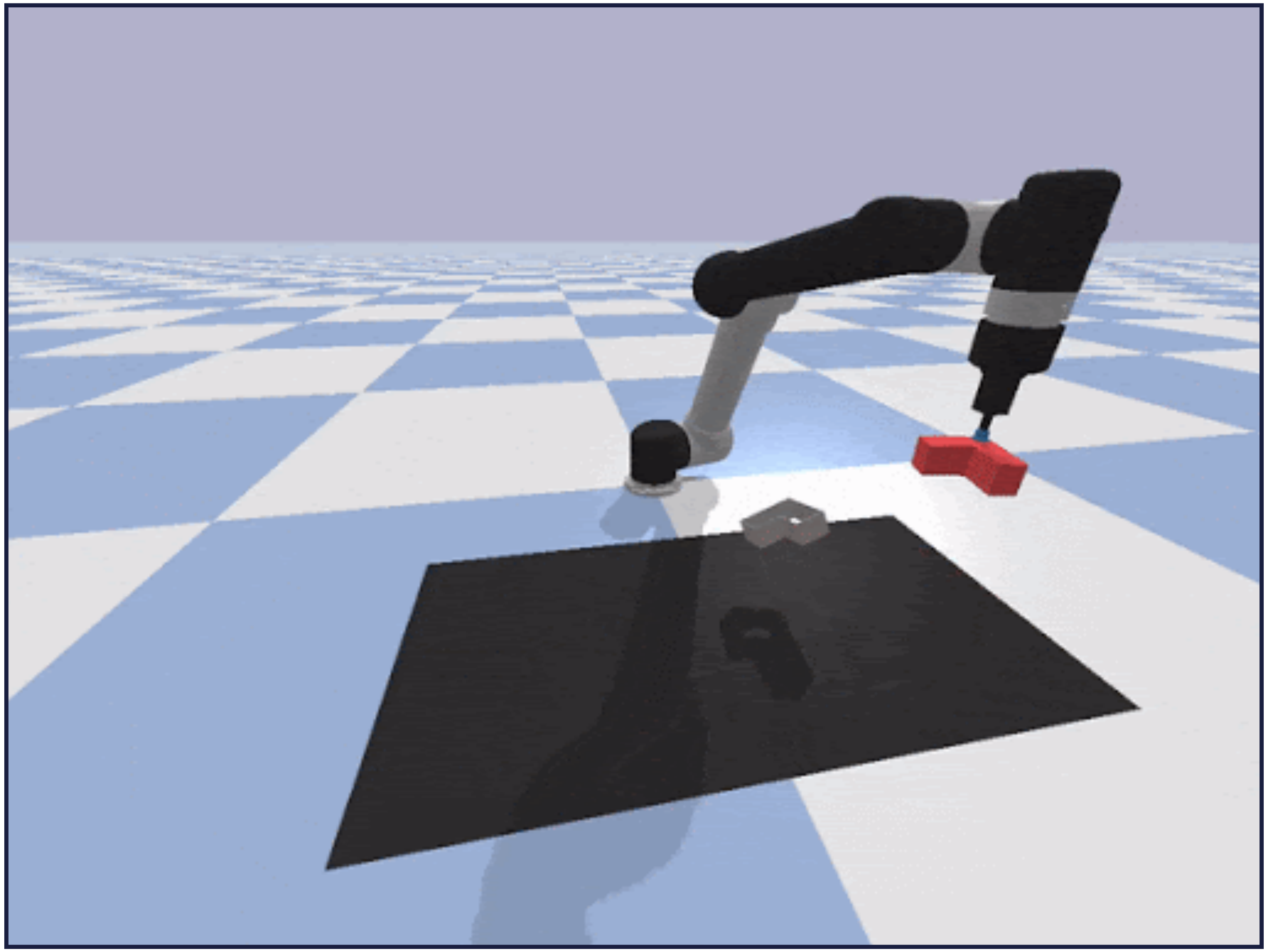
Sweeping Piles

Manipulating Rope

Note: we'll soon discuss work that explores deformable manipulation in much more detail!

Hybrid 6DoF Tasks

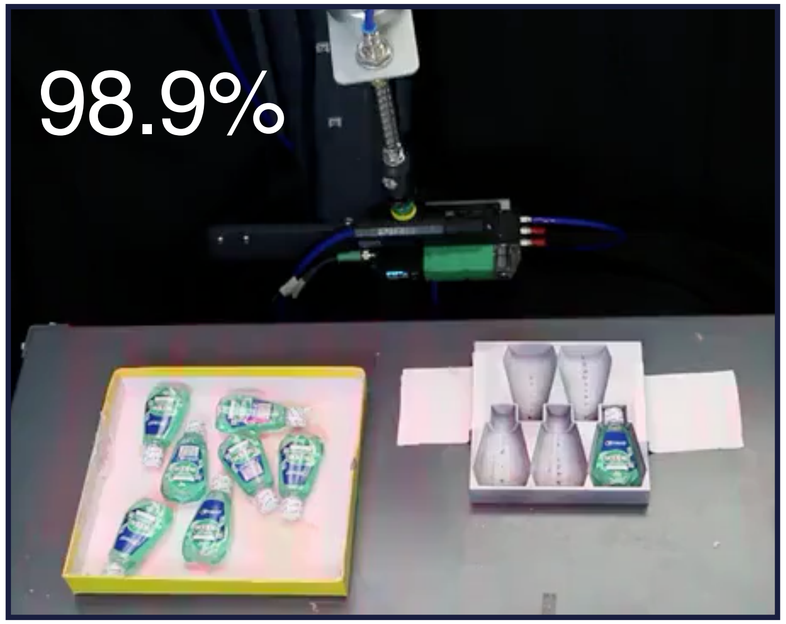
6DoF Block Insertion



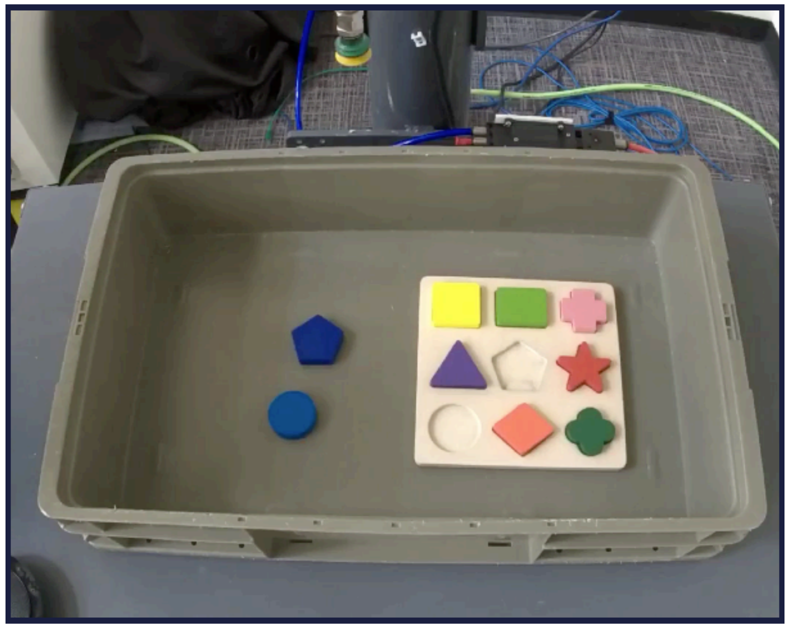
Approach: use Transporter Networks to infer SE(2) action, then regress remaining components.

Experiments on Real Robots

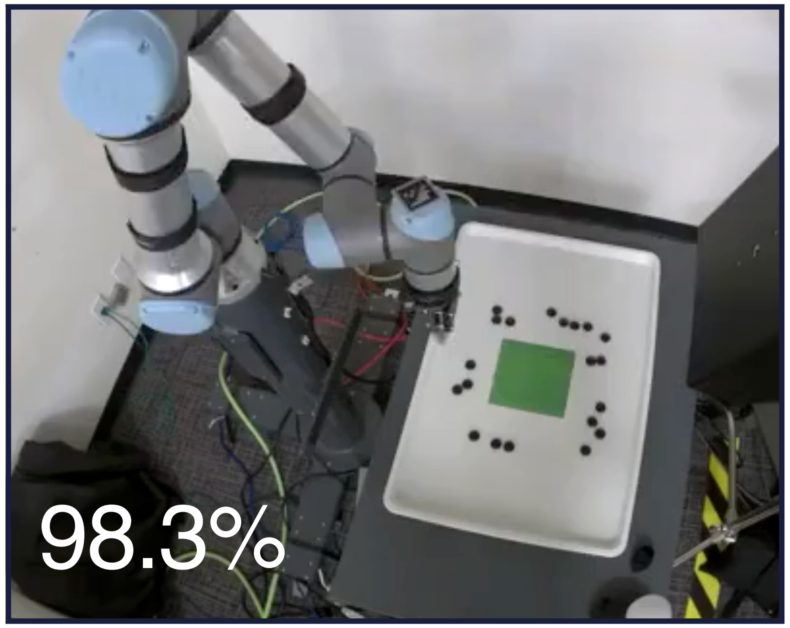
Kitting Small Bottles

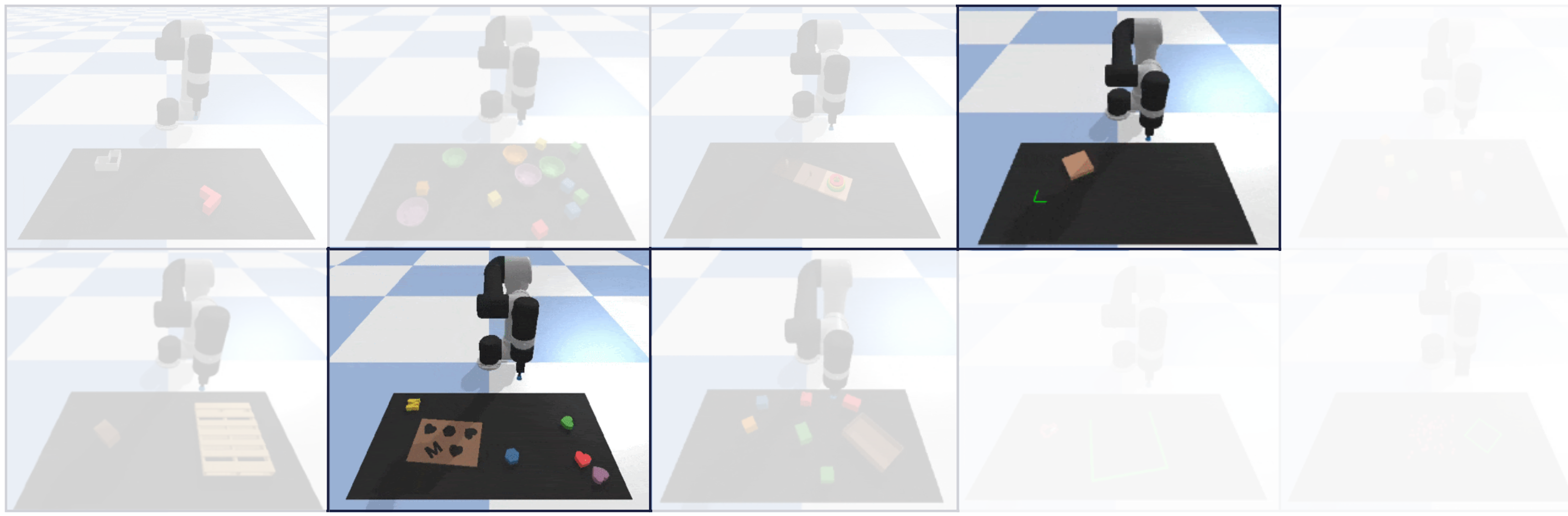


Kitting Wooden Pieces



Sweeping Go Pieces





Method	block-insertion				place-red-in-green				towers-of-hanoi				align-box-corner				stack-block-pyramid			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter Network	100	100	100	100	84.5	100	100	100	73.1	83.9	97.3	98.1	35.0	85.0	97.0	98.3	13.3	42.6	56.2	78.2
Form2Fit [22]	17.0	19.0	23.0	29.0	83.4	100	100	100	3.6	4.4	3.7	7.0	7.0	2.0	5.0	16.0	19.7	17.5	18.5	32.5
Conv. MLP	0.0	5.0	6.0	8.0	0.0	3.0	25.5	31.3	0.0	1.0	1.9	2.1	0.0	2.0	1.0	1.0	0.0	1.8	1.7	1.7
GT-State MLP	4.0	52.0	96.0	99.0	0.0	0.0	3.0	82.2	10.7	10.7	6.1	5.3	47.0	29.0	29.0	59.0	0.0	0.2	1.3	15.3
GT-State MLP 2-Step	6.0	38.0	95.0	100.0	0.0	0.0	19.0	92.8	22.0	6.4	5.6	3.1	49.0	12.0	43.0	55.0	0.0	0.8	12.2	17.5

Method	palletizing-boxes				assembling-kits				packing-boxes				manipulating-rope				sweeping-piles			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter Network	63.2	77.4	91.7	97.9	28.4	78.6	90.4	94.6	56.8	58.3	72.1	81.3	21.9	73.2	85.4	92.1	52.4	74.4	71.5	96.1
Form2Fit [22]	21.6	42.0	52.1	65.3	3.4	7.6	24.2	37.6	29.9	52.5	62.3	66.8	11.9	38.8	36.7	47.7	13.2	15.6	26.7	38.4
Conv. MLP	31.4	37.4	34.6	32.0	0.0	0.2	0.2	0.0	0.3	9.5	12.6	16.1	3.7	6.6	3.8	10.8	28.2	48.4	44.9	45.1
GT-State MLP	0.6	6.4	30.2	30.1	0.0	0.0	1.2	11.8	7.1	1.4	33.6	56.0	5.5	11.5	43.6	47.4	7.2	20.6	63.2	74.4
GT-State MLP 2-Step	0.6	9.6	32.8	37.5	0.0	0.0	1.6	4.4	4.0	3.5	43.4	57.1	6.0	8.2	41.5	58.7	9.7	21.4	66.2	73.9

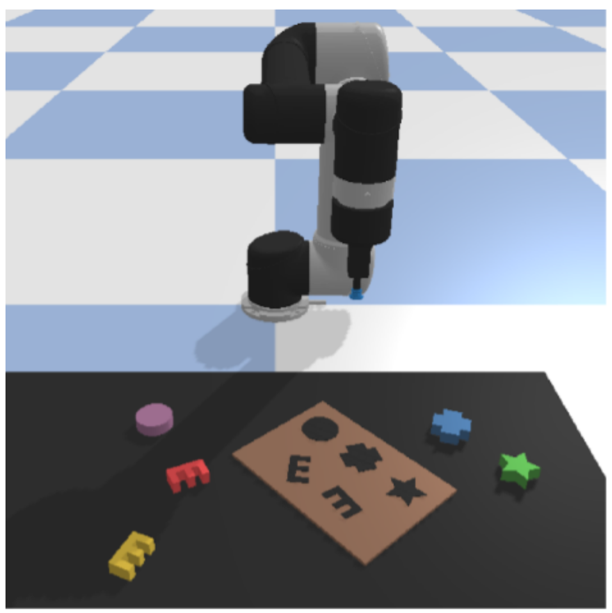
Table 2. **Baseline comparisons.** Task performance (mean %) vs. # of demonstration episodes (1, 10, 100, or 1000) used in training.

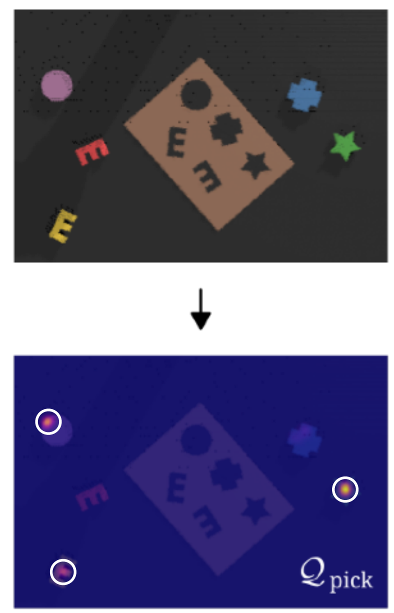
Analysis: Learned Multimodal Actions

Assembling Kits of Unseen Objects

Picking Predictions

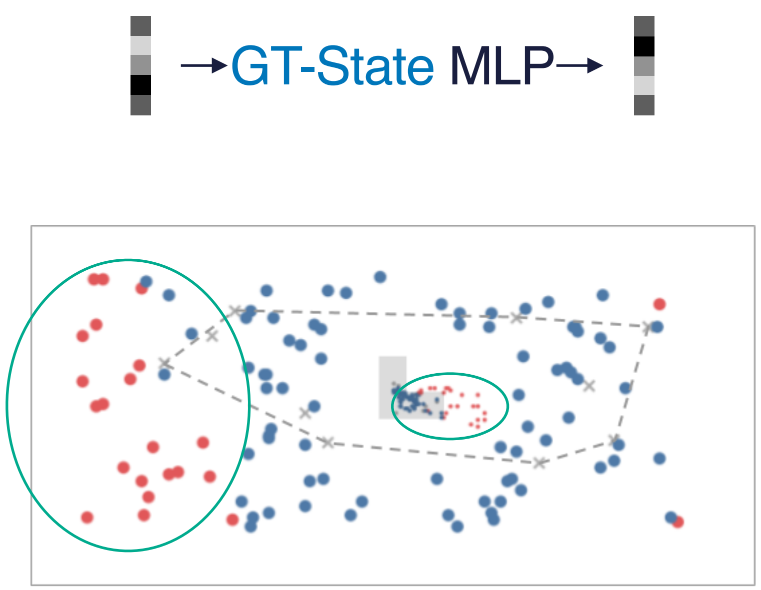
Placing Predictions



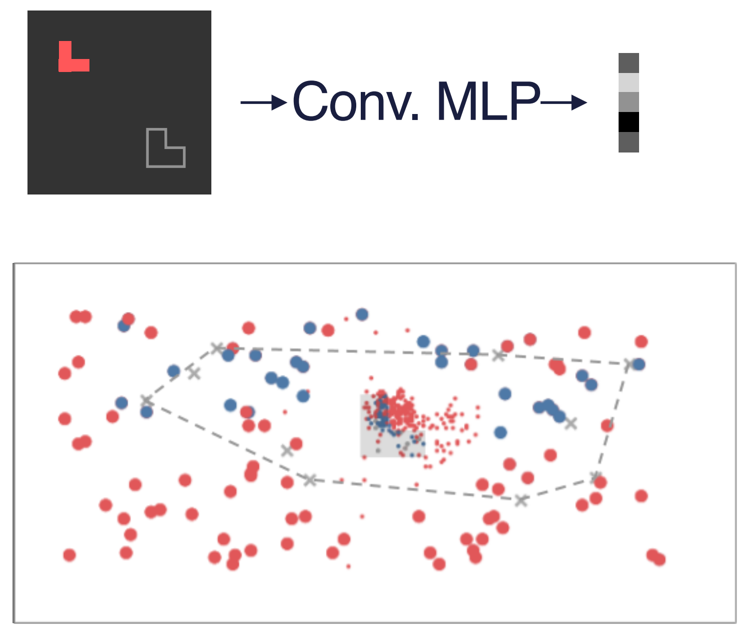


- Given one picking point, there may be a distribution of possible (valid) placing spots.
- The “heat maps" on the images specify the distribution!

Analysis: Interpolation and Extrapolation

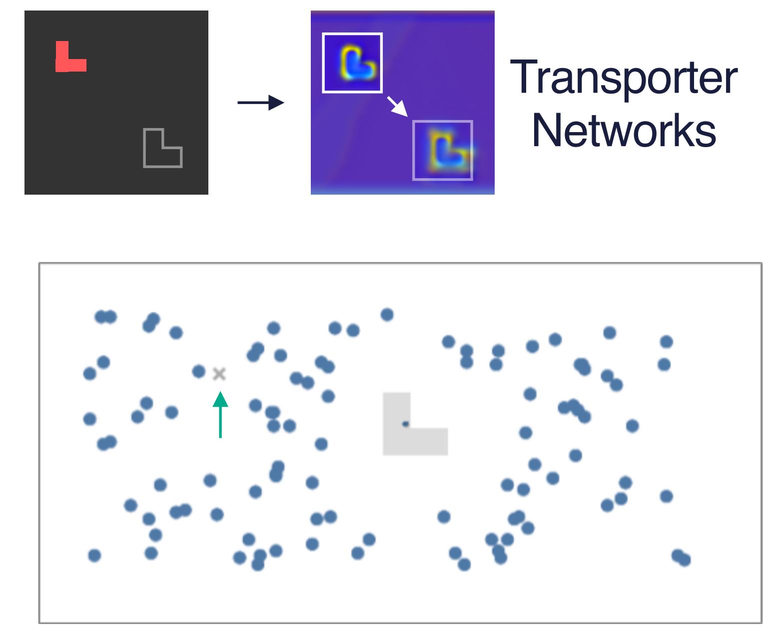


10 Demonstrations



10 Demonstrations

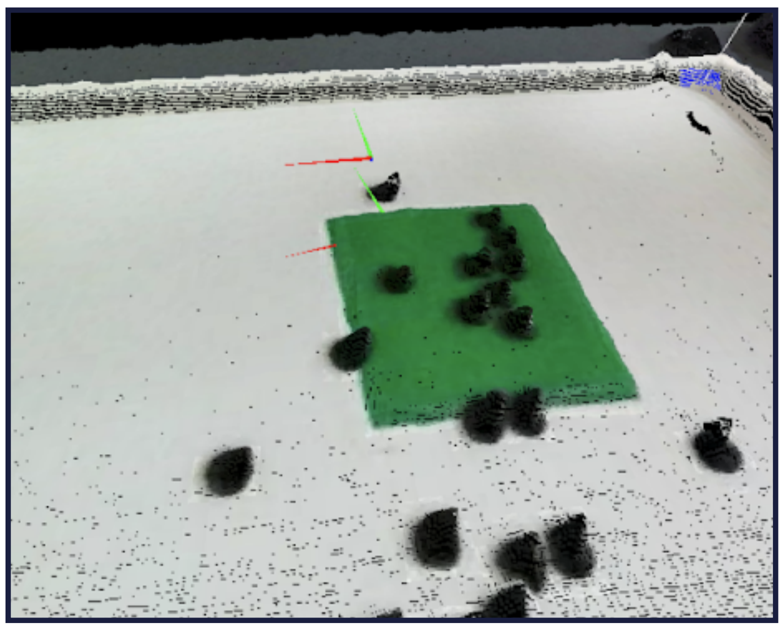
success or failure



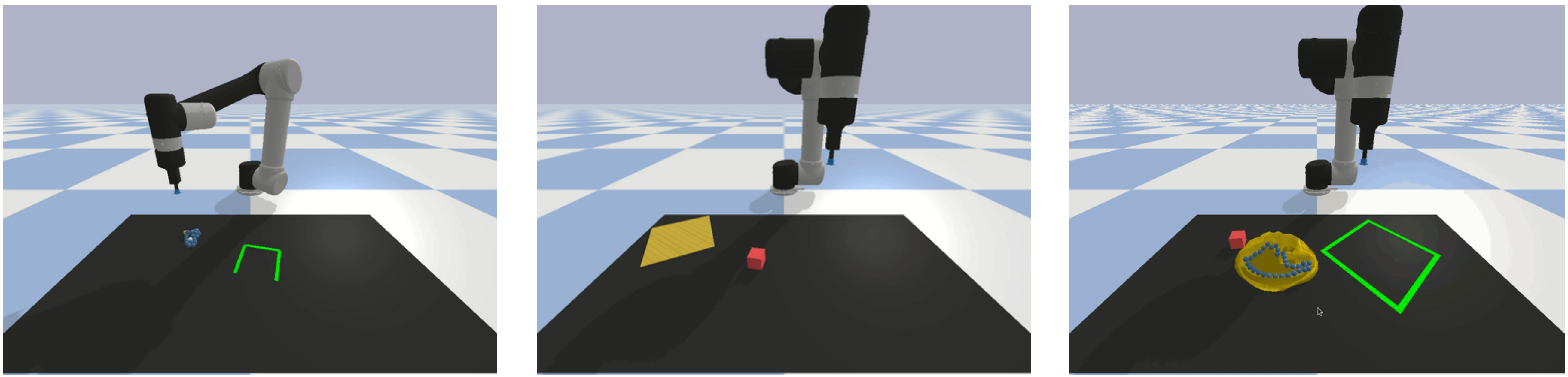
1 demonstrations

Limitations

Noisy 3D Data



Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks

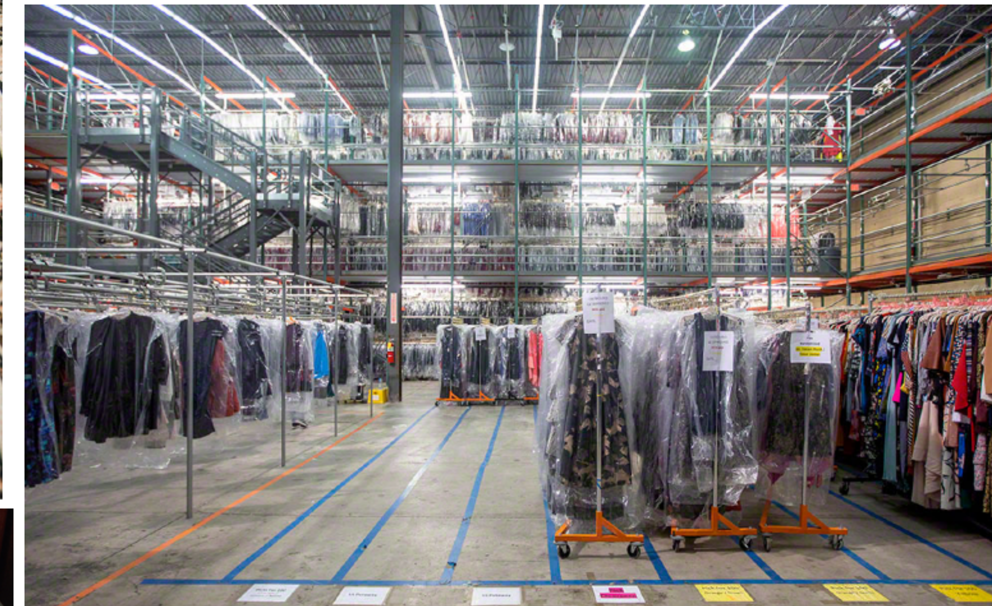


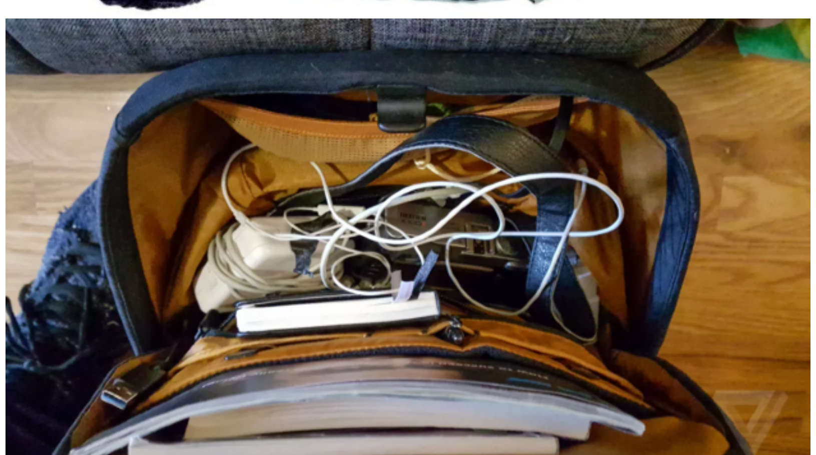
Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, Andy Zeng

AUTOLAB
Google AI
BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

International Conference on Robotics and Automation (ICRA) 2021

Our Starting Motivation: Deformable Manipulation



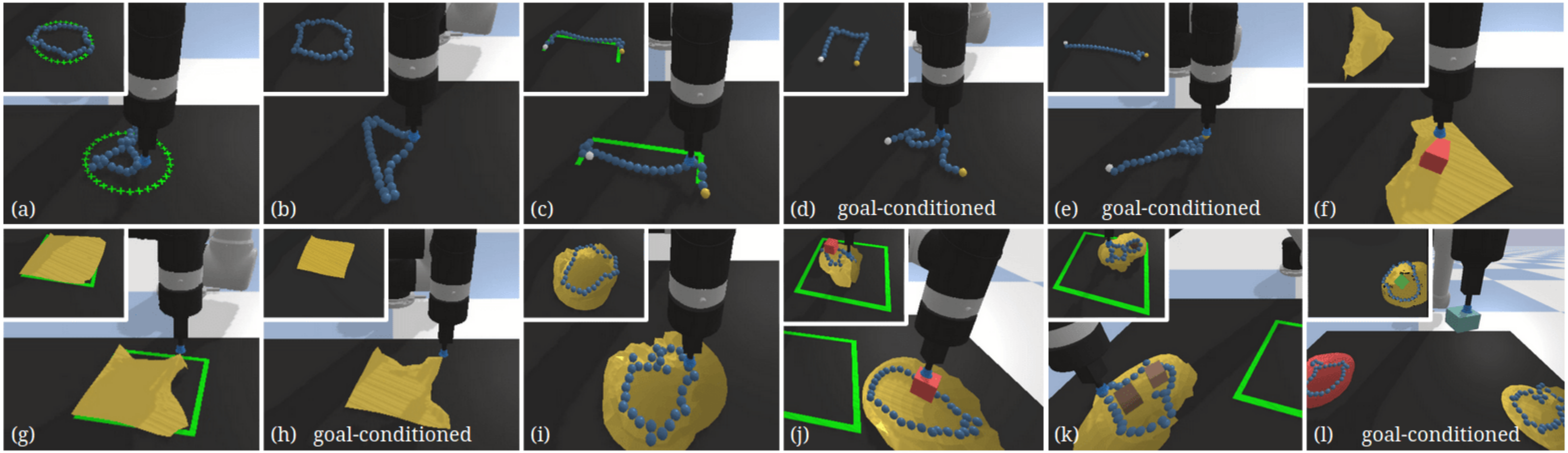


Main Contributions

- A suite of 12 simulated tasks in PyBullet spanning cables, fabrics, and bags.
- Model architectures for manipulating objects towards desired goal configurations, specified with images [might be suited for deformables].

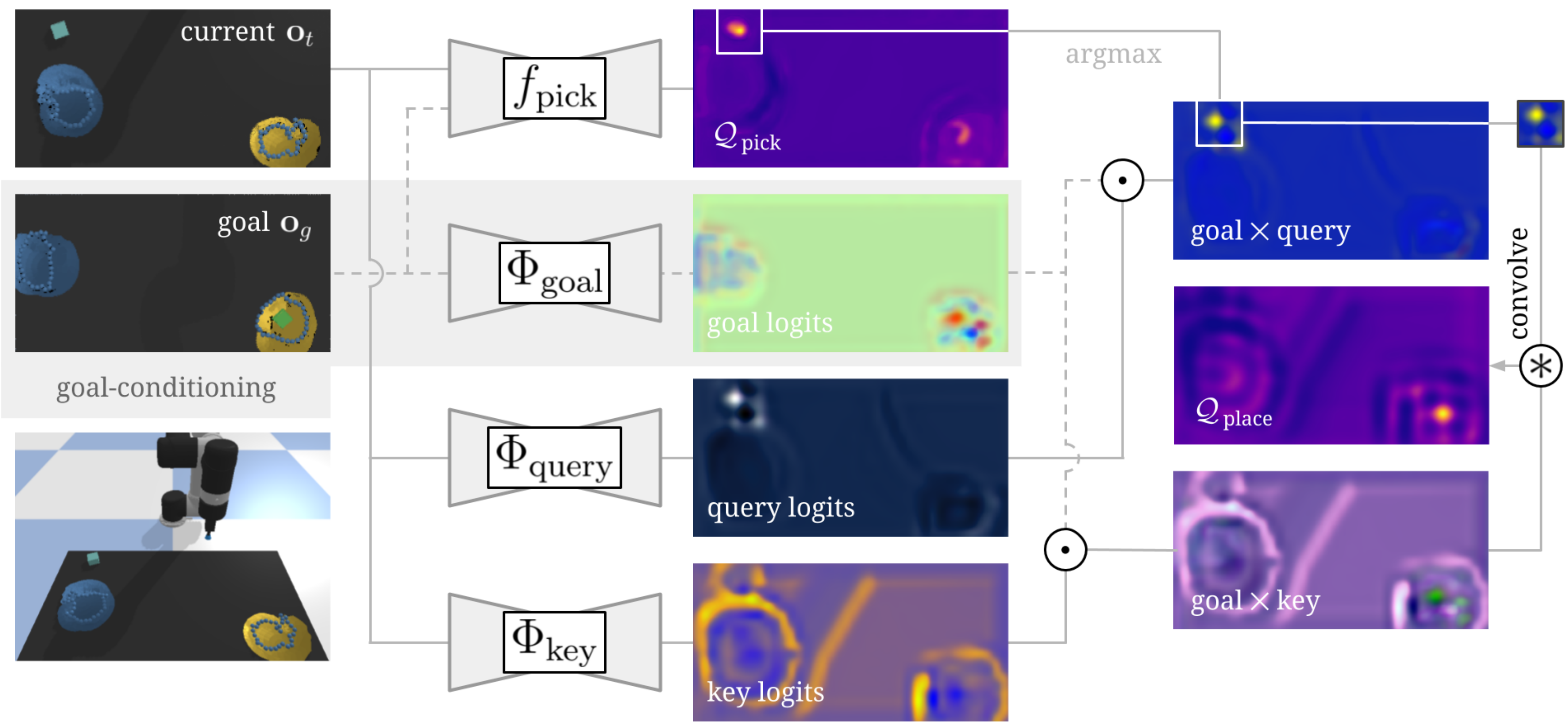
Main Contributions

- **A suite of 12 simulated tasks in PyBullet spanning cables, fabrics, and bags.**
- Model architectures for manipulating objects towards desired goal configurations, specified with images [might be suited for deformables].



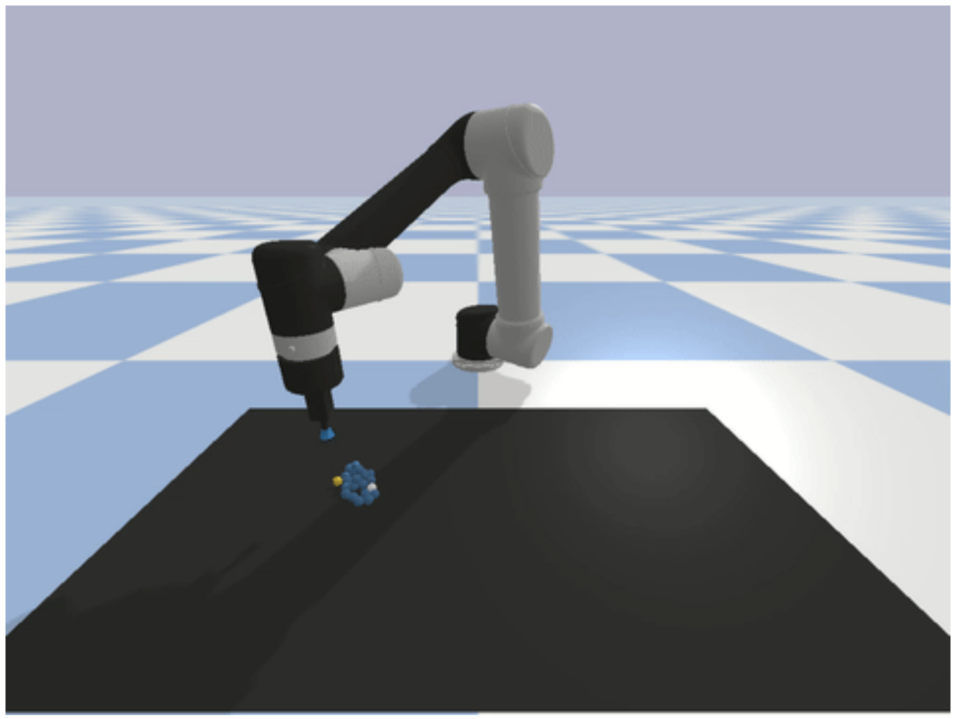
Main Contributions

- A suite of 12 simulated tasks in PyBullet spanning cables, fabrics, and bags.
- **Model architectures for manipulating objects towards desired goal configurations, specified with images [might be suited for deformables].**

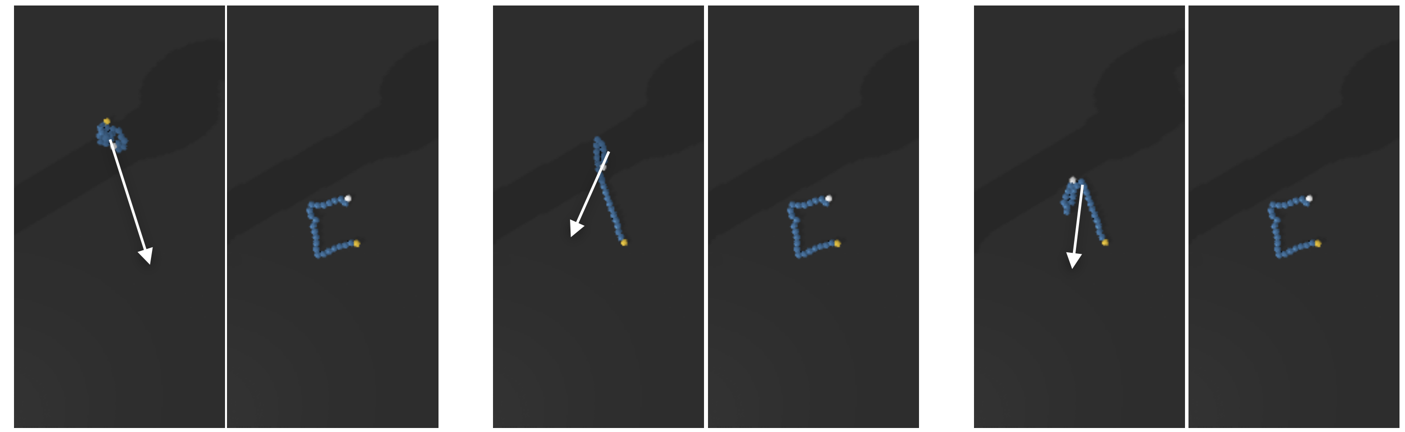


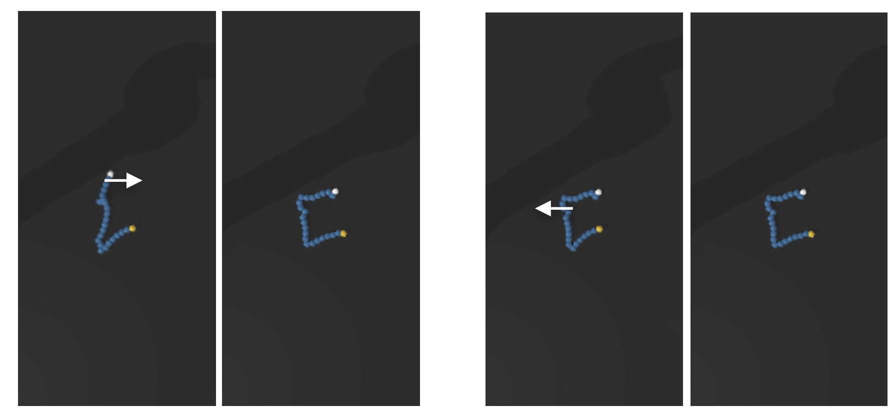
Tasks with Cables (1D Deformables)

Demonstrator

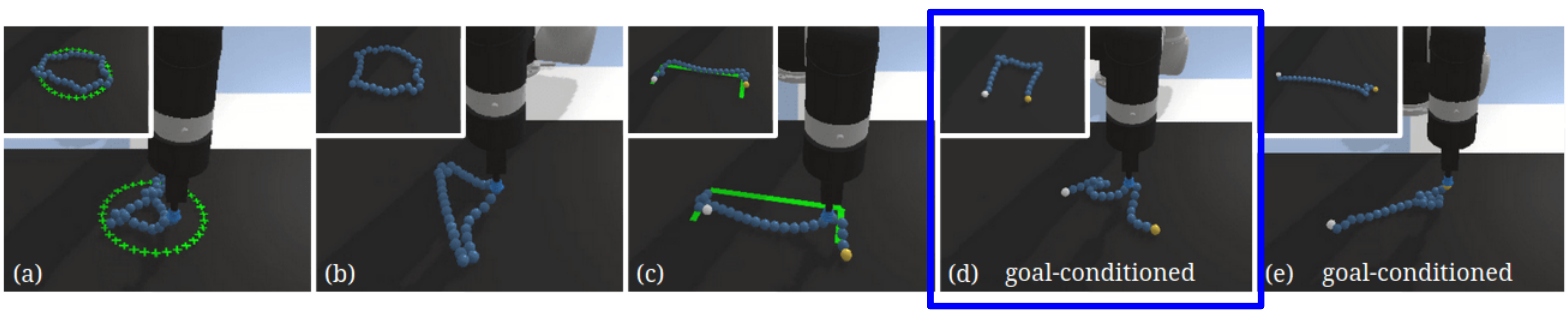


Demonstration Data



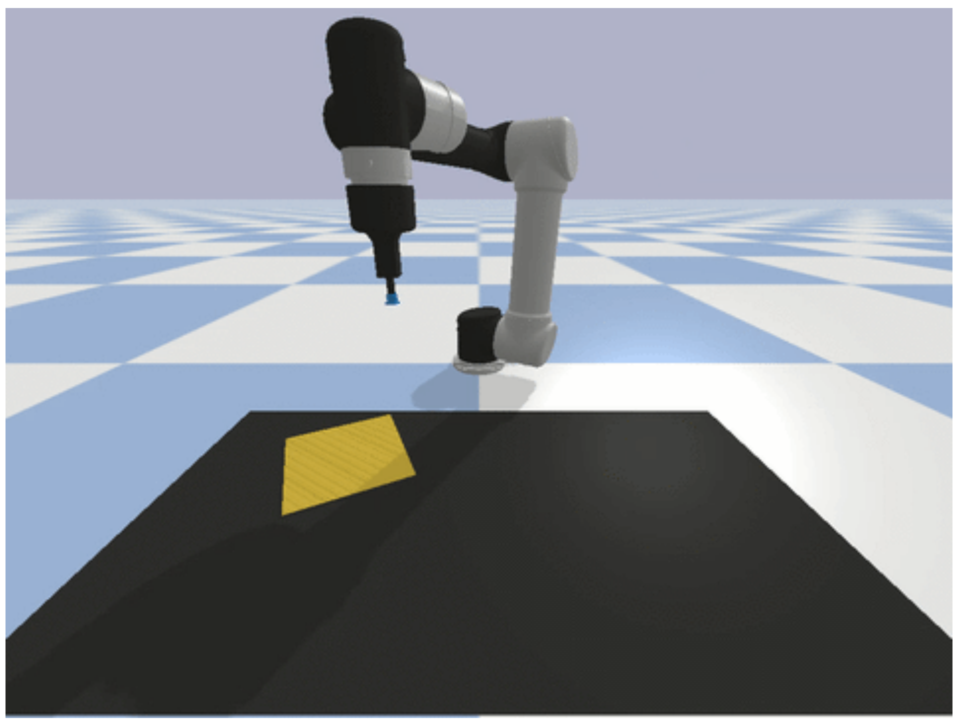


Demonstration Data
Each data point has a fixed goal image in addition to current image + action.

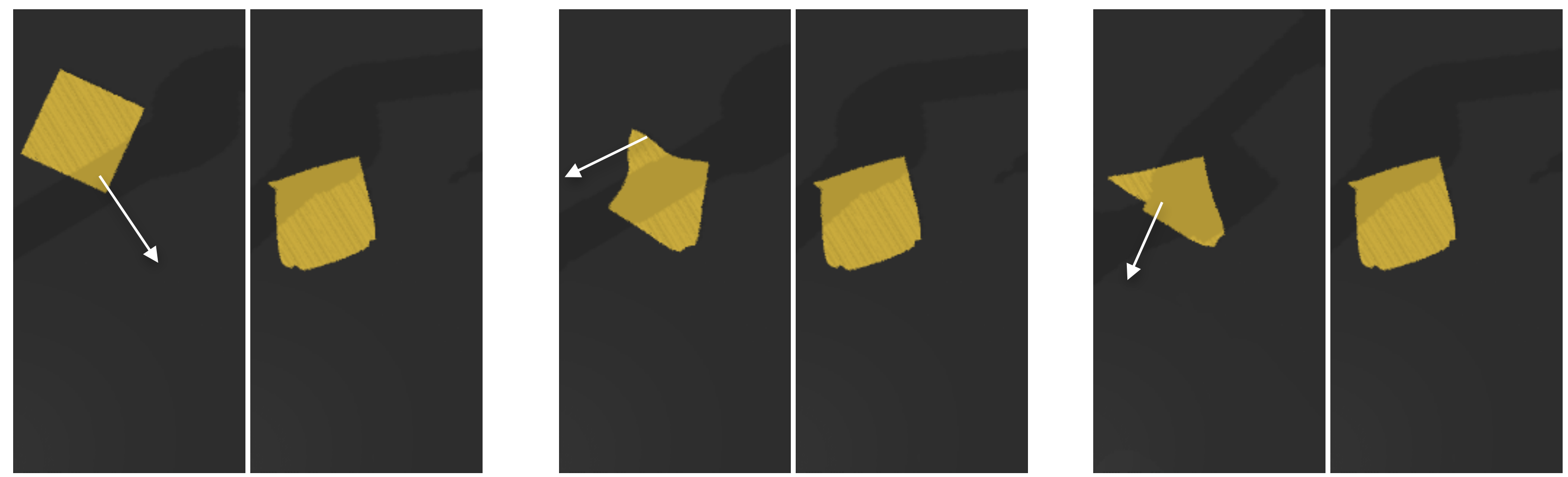


Tasks with Fabrics (2D Deformables)

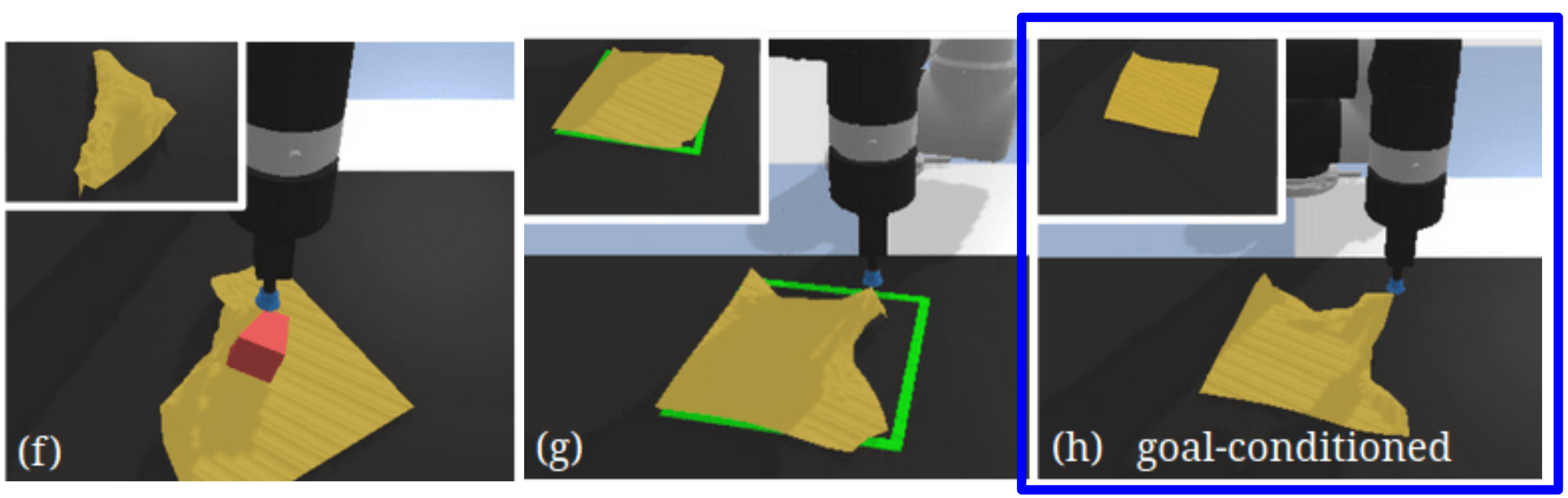
Demonstrator



Demonstration Data

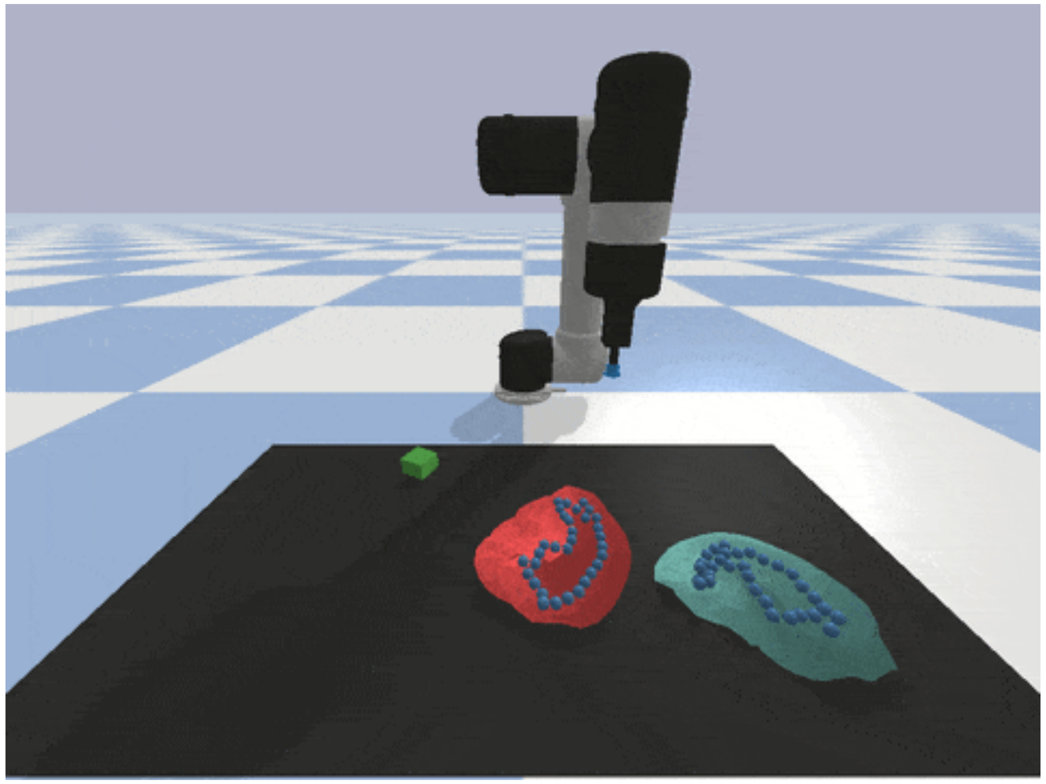


Demonstration Data
Each data point has a fixed goal image in addition to current image + action.

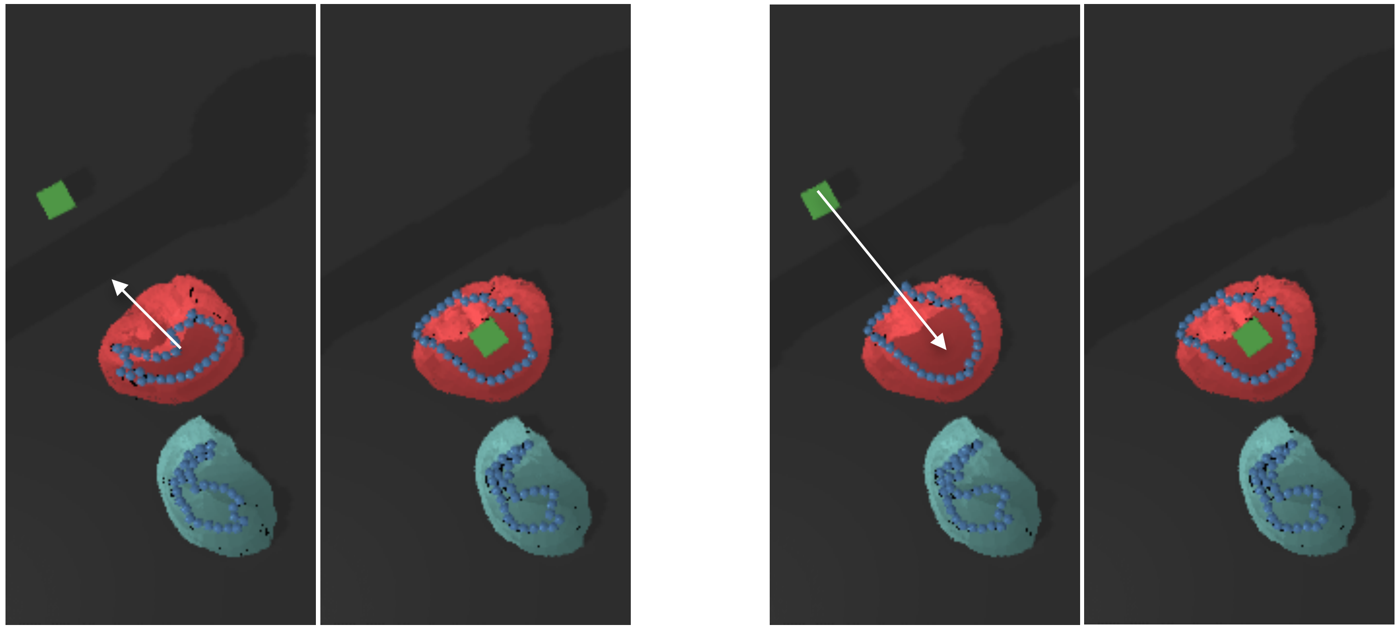


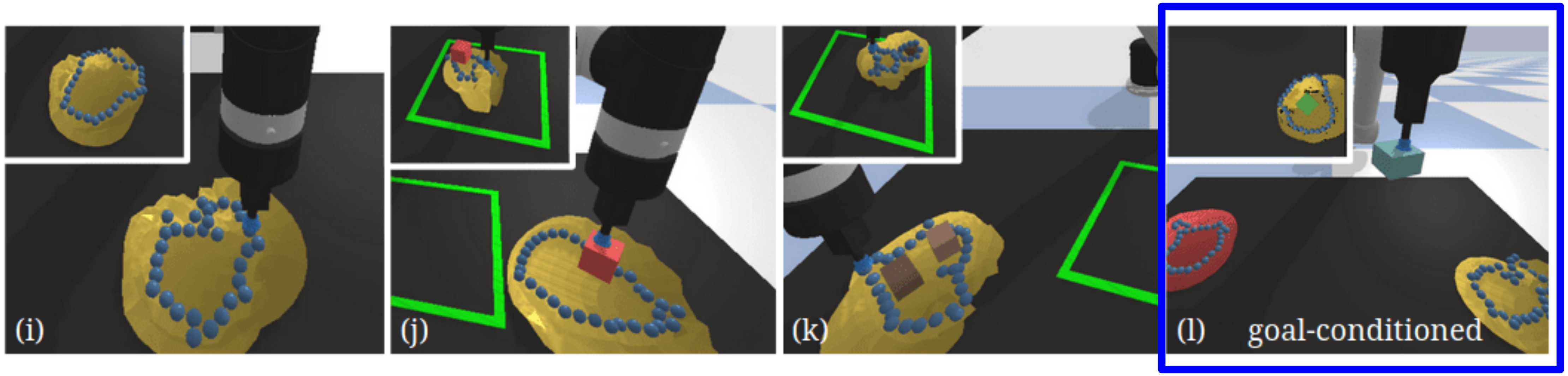
Tasks with Bags (3D Deformables)

Demonstrator

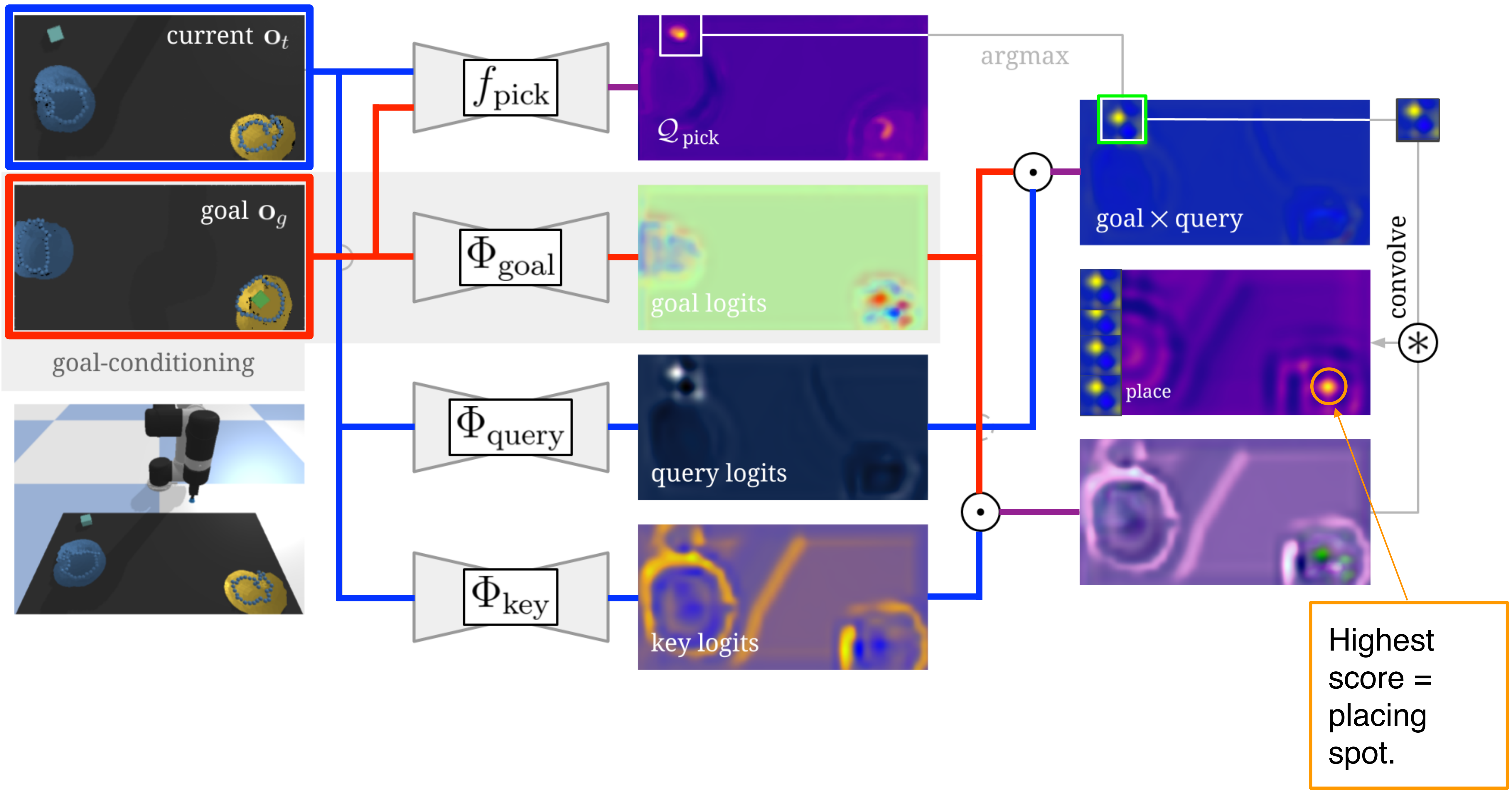


Demonstration Data



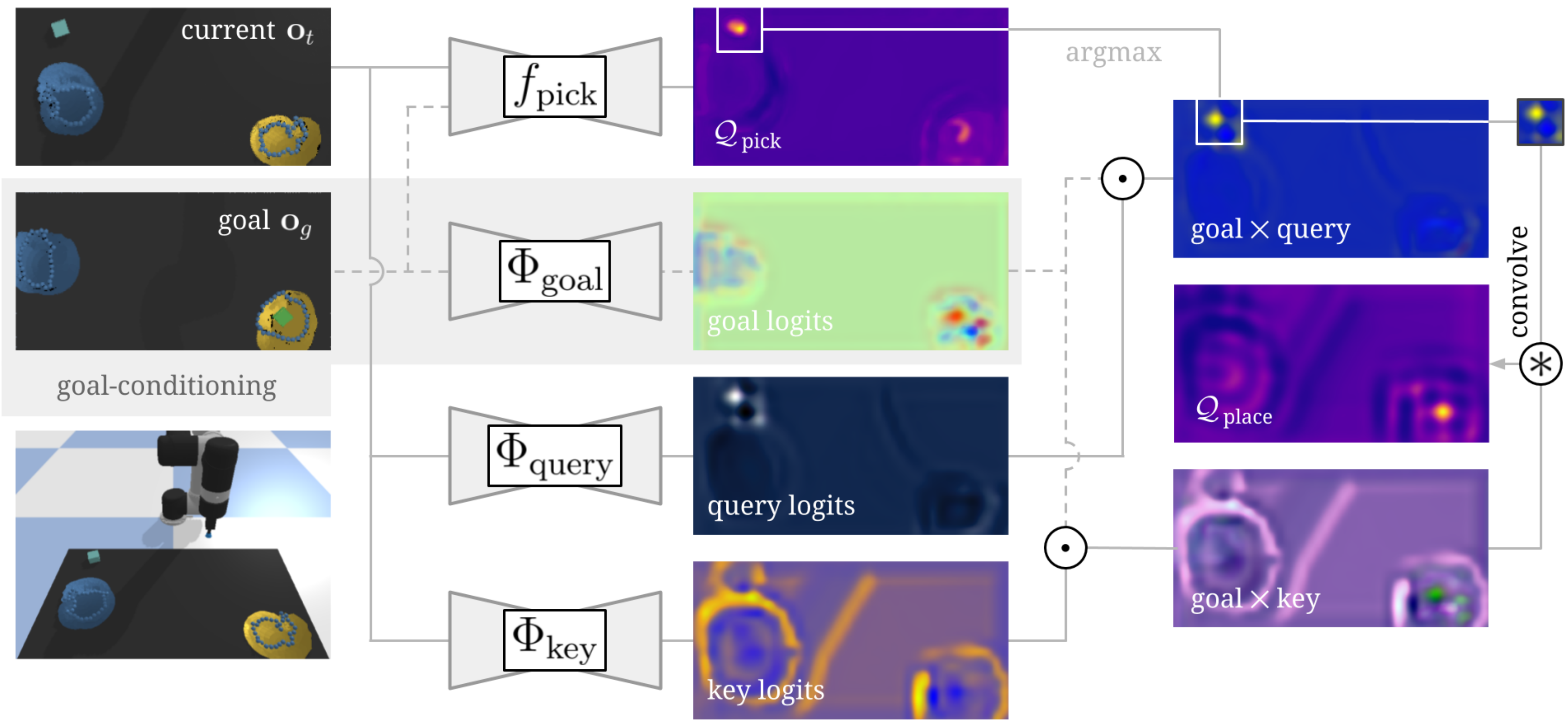


We Bring Goal Conditioning into Transporter Networks



Naive approach: stack current + goal images together channel-wise.

Training Procedure



- For each task, assume we have demonstrations of state-action (s,a) pairs.
- For GCTN, bring the <u>last</u> observation from each episode into the training sample as a second observation, (s,a,s'). Note the assumption this makes!

Training Procedure

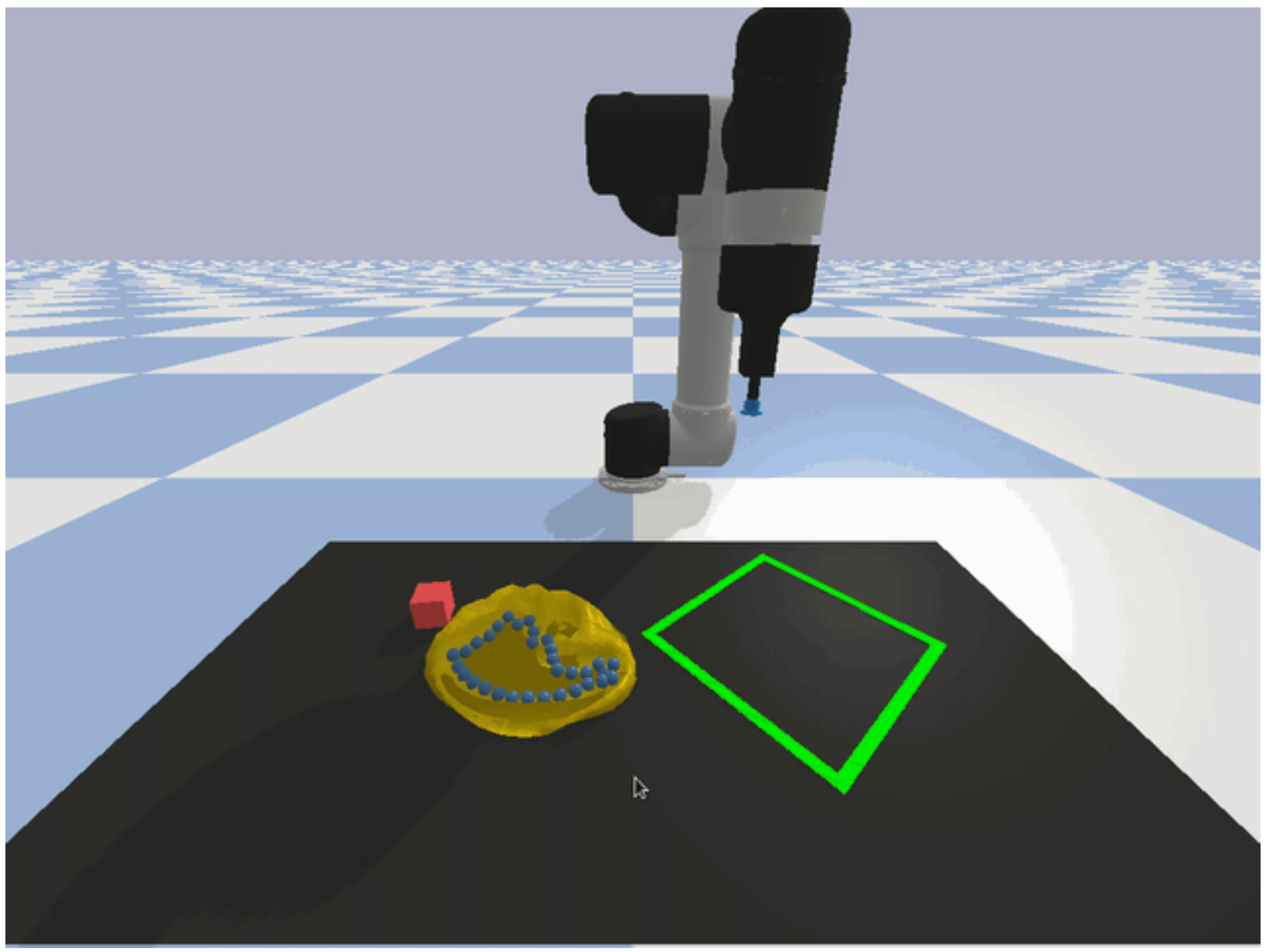
TABLE I: **DeformableRavens**. Tasks involve rearranging deformable objects (e.g., cables, fabrics, and bags). Each comes with a scripted expert demonstrator that succeeds with high probability, except for the four bag tasks which are challenging; for these, we filter to use only successful episodes in training. Some require *precise placing* to trigger a success. Tasks with a *visible zone* will have a green target zone on the workspace to indicate where items should be placed (e.g., a square target zone that a fabric must cover); other *goal-conditioned* tasks use a separate goal image to specify the success criteria for object rearrangement. See Figure 2 for visualizations.

Task (Max. Episode Length)	demos stats(%)	precise placing	visible zone	goal cond.
(a) cable-ring§ (20)	99.1	✗	✓	✗
(b) cable-ring-notarget§ (20)	99.3	✗	✗	✗
(c) cable-shape* (20)	98.8	✓	✓	✗
(d) cable-shape-notarget* (20)	99.1	✓	✗	✓
(e) cable-line-notarget* (20)	100.0	✓	✗	✓
(f) fabric-cover (2)	97.0	✗	✗	✗
(g) fabric-flat† (10)	98.3	✓	✓	✗
(h) fabric-flat-notarget† (10)	97.4	✓	✗	✓
(i) bag-alone-open§ (8)	60.2	✗	✗	✗
(j) bag-items-1 (8)	41.7	✗	✓	✗
(k) bag-items-2 (9)	32.5	✗	✓	✗
(l) bag-color-goal (8)	89.1	✗	✗	✓

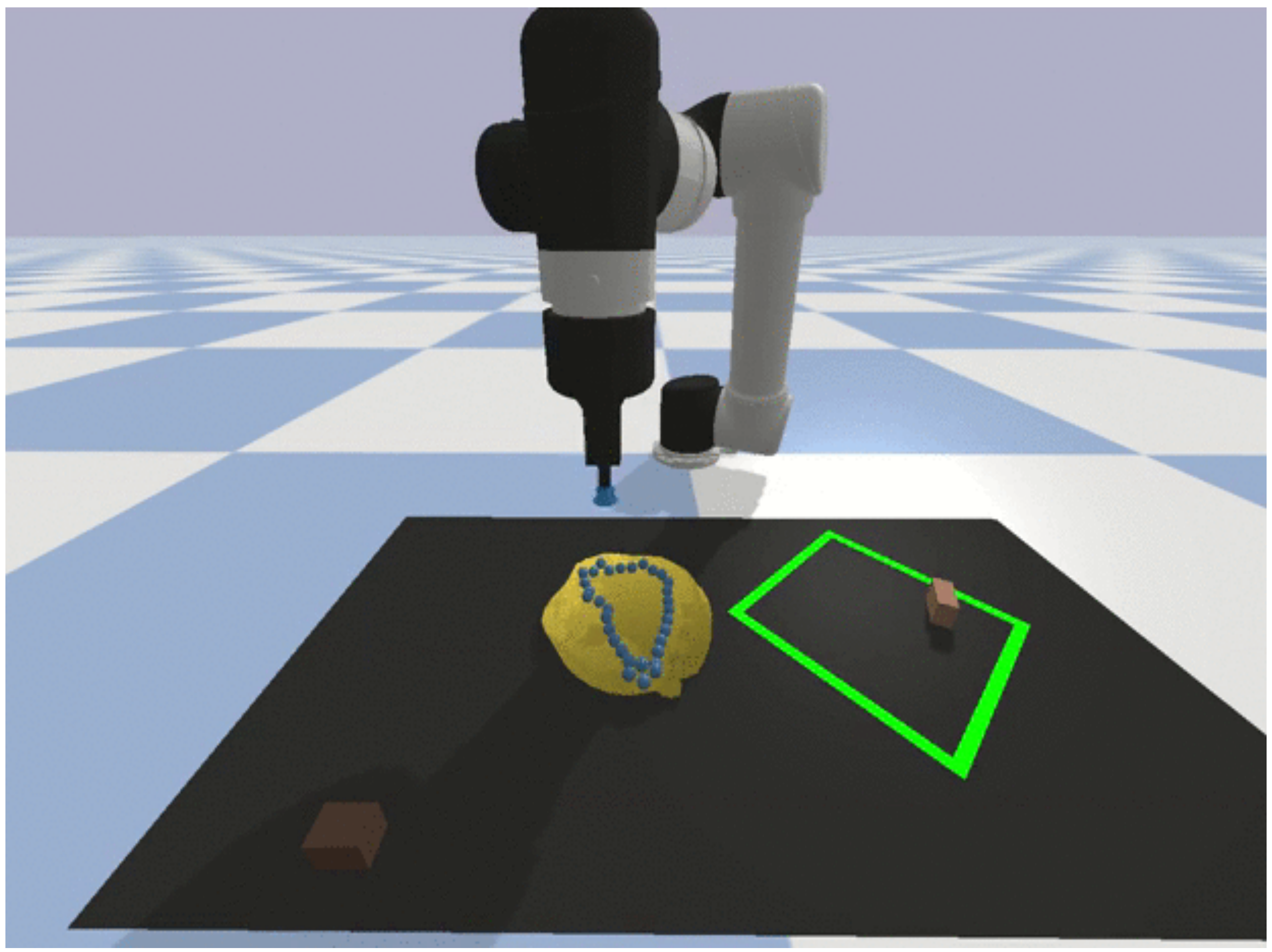
§evaluated based on maximizing the convex hull area of a ring.
*evaluated by the percentage of a cable within a target zone.
†evaluated using fabric coverage, as in Seita et al. [53], [54].

- Also: script a demonstrator for each task, but for some of the harder ones, filter out demonstrations by whether they succeeded or not.
- So we only provide “successful” demos (though they are also slightly stochastic).

Learned Policy: Transporter Network



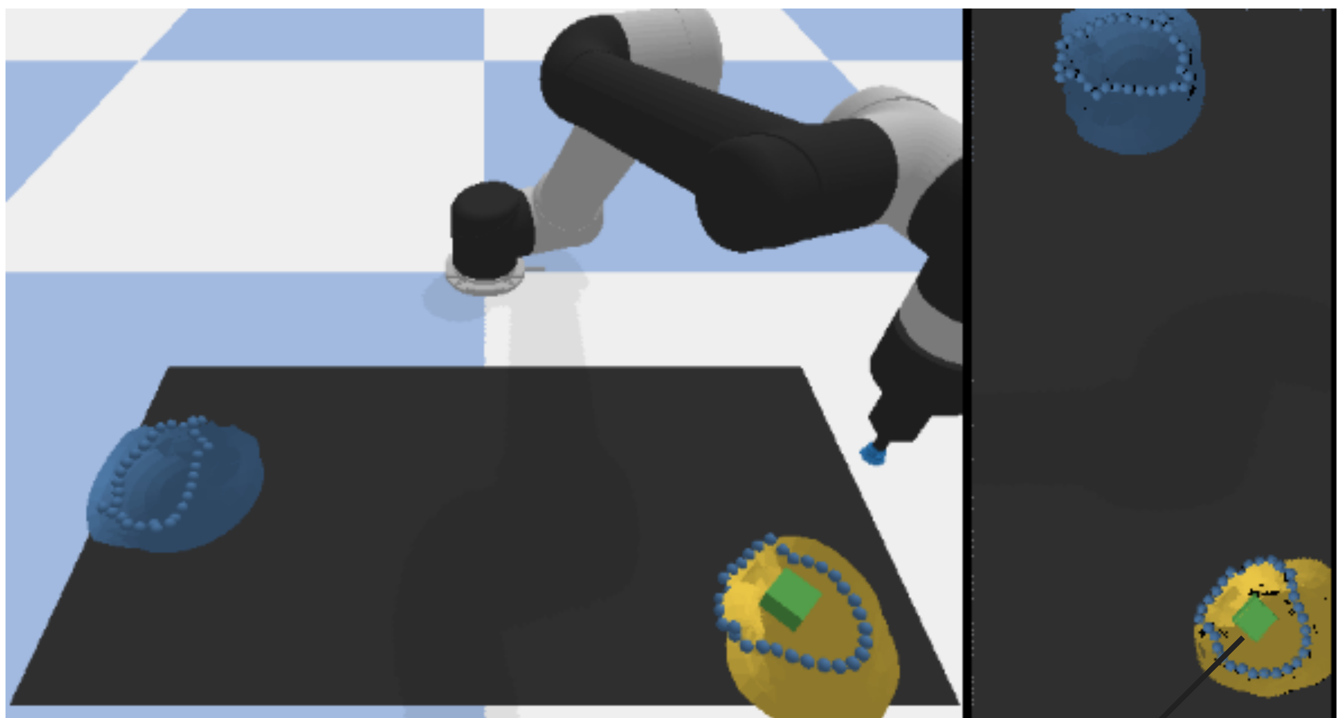
One item in bag
(zoomed-in for clarity)



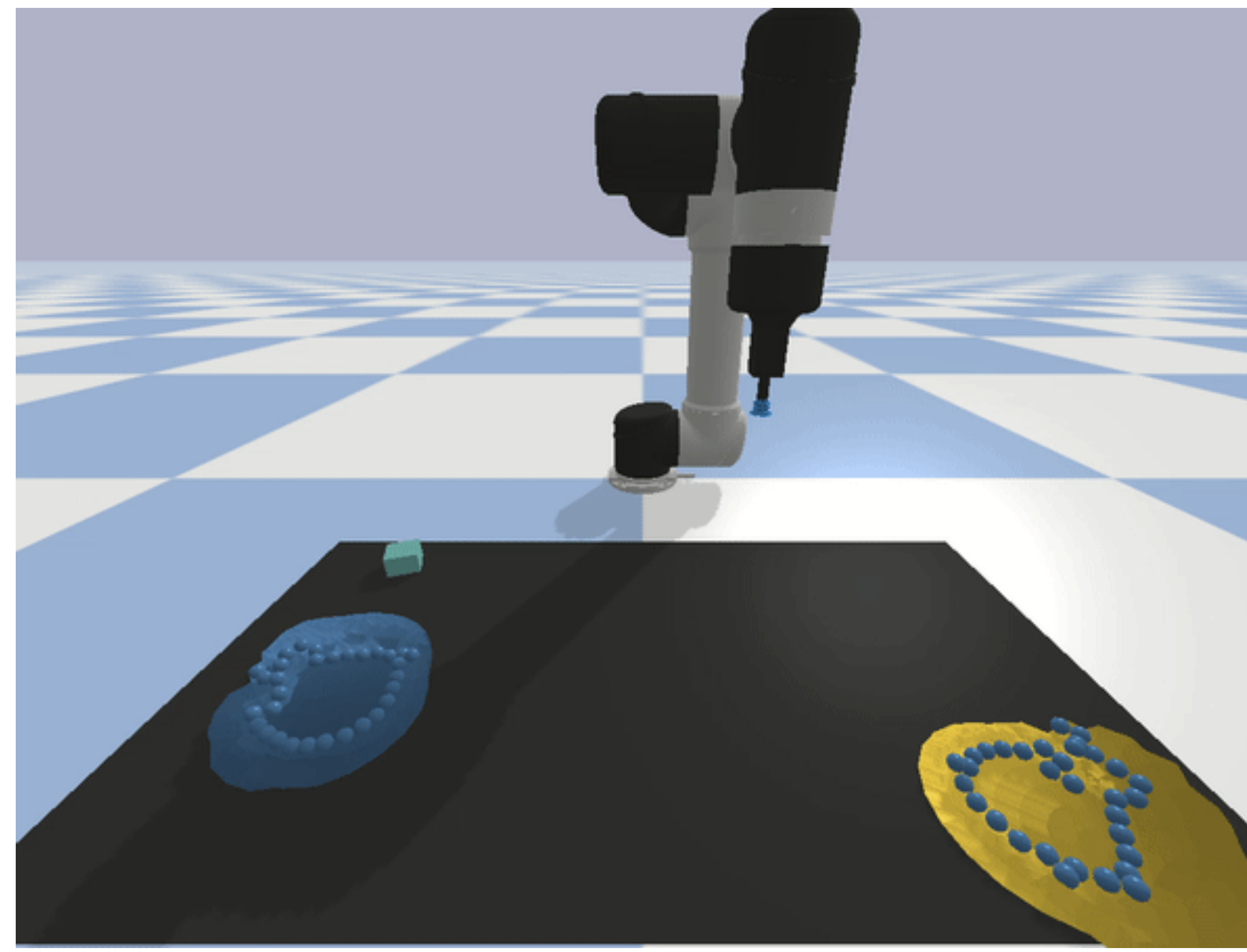
Two items in bag
(skipping some in-between action pauses)

Learned Policy: Transporter with Goal Conditioning

Goal Configuration



Block is here



Quantitative Results — Task Success Rates

	cable-ring				cable-ring-notarget				cable-shape				fabric-cover			
Method	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
GT-State MLP	0.0	0.0	0.0	0.0	0.0	1.7	3.3	5.0	0.4	0.8	1.0	0.5	3.3	25.0	18.3	21.7
GT-State MLP 2-Step	0.0	1.7	1.7	0.0	1.7	0.0	0.0	1.7	0.7	0.6	0.9	0.5	3.3	16.7	6.7	3.3
Transporter	16.7	50.0	55.0	68.3	15.0	68.3	73.3	70.0	75.6	80.6	90.1	86.5	85.0	100.0	100.0	100.0

	fabric-flat				bag-alone-open				bag-items-1				bag-items-2			
Method	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
GT-State MLP	26.0	45.6	65.6	71.3	15.0	16.7	35.0	43.3	1.7	20.0	30.0	31.7	0.0	0.0	6.7	8.3
GT-State MLP 2-Step	21.8	30.9	45.5	41.7	11.7	15.0	18.3	26.7	0.0	8.3	28.3	31.7	0.0	1.7	6.7	11.7
Transporter	42.1	86.5	89.5	88.8	18.3	50.0	61.7	63.3	25.0	36.7	48.3	51.7	5.0	30.0	41.7	46.7

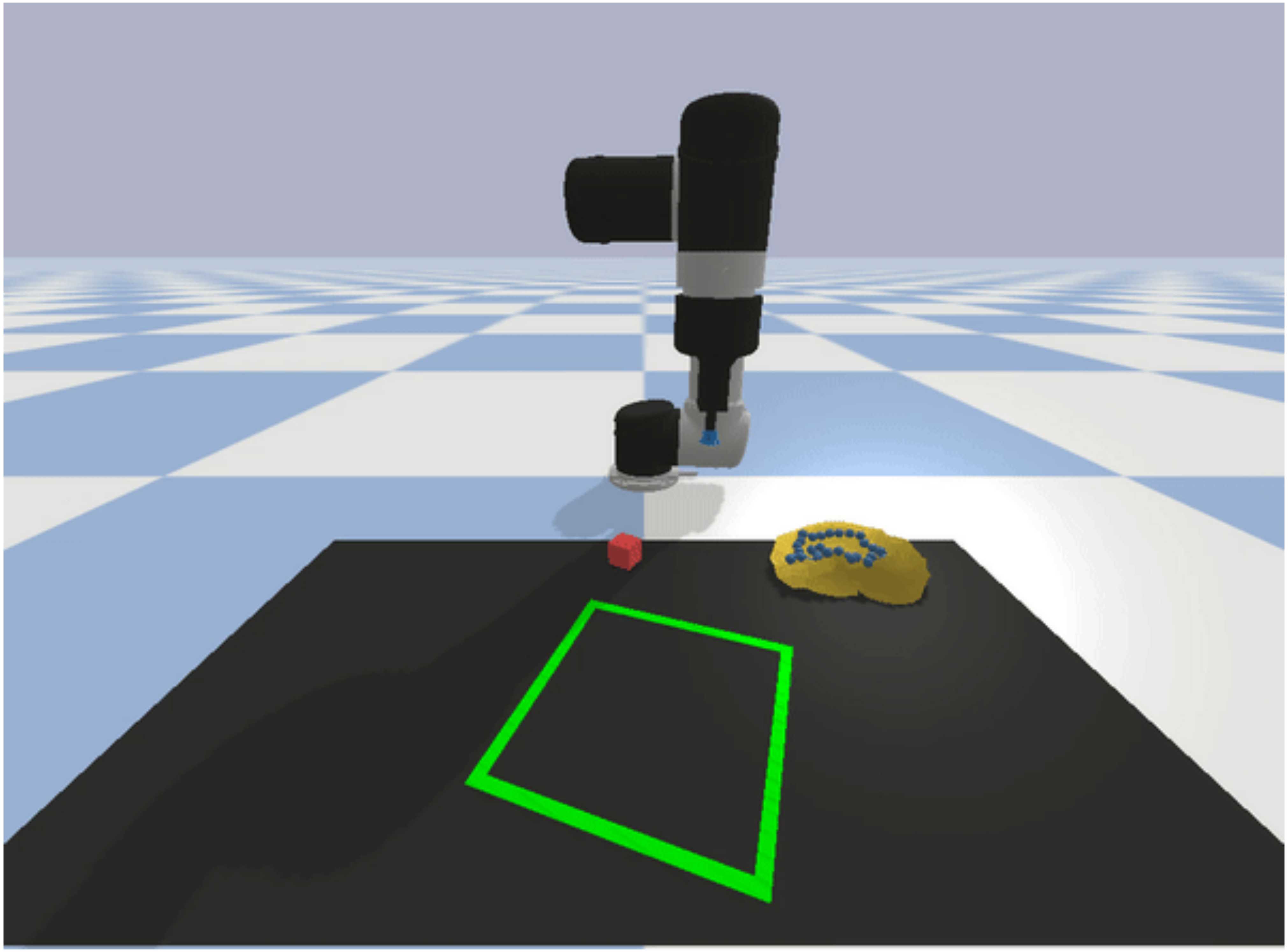
Tasks without Goal Conditioning

	cable-line-notarget				cable-shape-notarget				fabric-flat-notarget				bag-color-goal			
Method	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
GT-State MLP	11.1	44.5	72.7	77.4	11.1	42.7	66.0	65.4	14.8	49.8	62.1	63.2	0.8	0.8	10.0	14.9
GT-State MLP 2-Step	8.5	39.4	58.8	65.4	9.4	44.9	54.9	56.4	23.0	51.1	59.6	61.9	4.9	5.0	5.0	15.0
Transporter-Goal-Stack	63.5	82.8	53.0	54.4	54.4	47.5	45.1	45.6	16.5	26.3	25.3	20.1	12.4	21.6	65.4*	70.3*
Transporter-Goal-Split	74.9	95.6	53.9	99.2	48.4	75.1	64.9	76.4	27.6	35.6	30.1	77.0	10.0	63.1	40.1*	49.8*

Tasks with Goal Conditioning

*trained with 40K iterations.

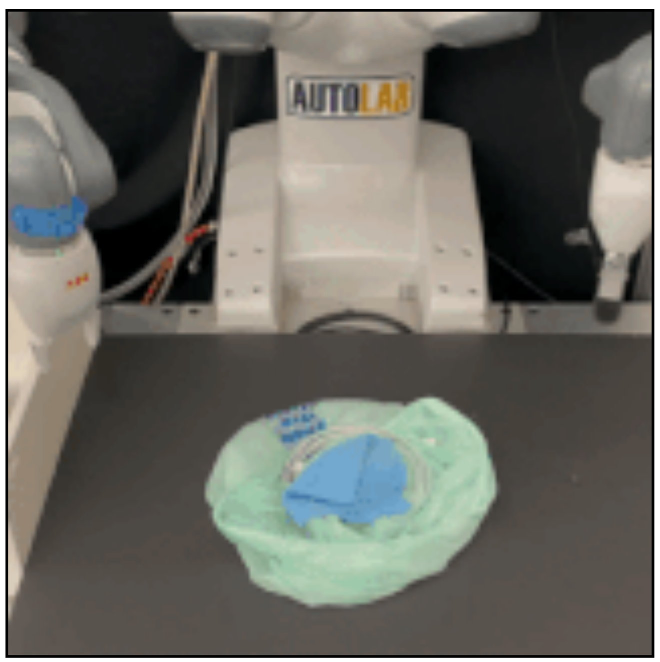
Current Limitation



The coarse pick-and-place policy cannot react in real time.

Conclusions and Future Work

- An open-source benchmark for 1D, 2D, and 3D deformable manipulation.
- An image goal-conditioned extension of Transporter Networks.
- Learned pick-and-place policies to manipulate deformables.
- Future work:
  - Extend to physical bagging (some progress here).
  - Go beyond pick-and-place actions, e.g., “stuff and kick”?



Remember when we used to travel (regularly)? We used suitcases.

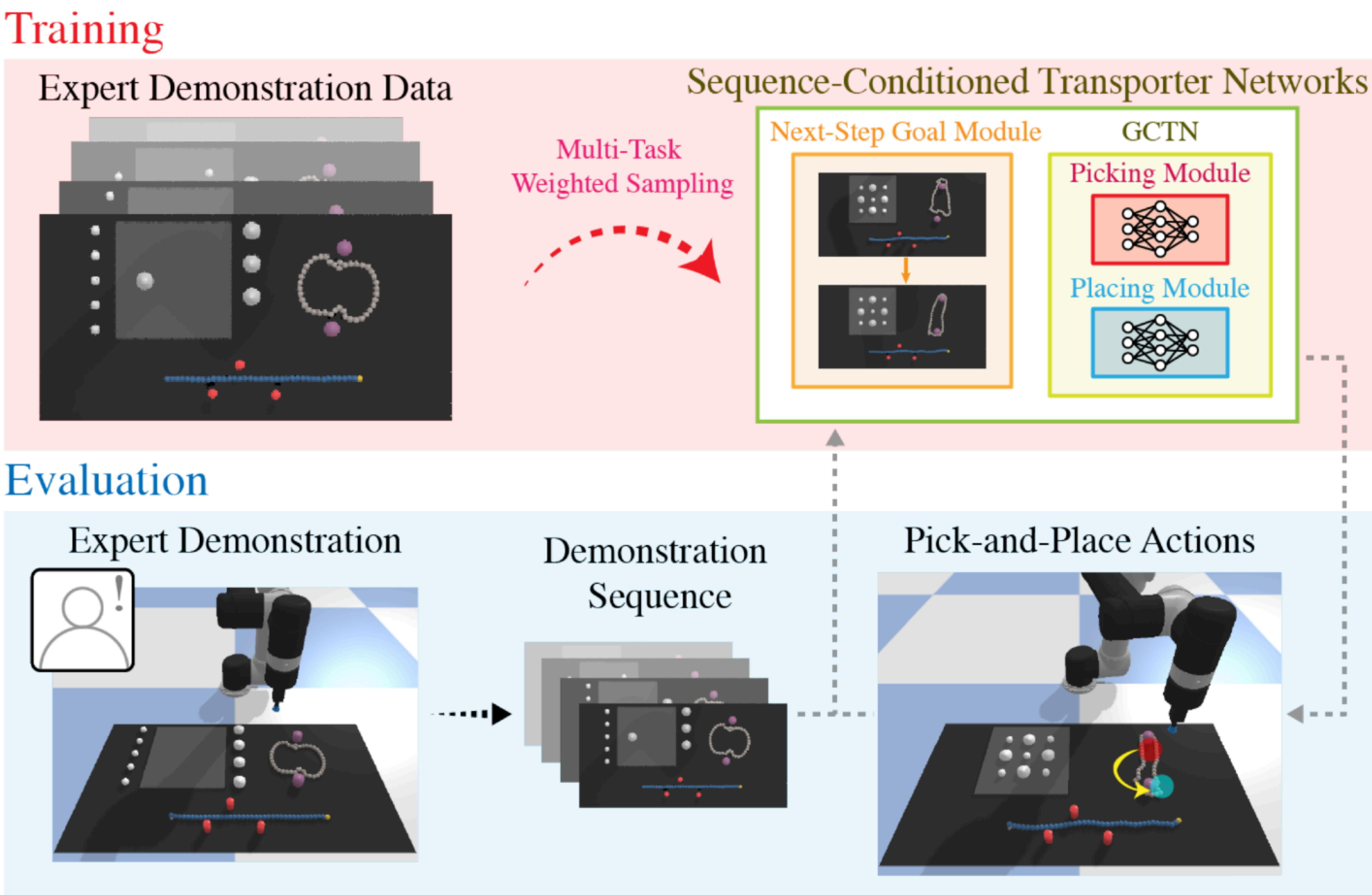
Seita et al., Initial Results on Grasping and Lifting Physical Deformable Bags, IROS workshop 2021.

Extension of Transporter Networks: CLIPort

- Combines OpenAI's CLIP with Transporter Networks
- Can do language-conditioned tasks.

Shridhar et al., *CLIPort: What and Where Pathways for Robotic Manipulation*, CoRL 2021.
Radford*, Kim* et al., CLIP: Connecting Text and Images. https://openai.com/blog/clip/

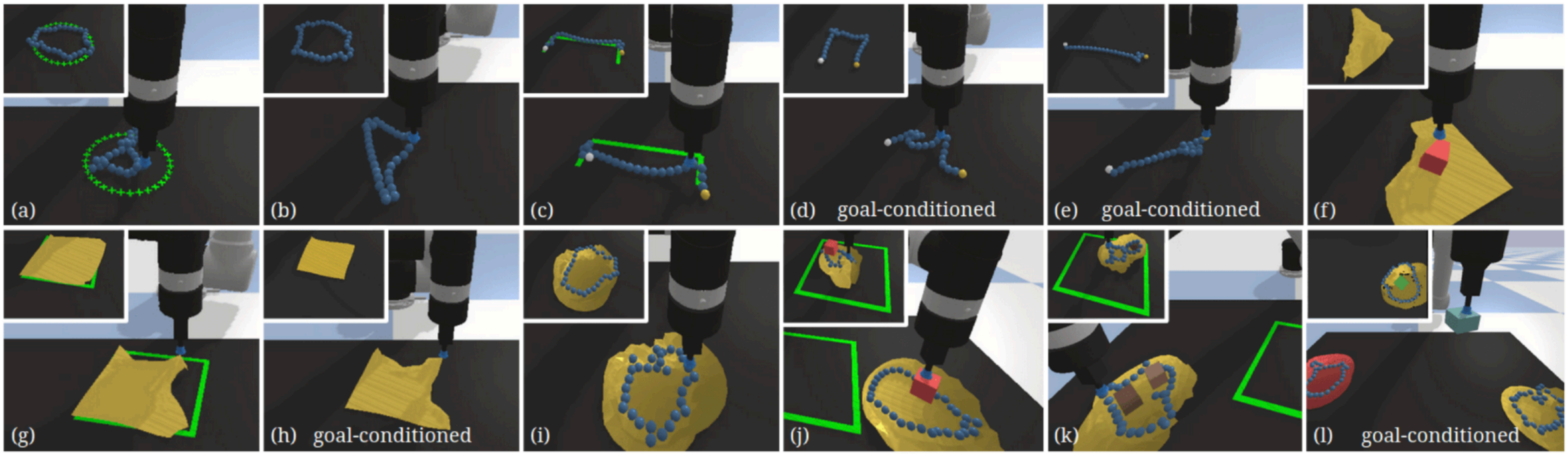
Extension of Transporter Networks: SCTN



- Sequence-Conditioned Transporter Networks to extend GCTN.
- Proposes *MultiRavens* for compositional tasks.

Lim et al., *Multi-Task Learning with Sequence-Conditioned Transporter Networks*, CoRL 2021.

Lots of Benchmarks to Try!



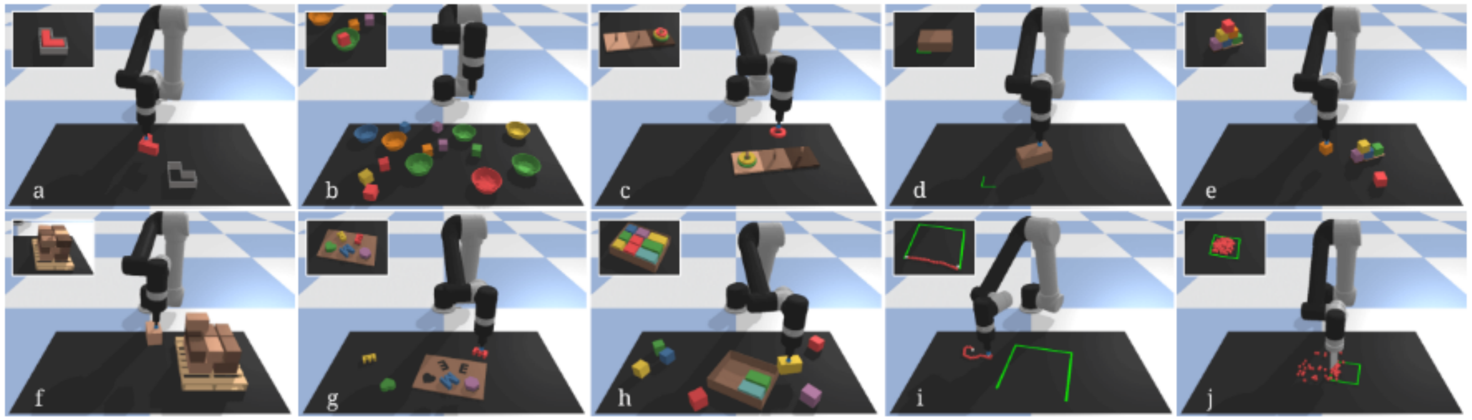
- Ravens
- DeformableRavens (shown above)
- CLIPort Tasks ("ClipRavens"?)
- MultiRavens

Benchmarks have been important for the development of computer vision and NLP, and it's encouraging to see more benchmarks for robot manipulation.

Open-Source!

Ravens

Ravens is a collection of simulated tasks in PyBullet for learning vision-based robotic manipulation, with emphasis on pick and place. It features a Gym-like API with 10 tabletop rearrangement tasks, each with (i) a scripted oracle that provides expert demonstrations (for imitation learning), and (ii) reward functions that provide partial credit (for reinforcement learning).



(a) **block-insertion**: pick up the L-shaped red block and place it into the L-shaped fixture.
(b) **place-red-in-green**: pick up the red blocks and place them into the green bowls amidst other objects.
(c) **towers-of-hanoi**: sequentially move disks from one tower to another—only smaller disks can be on top of larger ones.
(d) **align-box-corner**: pick up the randomly sized box and align one of its corners to the L-shaped marker on the tabletop.
(e) **stack-block-pyramid**: sequentially stack 6 blocks into a pyramid of 3-2-1 with rainbow colored ordering.
(f) **palletizing-boxes**: pick up homogeneous fixed-sized boxes and stack them in transposed layers on the pallet.
(g) **assembling-kits**: pick up different objects and arrange them on a board marked with corresponding silhouettes.
(h) **packing-boxes**: pick up randomly sized boxes and place them tightly into a container.
(i) **manipulating-rope**: rearrange a deformable rope such that it connects the two endpoints of a 3-sided square.
(j) **sweeping-piles**: push piles of small objects into a target goal zone marked on the tabletop.

Some tasks require generalizing to unseen objects (d,g,h), or multi-step sequencing with closed-loop feedback (c,e,f,h,i,j).

Team: this repository is developed and maintained by Andy Zeng, Pete Florence, Daniel Seita, Jonathan Tompson, and (your name here)... This is the reference repository for the paper:

Transporter Networks: Rearranging the Visual World for Robotic Manipulation

Summary and Takeaways

- Introduced Transporter Network and extensions.
- Formulate robot manipulation as a sequence of rigid displacements by rearranging pixels (3D space).
- Use an <u>action-centric</u> (not object-centric) approach.
- Use orthographic images and equivariance for data augmentation.
- Highly sample-efficient robot manipulation from a few visual (image) demonstrations.
- Use implicit models with fast inference from forward pass.