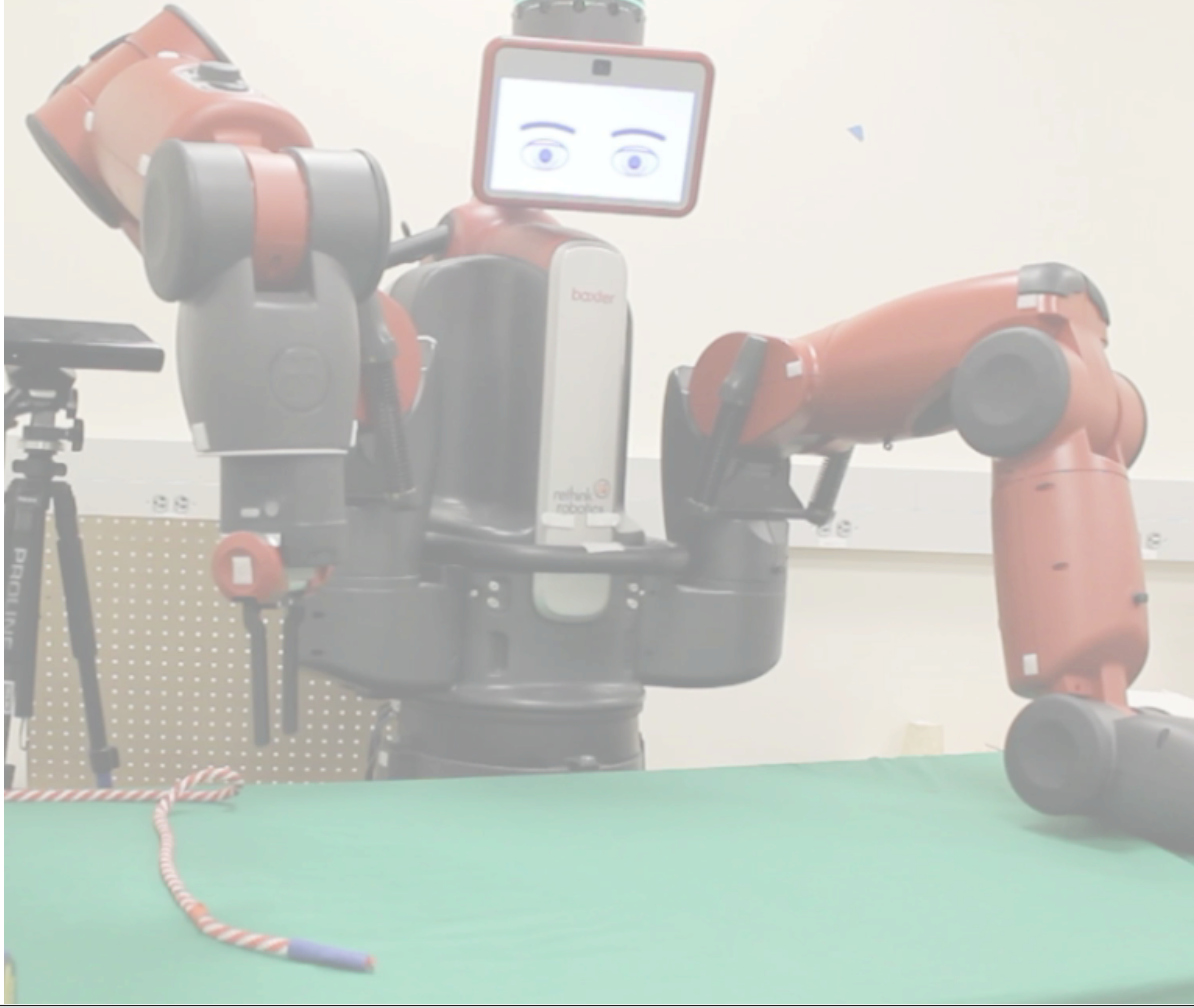
Lecture 16

Vision for Embodied Agents

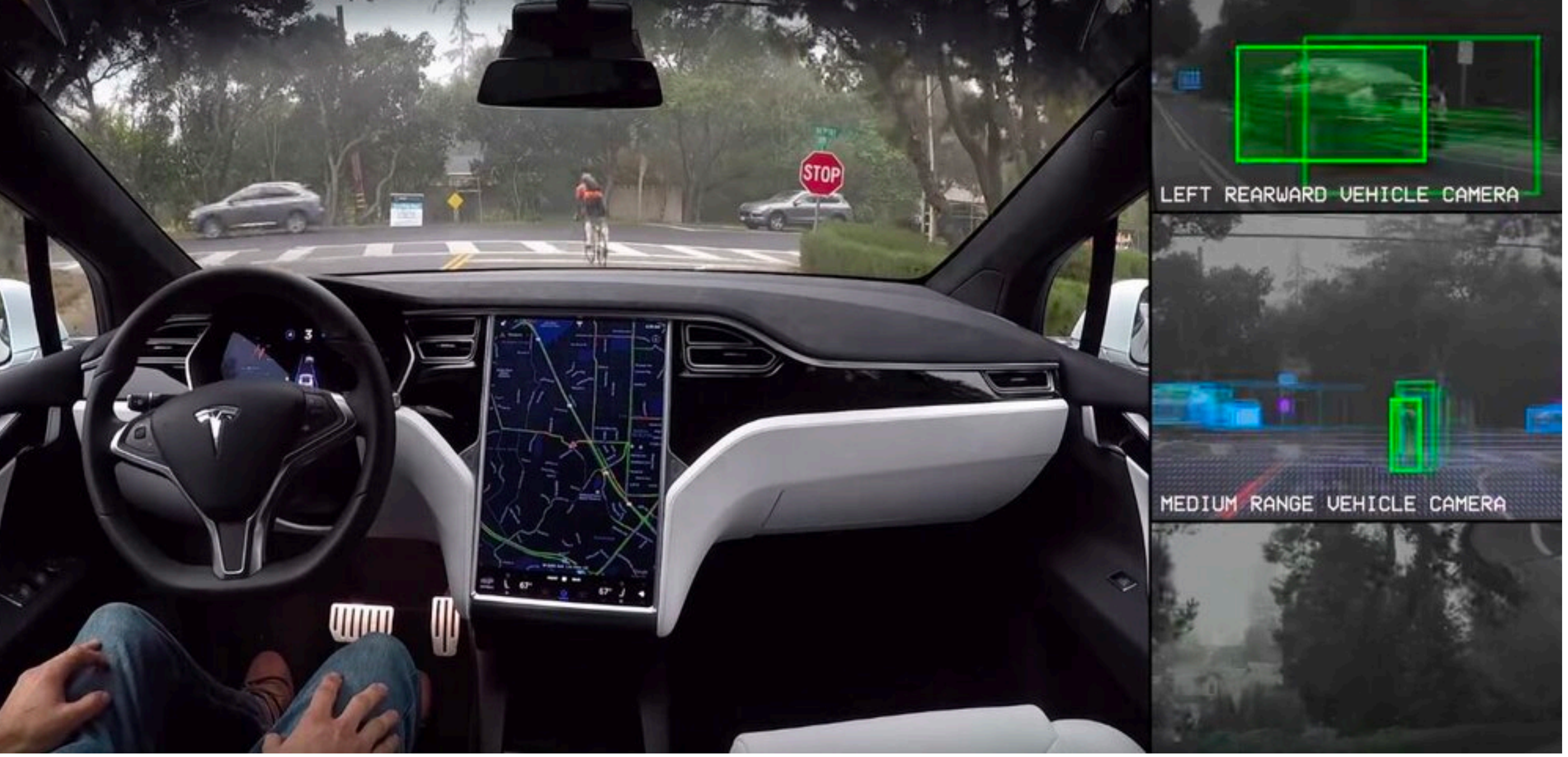


Spring 2021

Bill Freeman, Phillip Isola

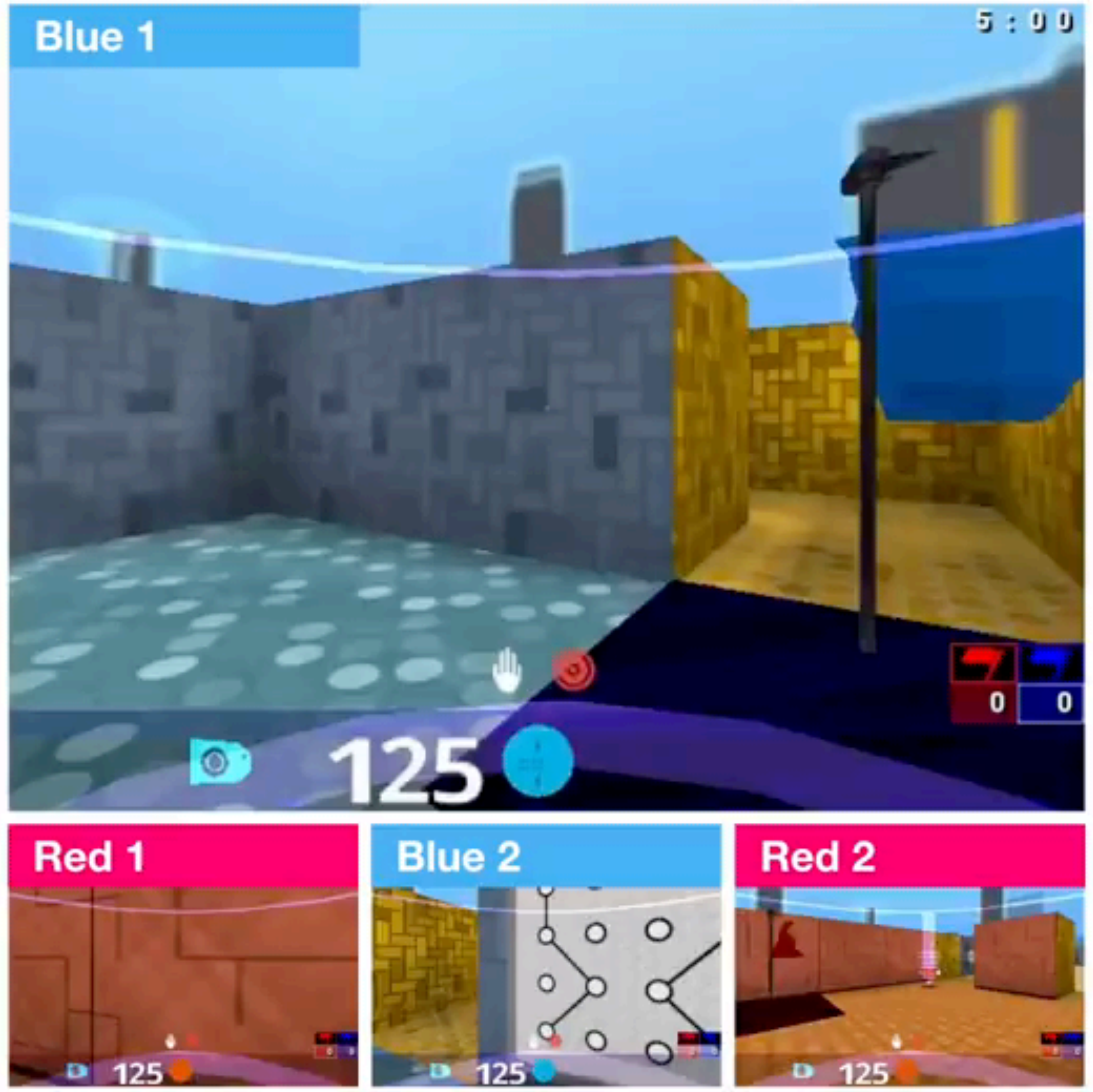
16. Vision for Embodied Agents

- Formalisms for intelligent agents *(environment, state, action, policy)*
- Imitation learning
- Reinforcement learning
  - Policy gradient method
- Object representations for interaction
  - 3D meshes
  - Dense descriptors

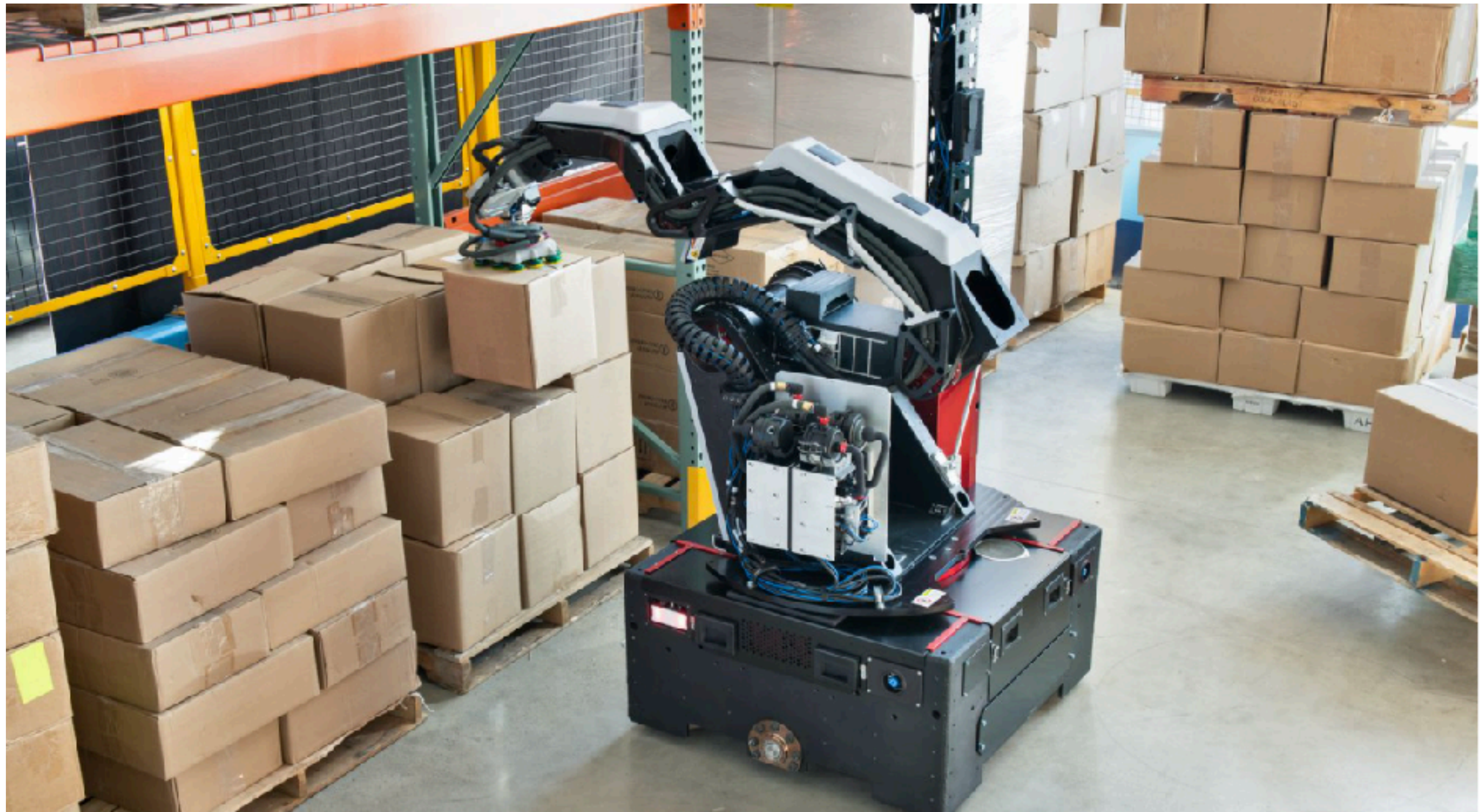


Tesla Autopilot

Agent observation raw pixels



[Jaderberg et al., Science 2019]

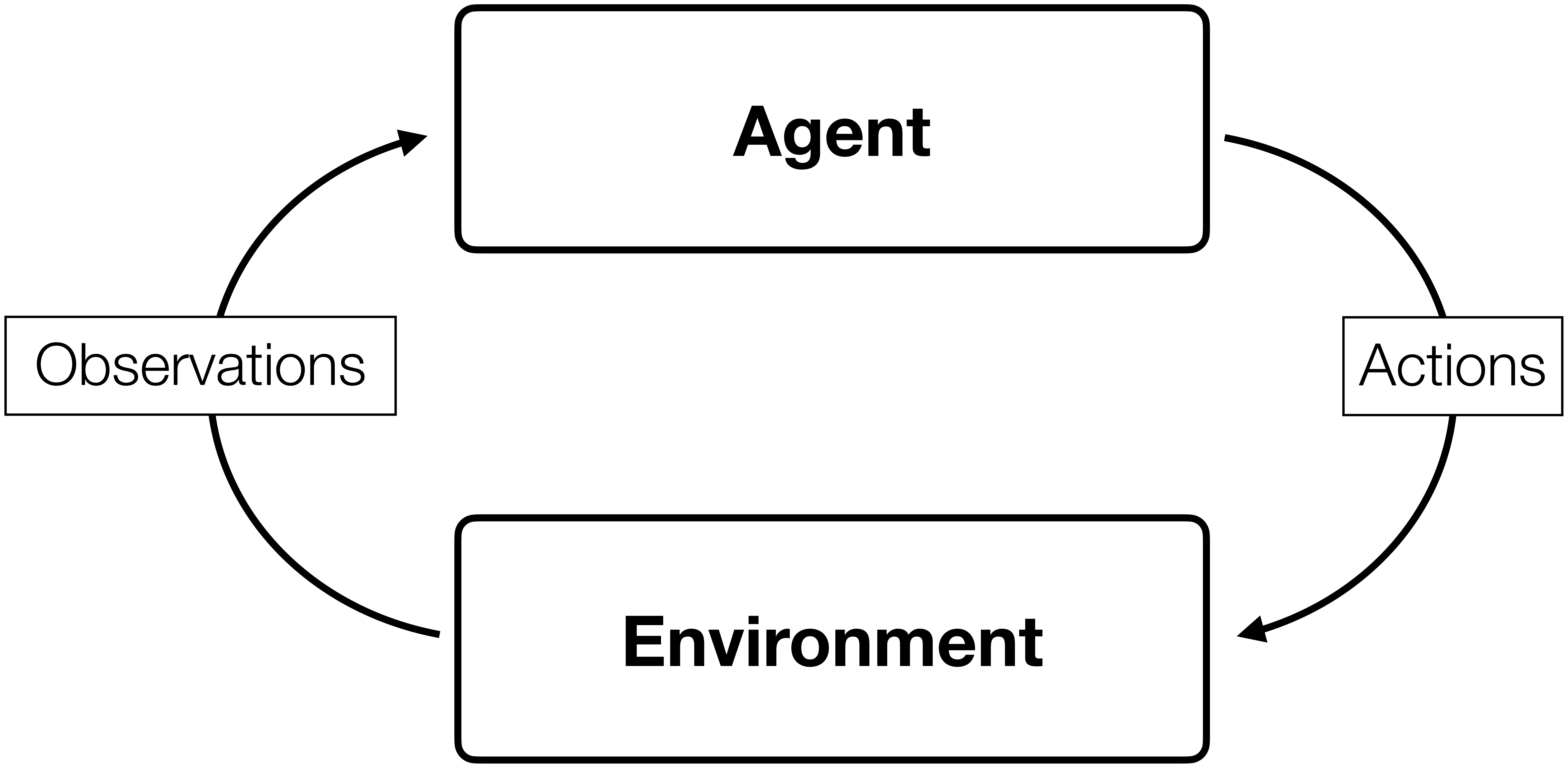


Boston Dynamics “Stretch”

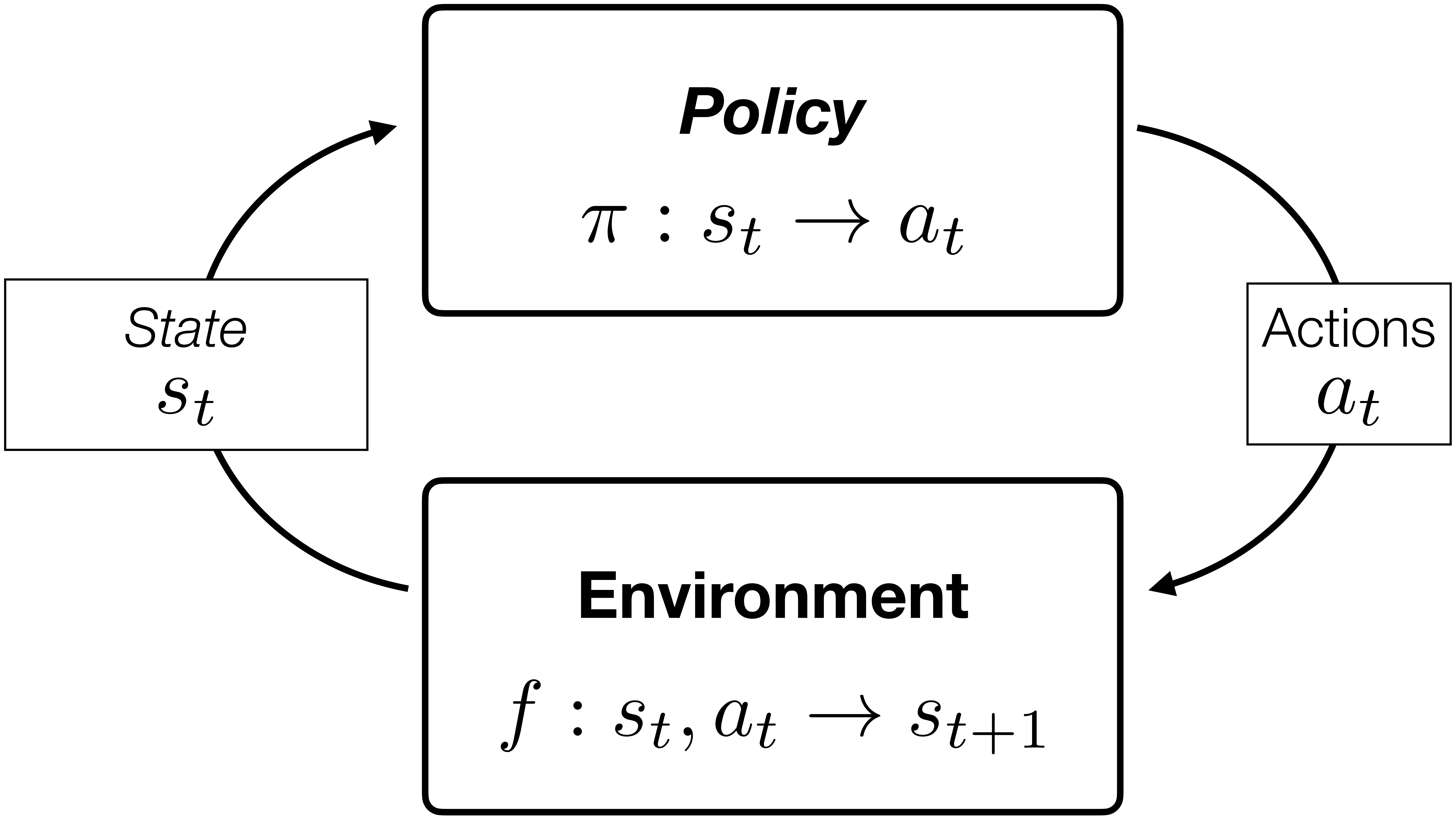
The whole purpose of visual perception, in humans, is to make good motor decisions.

We are **sensorimotor** systems.

Intelligent agents



Intelligent agents



Recipe for deep learning in a new domain

1. Transform your data into numbers (e.g., a vector)

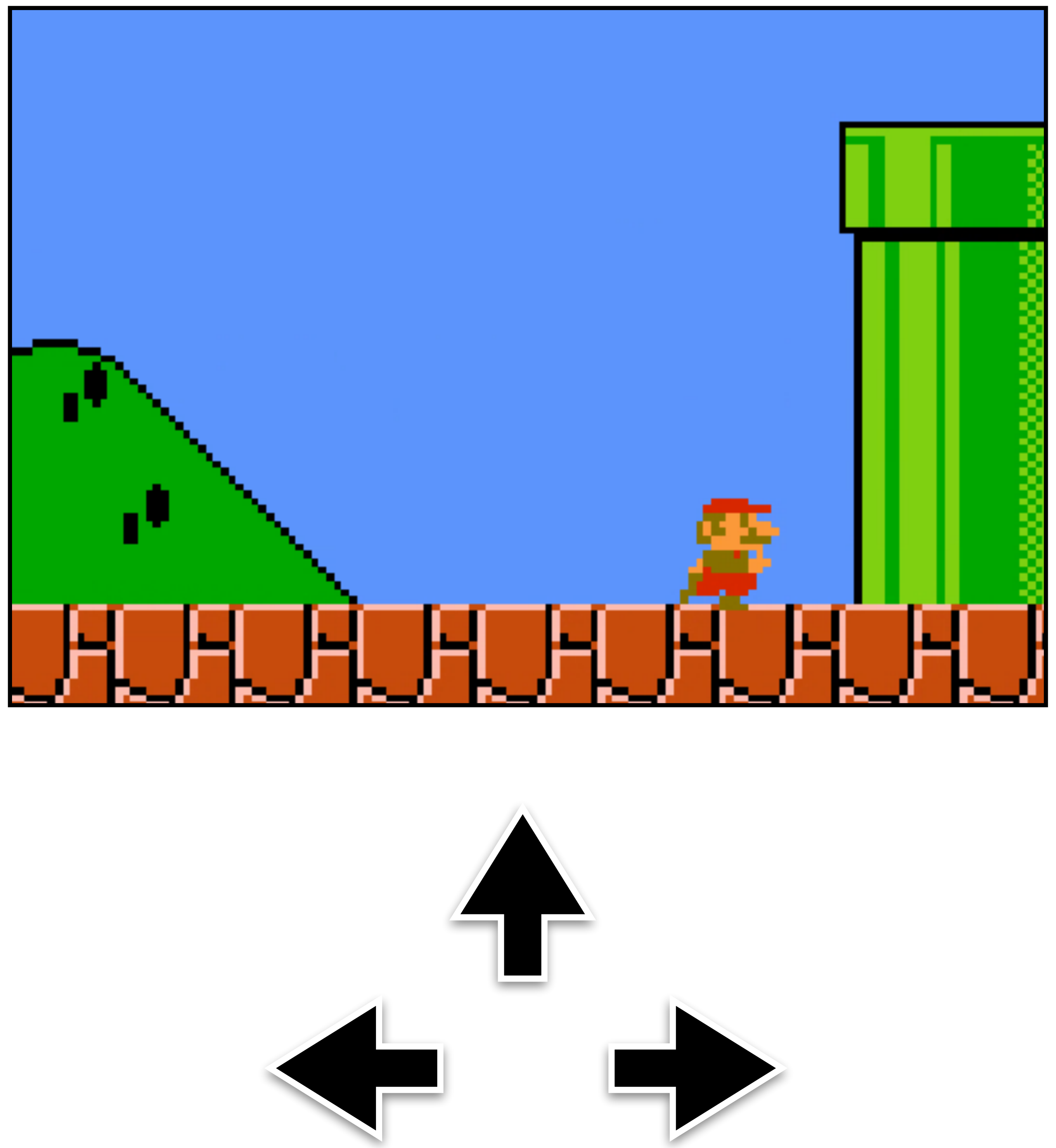
2. Transform your goal into an numerical measure (objective function)

3. #1 and #2 specify the “learning problem”

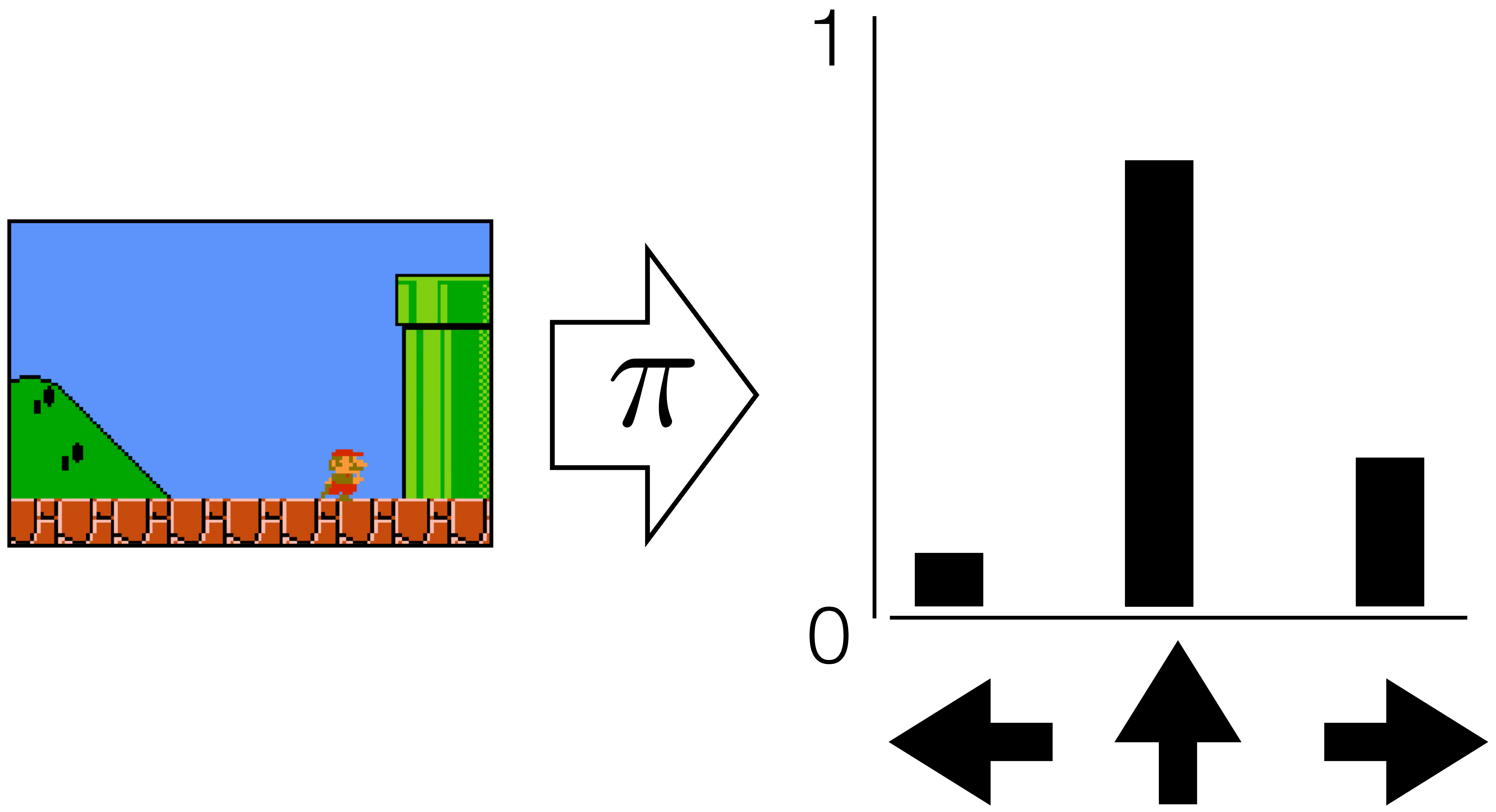
4. Use a generic optimizer (SGD) and an appropriate architecture (e.g., CNN or RNN) to solve the learning problem

How to represent a state? How to represent policy?

state: pixels!



policy: action classifier



Learning from examples

(aka **supervised learning**)

Training data

$$\{x^{(1)}, y^{(1)}\}$$
$$\{x^{(2)}, y^{(2)}\} \quad \rightarrow \quad \boxed{\text{Learner}} \quad \rightarrow \quad f : X \rightarrow Y$$
$$\{x^{(3)}, y^{(3)}\}$$
$$\dots$$

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}^{(i)}), \mathbf{y}^{(i)})$$

Imitation learning

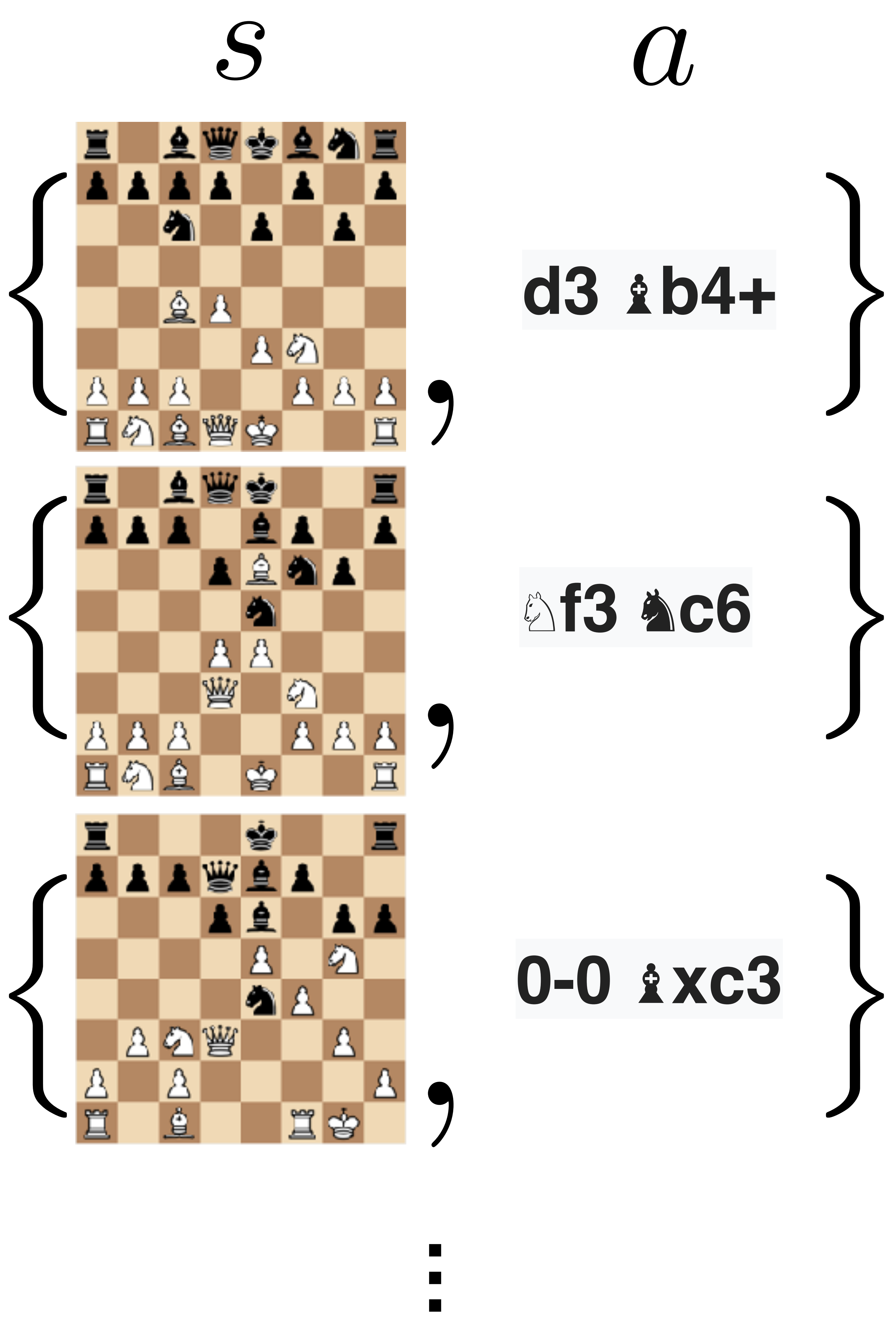
(still just **supervised learning**, applied to learn *policies*)

Training data

$$\{s_1, a_1\}$$
$$\{s_2, a_2\} \quad \rightarrow \quad \boxed{\text{Learner}} \quad \rightarrow \quad \pi : s \rightarrow a$$
$$\{s_3, a_3\}$$
$$\ldots$$

$$\pi^* = \arg\min_{\pi \in \Pi} \sum_{i=1}^{N} \mathcal{L}(\pi(s_i), a_i)$$

Imitation learning



$\rightarrow$

Learner

Objective

$$\pi(s) = \texttt{softmax}(g_\theta(s))$$

$$\mathcal{L}(a, \pi(s)) = H(a, \pi(s))$$

Hypothesis space

Convolutional neural net

Optimizer

Stochastic gradient descent

$\rightarrow \quad \pi$

Learning without examples

(includes **unsupervised learning** and **reinforcement learning**)

Data

$\{x^{(1)}\}$

$\{x^{(2)}\}$

$\{x^{(3)}\}$

$\dots$

$\rightarrow$ Learner $\rightarrow$?

Representation Learning

Data

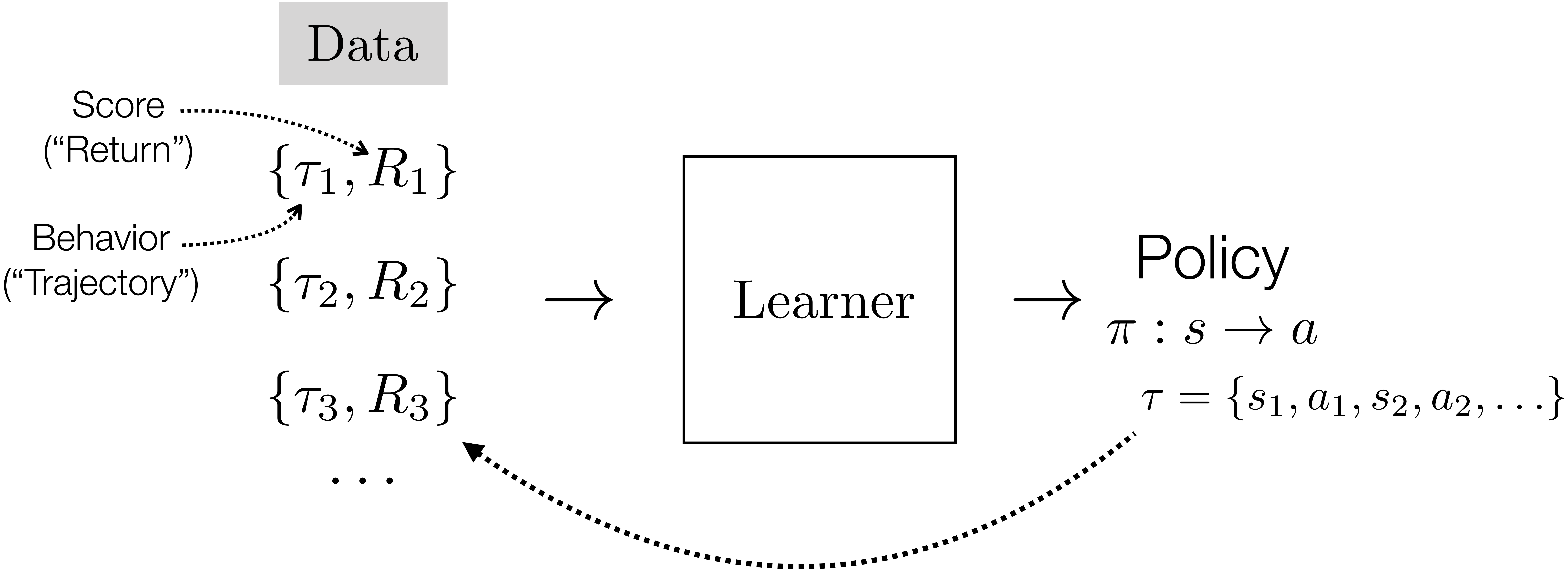
$\{x^{(1)}\}$

$\{x^{(2)}\}$ $\rightarrow$ Learner $\rightarrow$ Representations

$\{x^{(3)}\}$

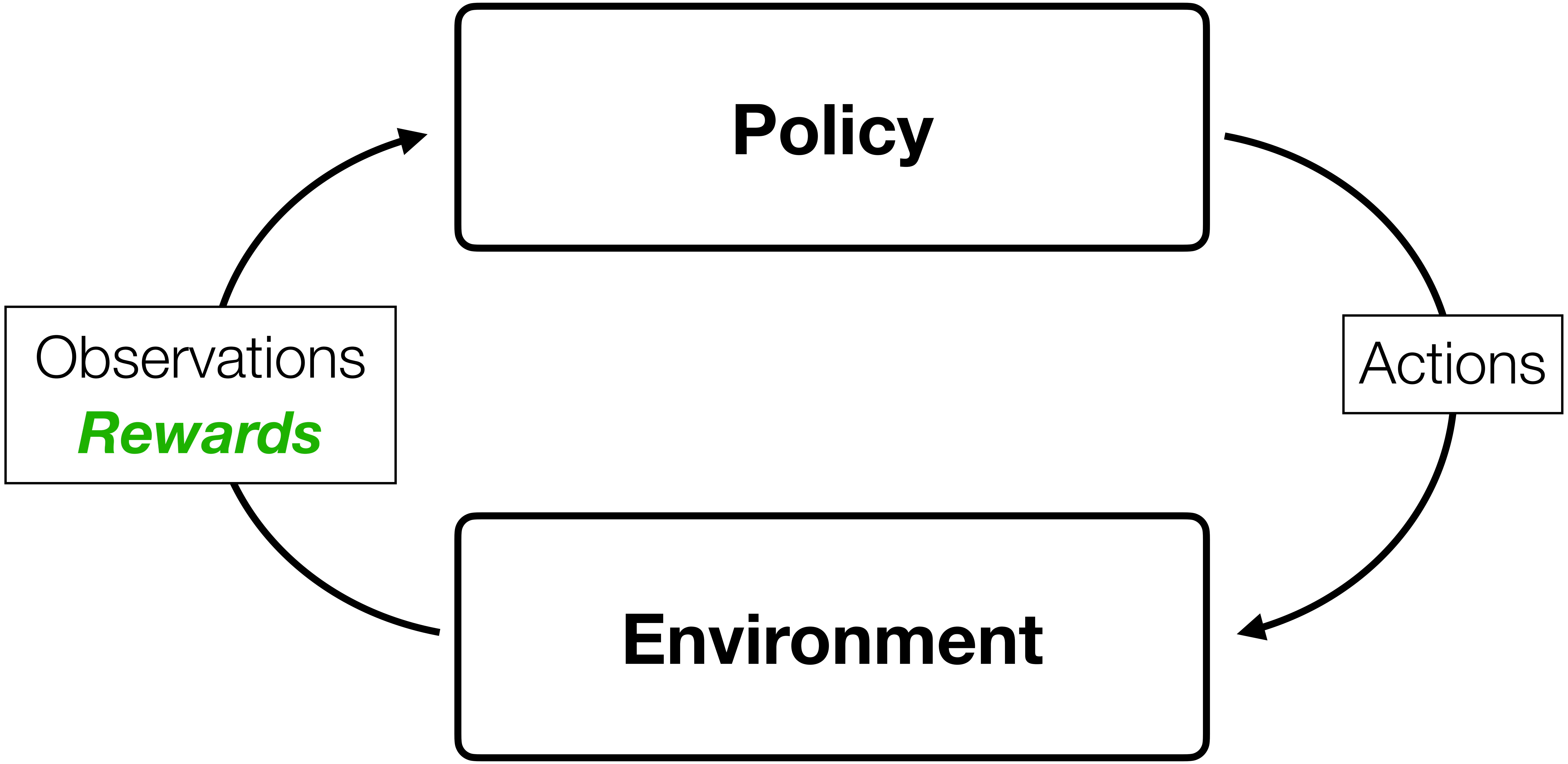
$\ldots$

Reinforcement learning



What's a good policy? (what's the learning objective?)

Reinforcement learning

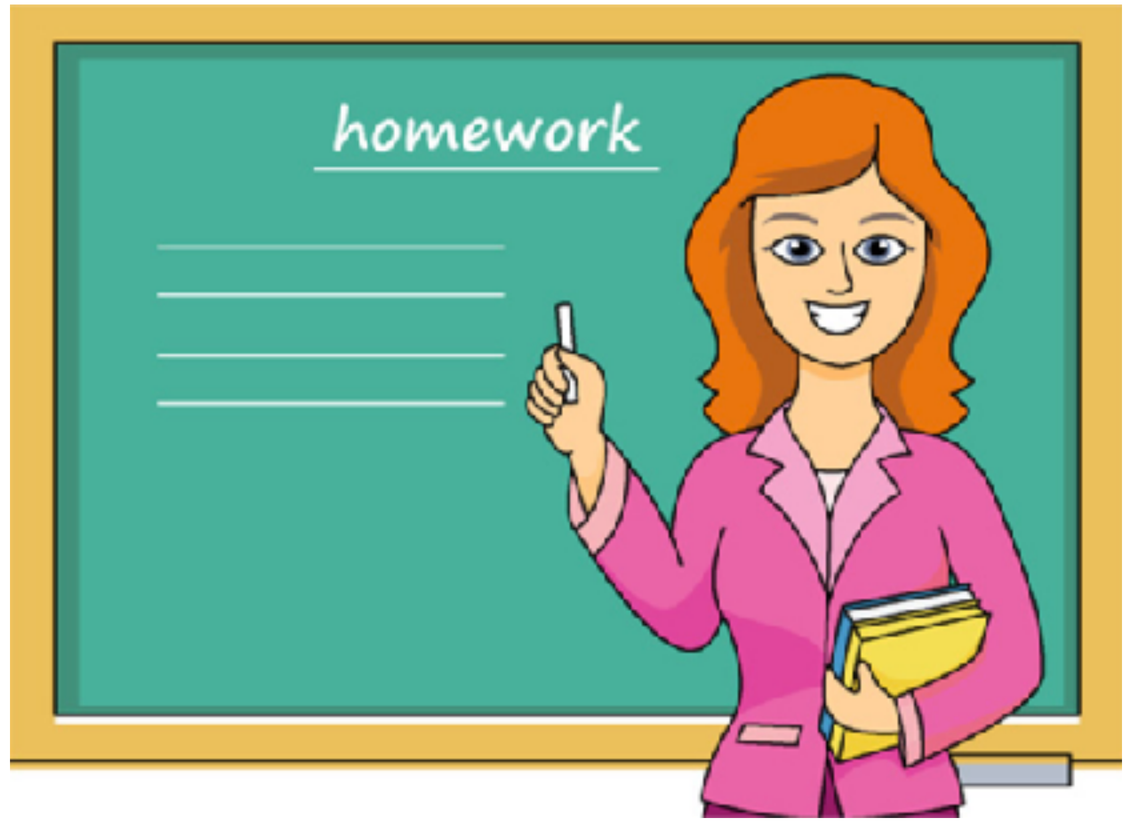


Learn a policy that takes actions that maximize **reward**

Imitation learning

Hand-curated training data

+ Instructive examples
+ Follows a curriculum
- Expensive
- Limited to teacher's knowledge

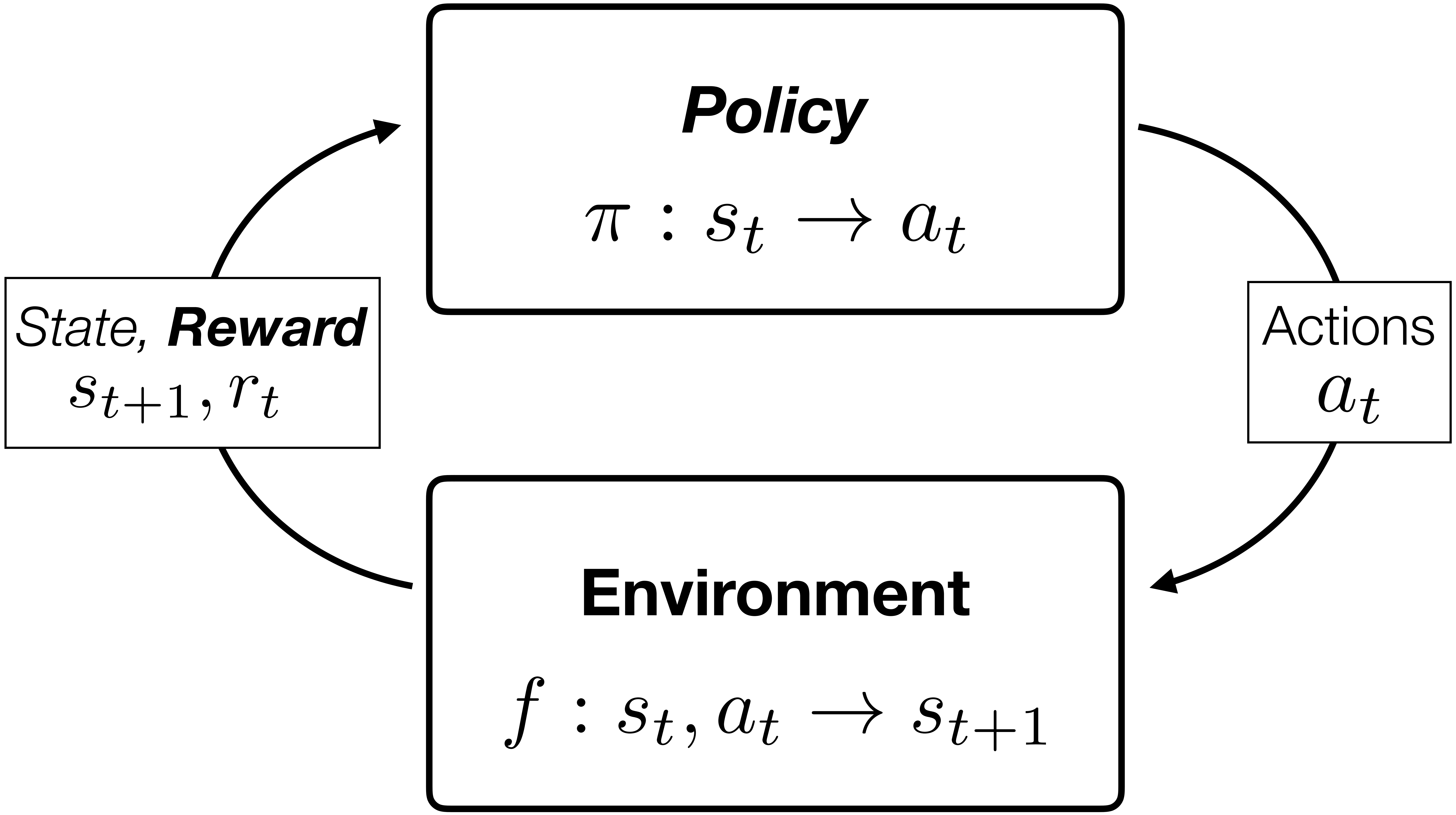


Reinforcement learning

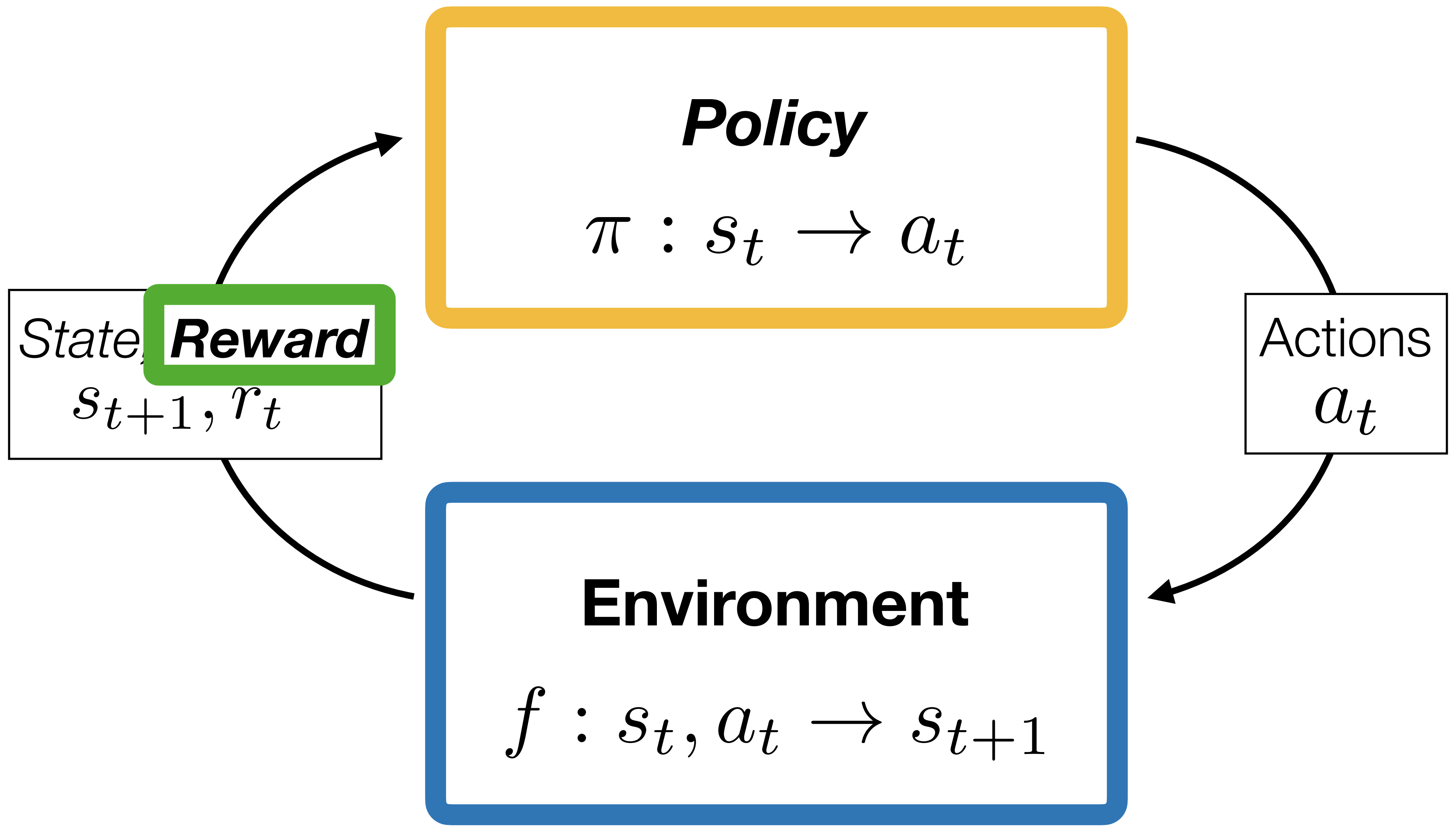
No training data, have to play around and collect the data *yourself*

+ No need for labeled data
+ Can learn things no human knows how to do
- Less instructive
- No curriculum
- Have to explore

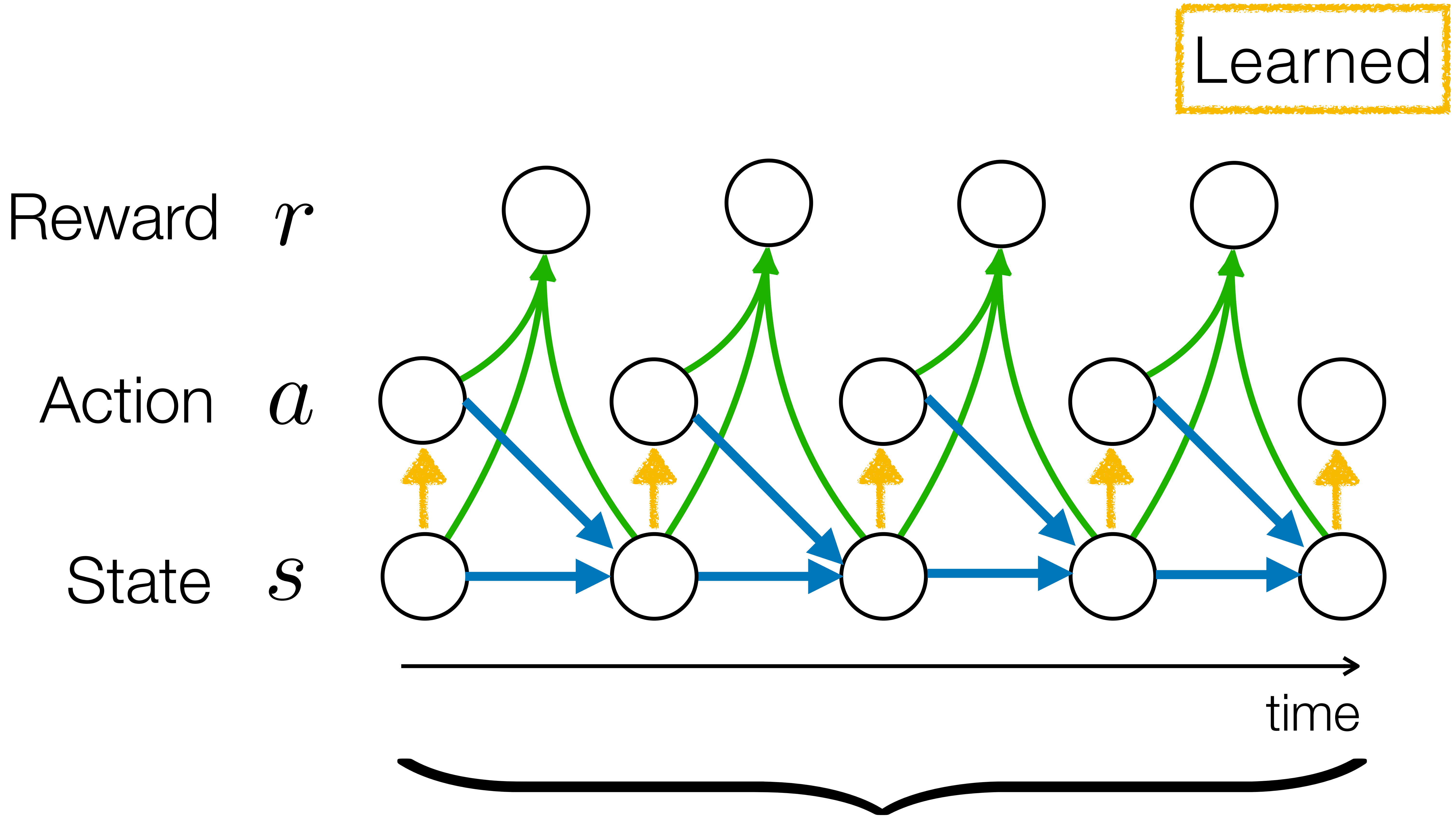
Reinforcement learning



Reinforcement learning

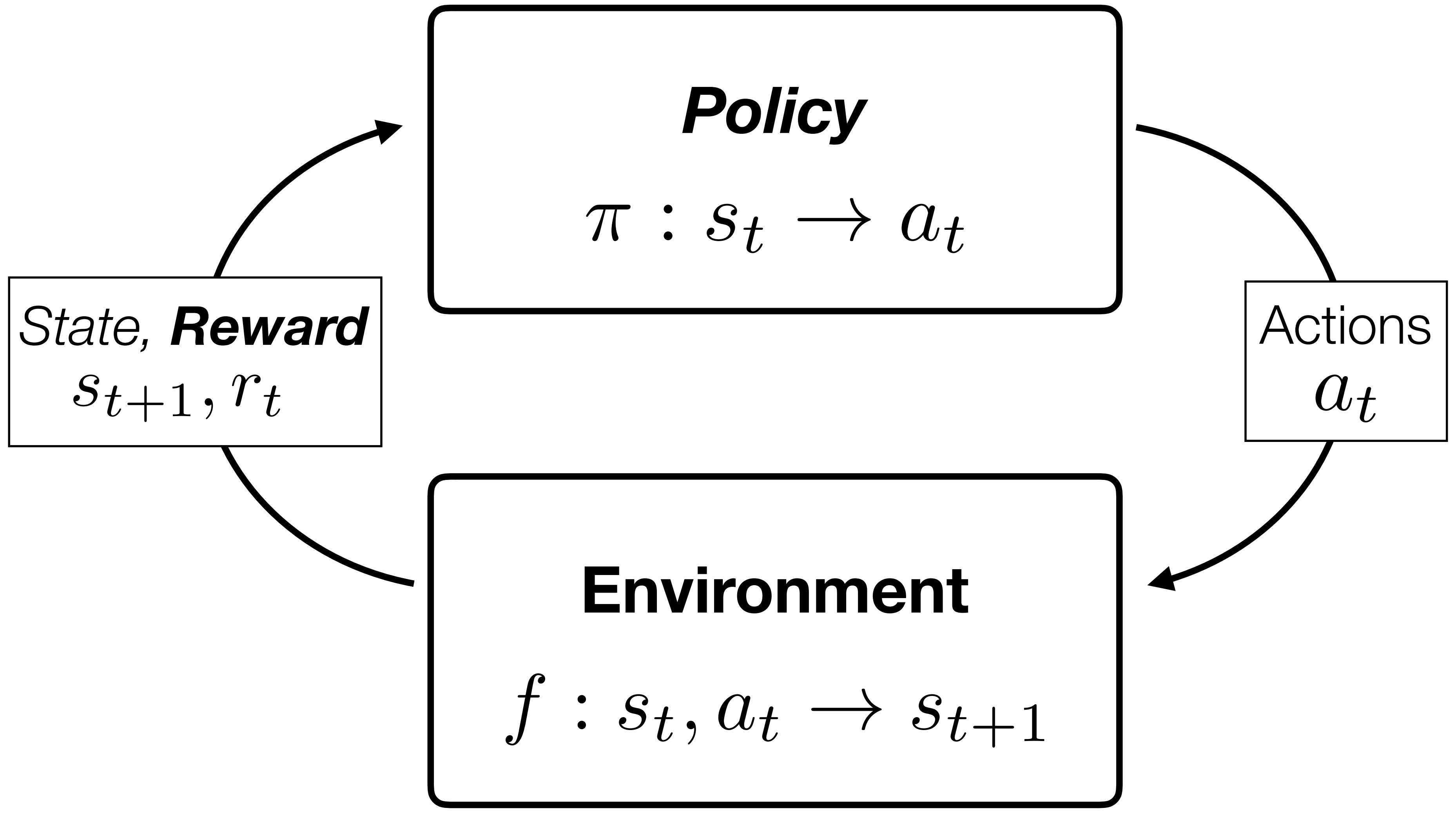


Markov decision process (MDP)



A sample from the MPD is called a **Trajectory** $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$

Reinforcement learning



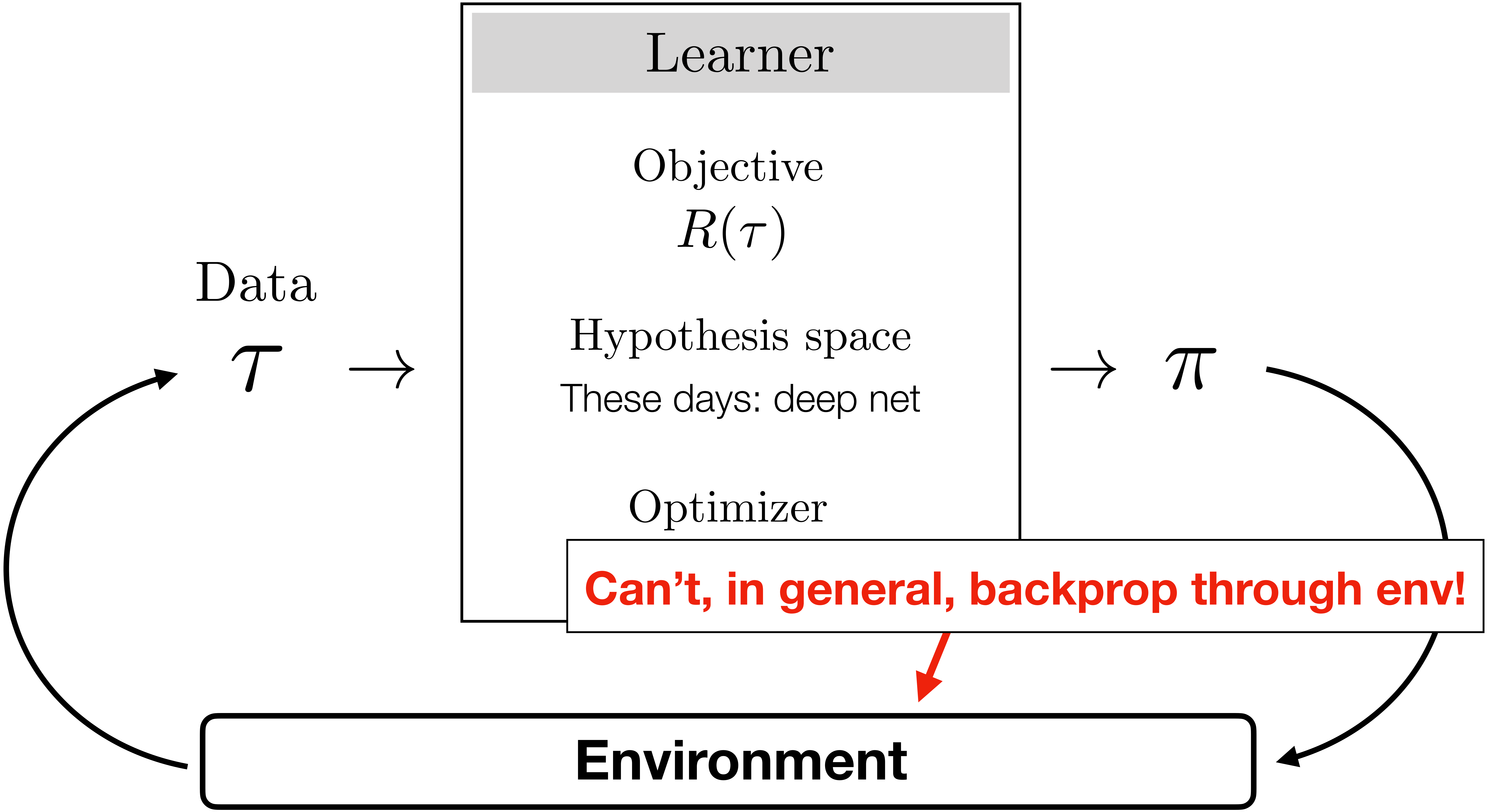
Trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$

Discounted Returns $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t, \quad \gamma \in (0, 1)$

Learn a policy that takes actions that maximize expected reward

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi}[R(\tau)]$$

Reinforcement learning

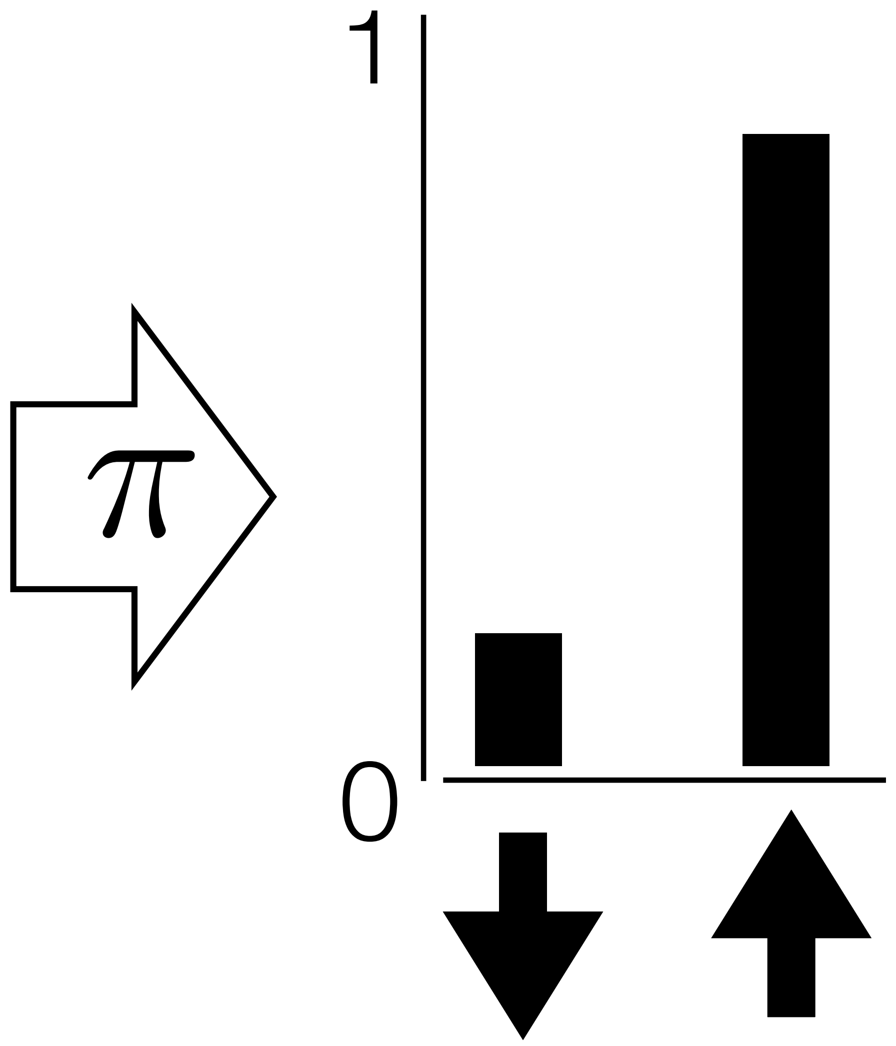


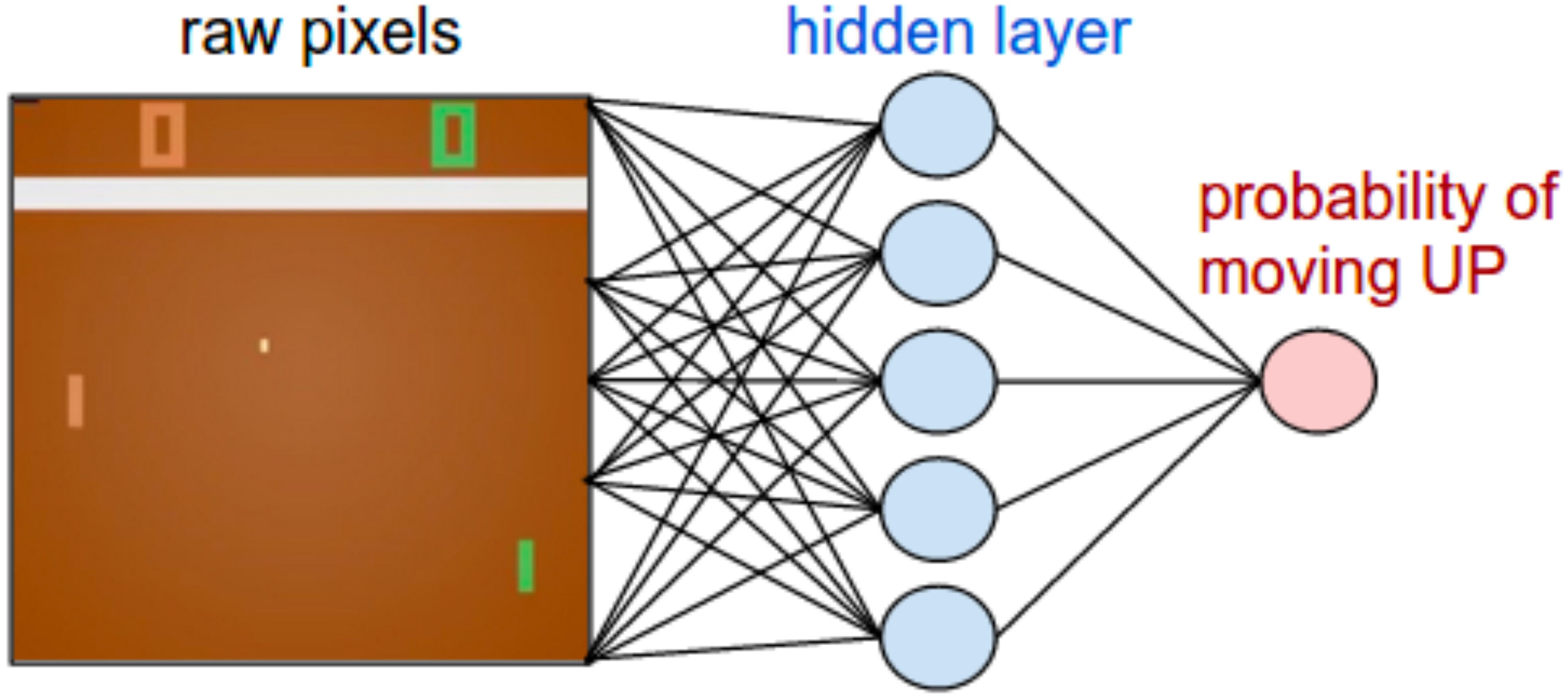
Environment is not differentiable! — How to optimize?

Idea #1 (trial and error):

Policy gradients: Run a policy for a while. See what actions led to high rewards. Increase their probability.

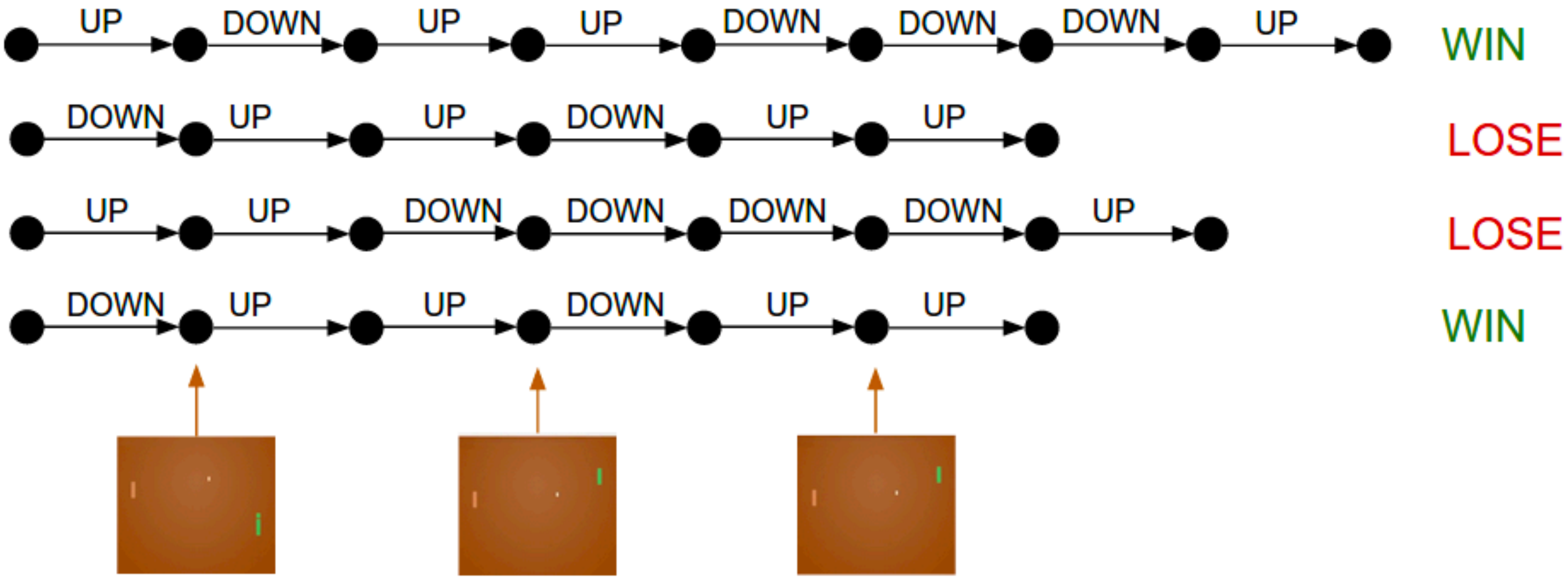
Pong





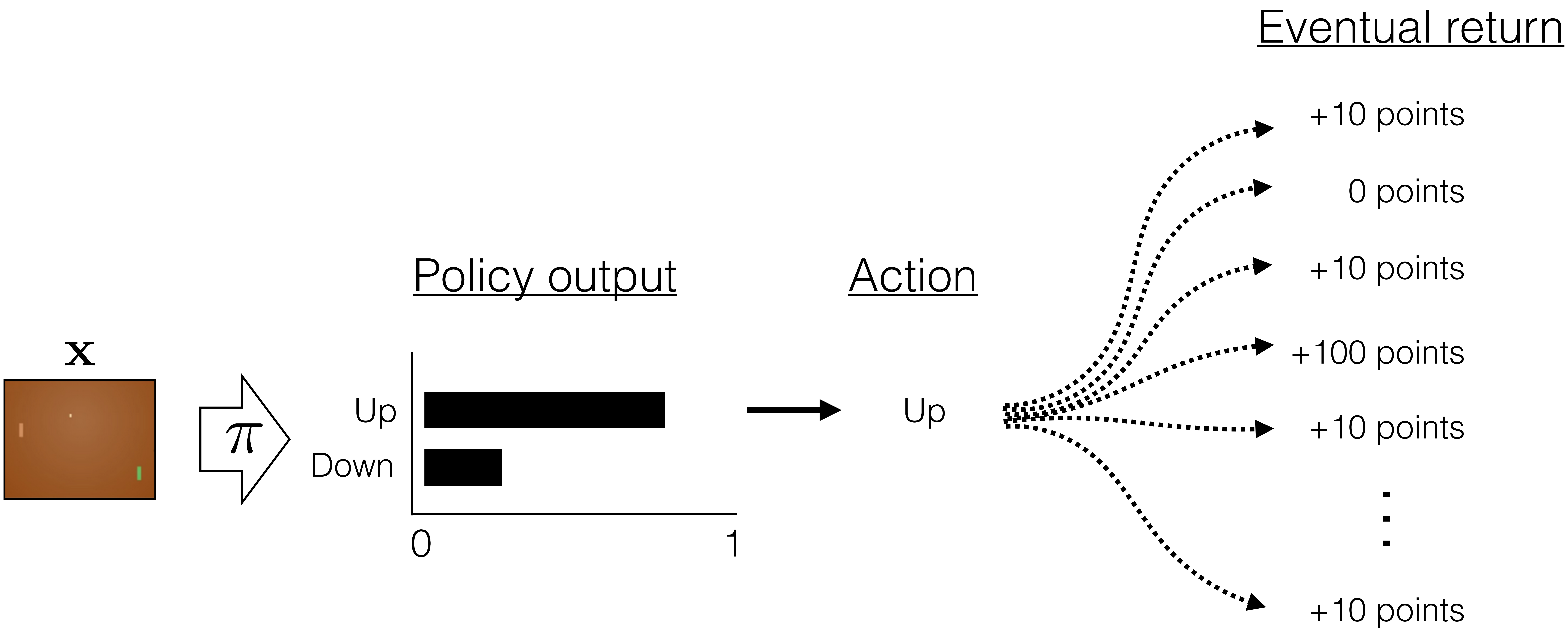
[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

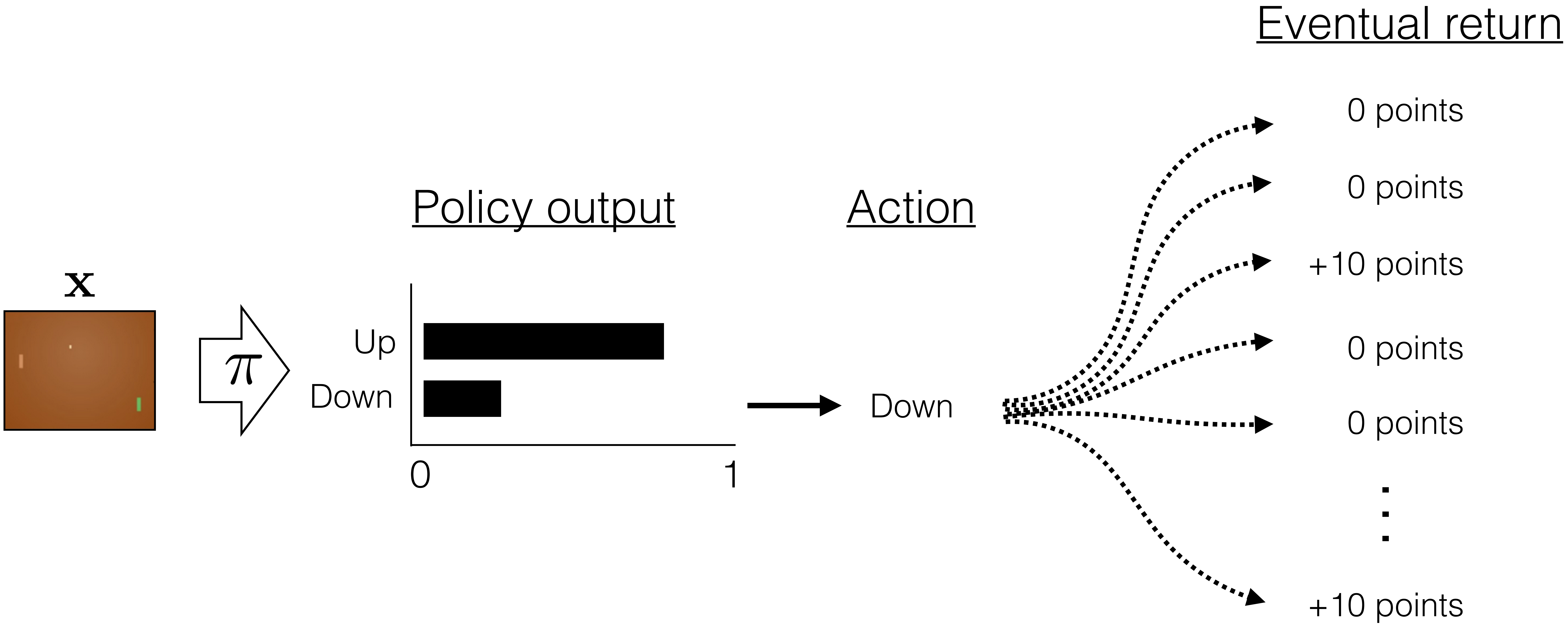
Policy gradients: Run a policy for a while. See what actions led to high rewards. Increase their probability.



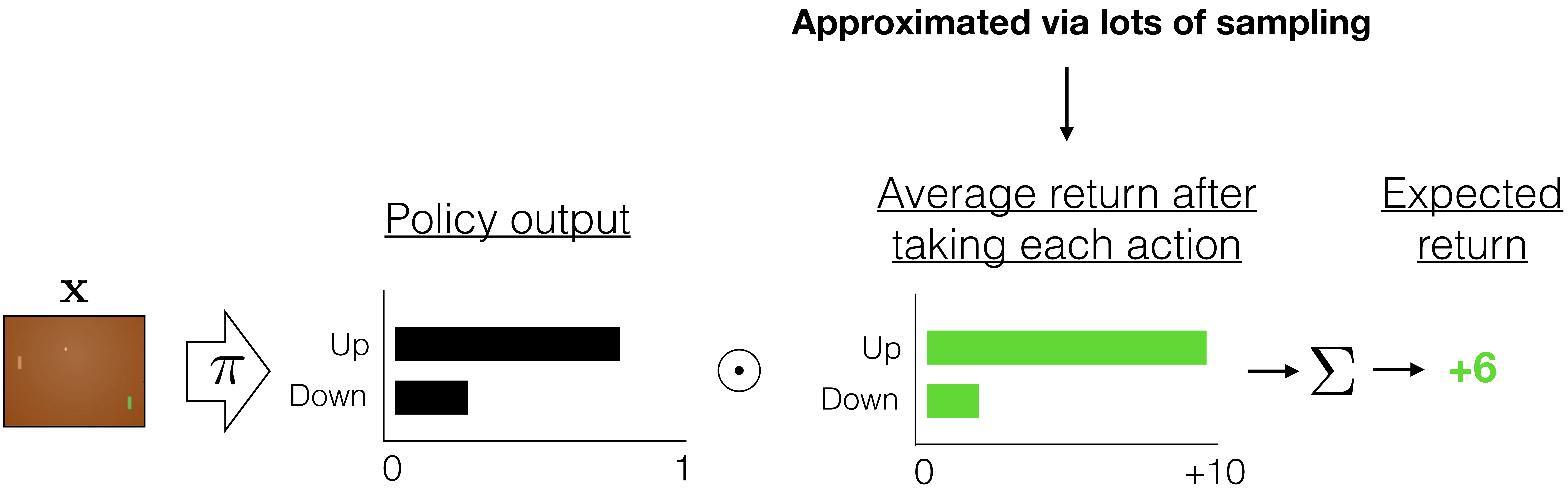
[Adapted from Andrej Karpathy: http://karpathy.github.io/2016/05/31/rl/]

Prediction $\hat{\mathbf{y}}$
$f_\theta : X \to \mathbb{R}^K$
Ground truth label $\mathbf{y}$
Loss
$H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$
$\mathbf{x}$
f
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
$\odot$
dolphin
cat
grizzly bear
angel fish
chameleon
clown fish
iguana
elephant
0
1
0
1
0


Eventual return
x
Policy output
Action
π
Up
Down
0
1
Up
+10 points
0 points
+10 points
+100 points
+10 points
+10 points



x
π
Policy output
Up
Down
0
1
Action
Down
Eventual return
0 points
0 points
+10 points
0 points
0 points
+10 points



$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau) \nabla_\theta \log \pi_\theta]$$ ← **Score function identity**

Environment is not differentiable! — How to optimize?

Policy gradients

1. Start with an arbitrary initial policy
2. **Rollout** this *stochastic* policy a bunch of times, sampling different random actions each time
3. Update your policy to place higher probability on actions that led to higher returns

Mathematically, this approximates gradient ascent on policy parameters, so as to maximize reward.

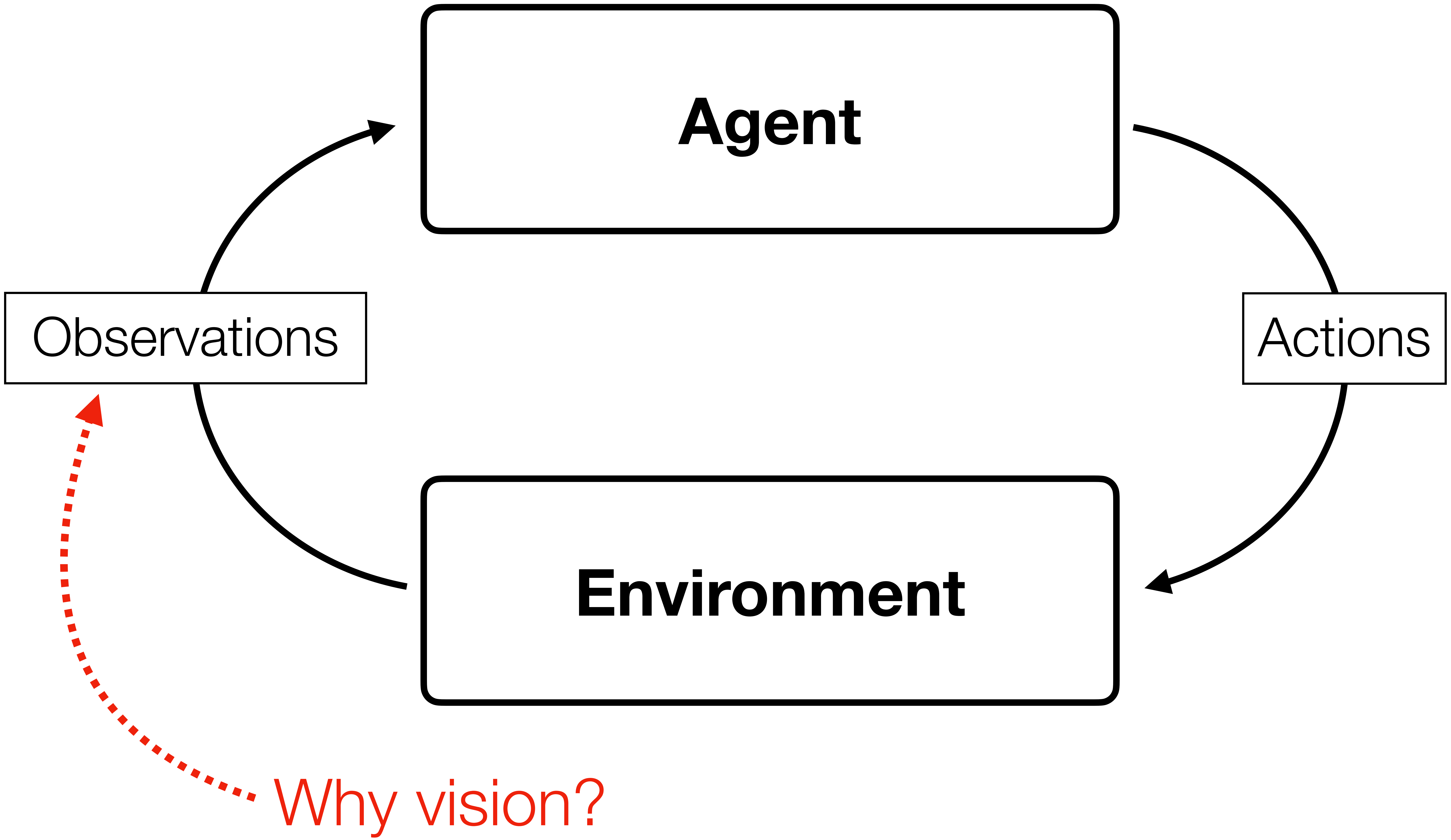
Reinforcement learning resources

[**Sutton & Barto**: http://incompleteideas.net/book/bookdraft2017nov5.pdf]

[**OpenAI Spinning Up**: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html]

[**Pong from pixels**: http://karpathy.github.io/2016/05/31/rl/]

Intelligent agents



Why vision?

(and audition, touch, etc… why perception?)

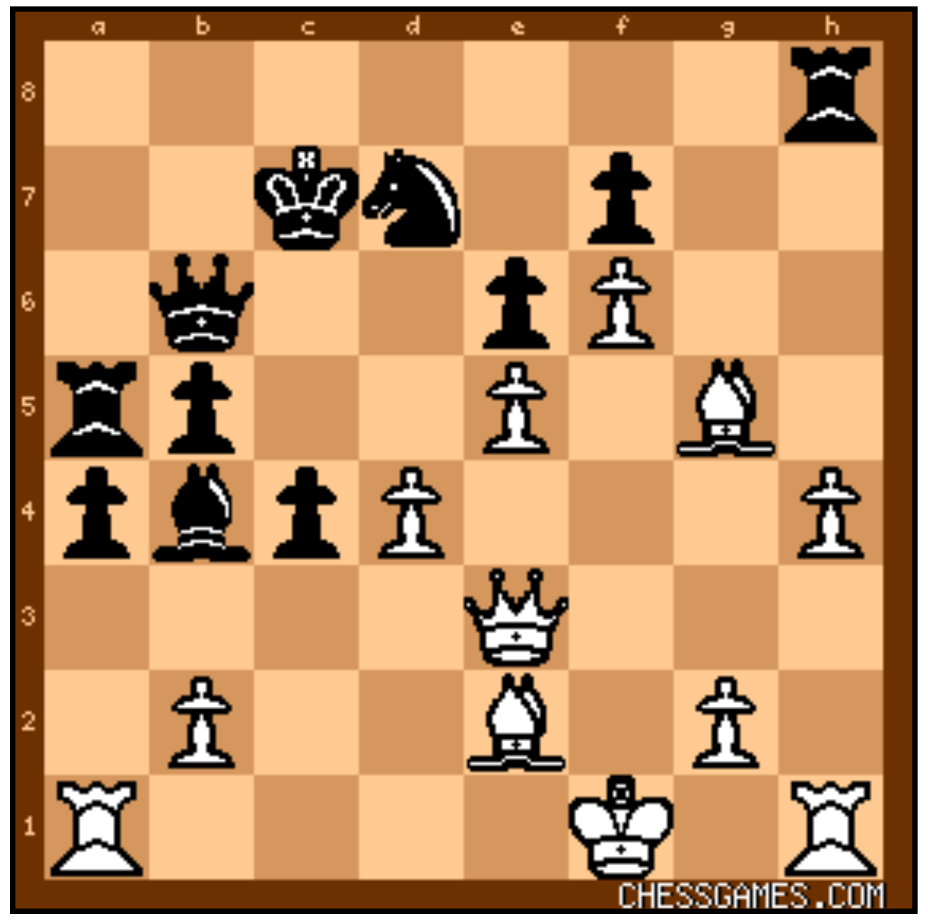
The brain's *model estimation* system

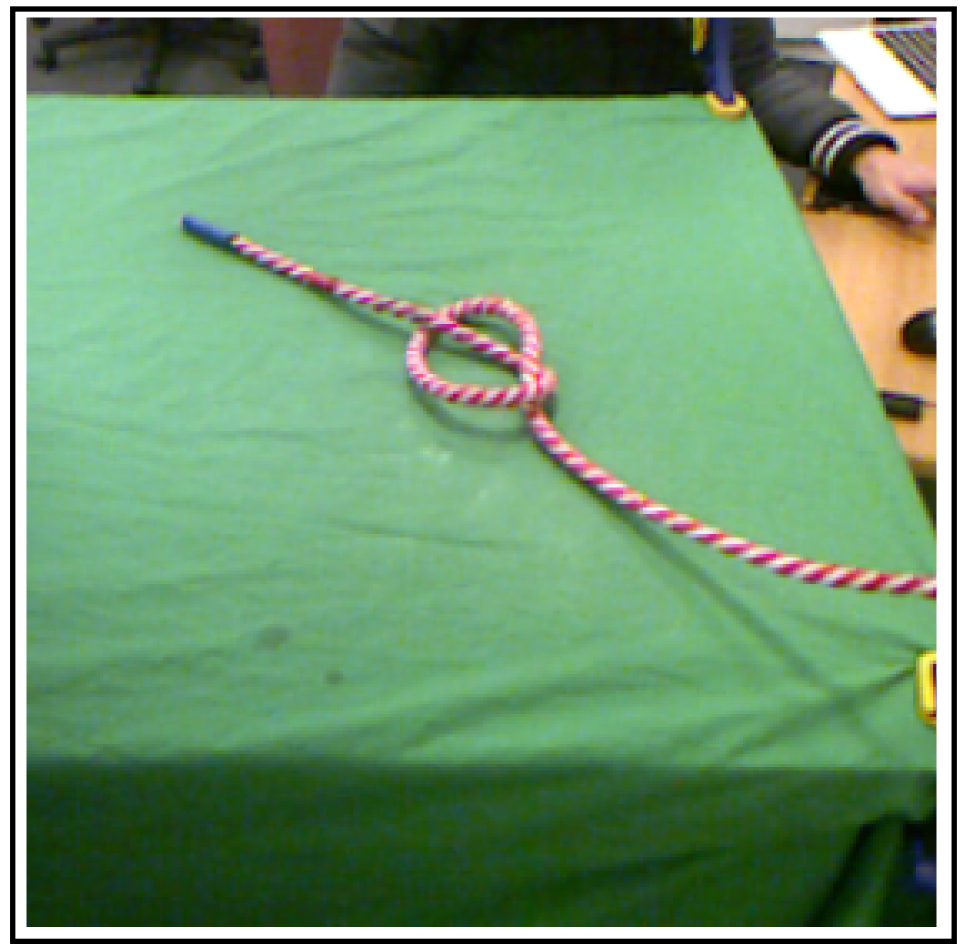
[Kanazawa, Tulsiani, et al., ECCV 2018]

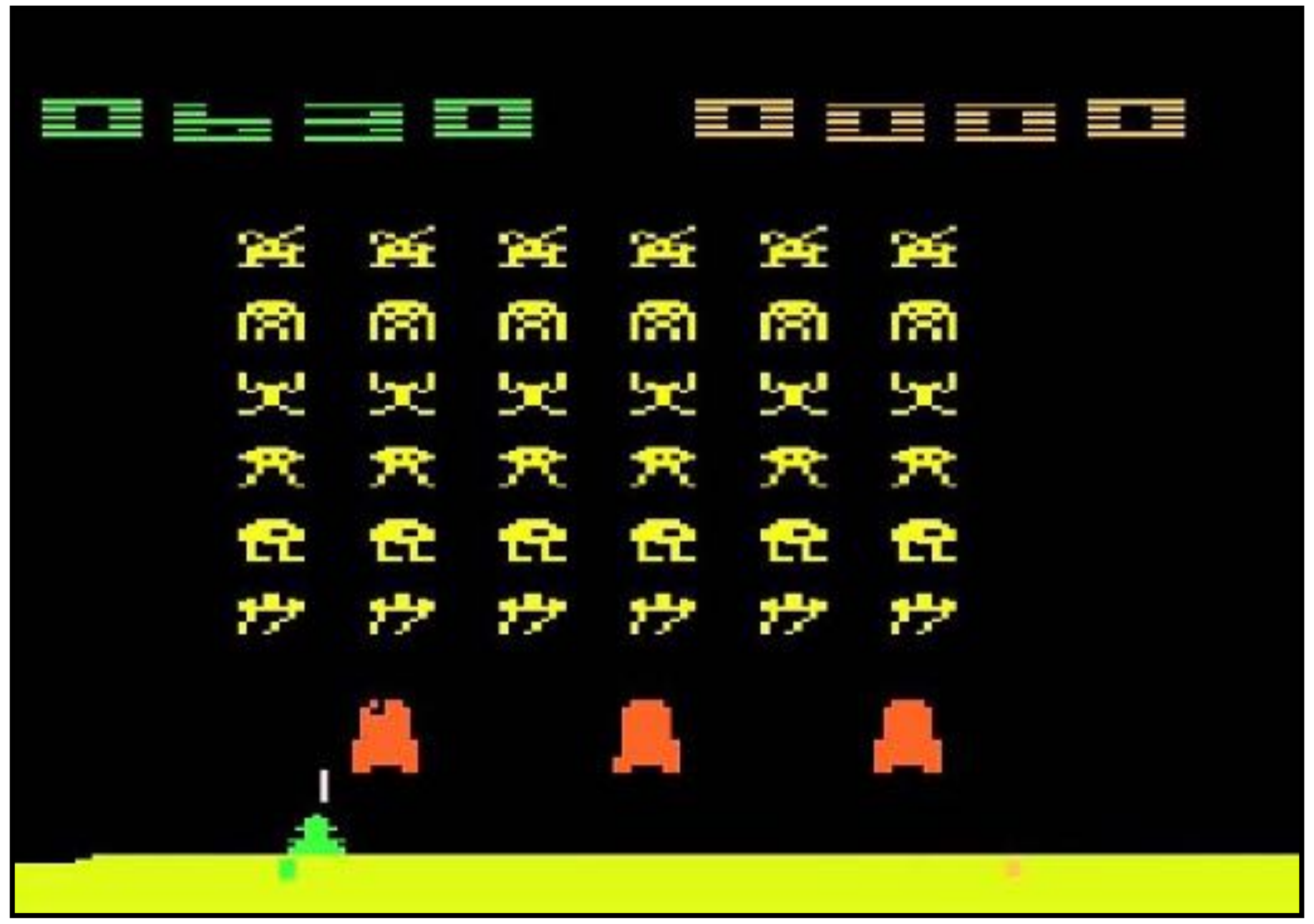
Why vision?

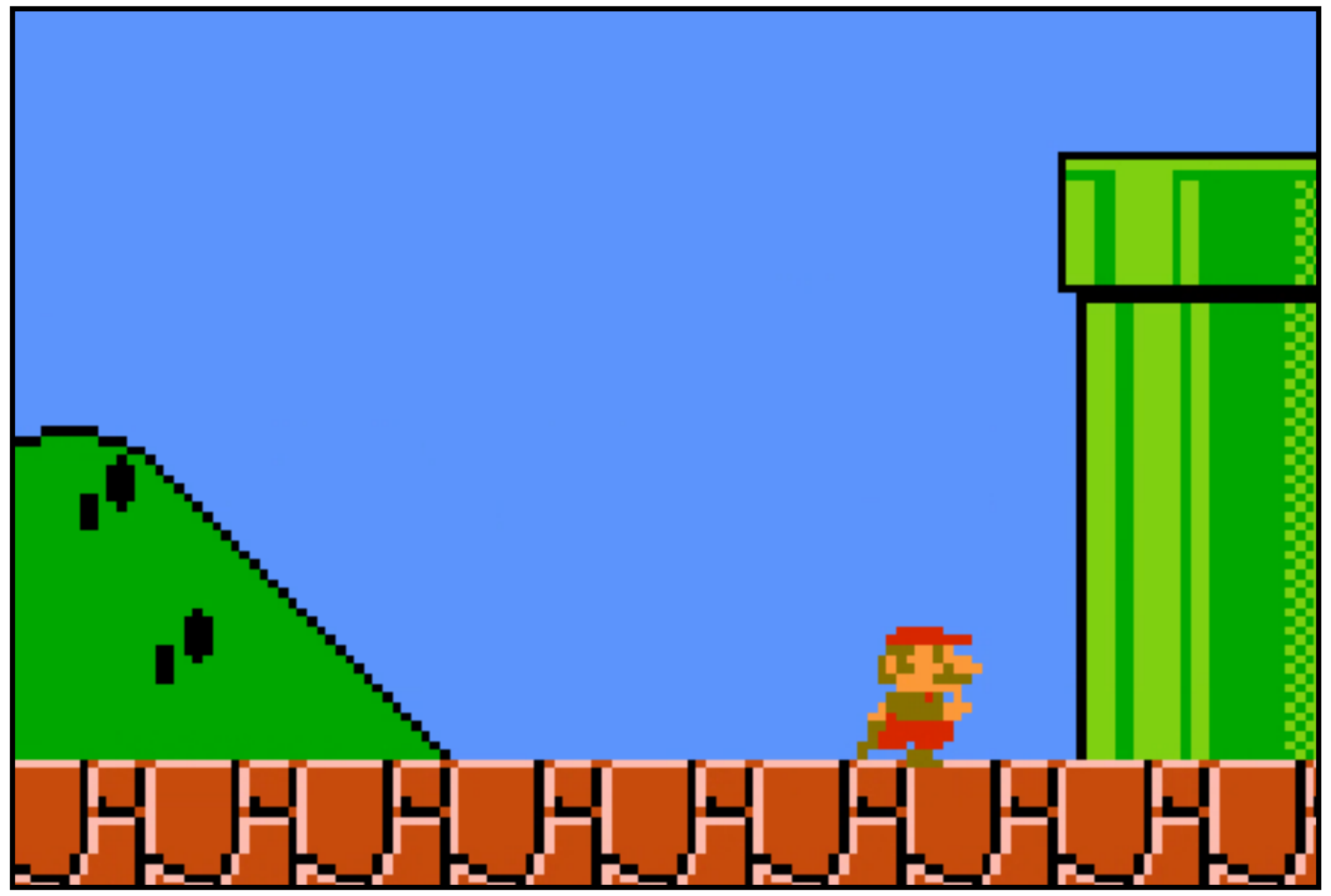
(and audition, touch, etc… why perception?)

Universal interface from external world to mental model

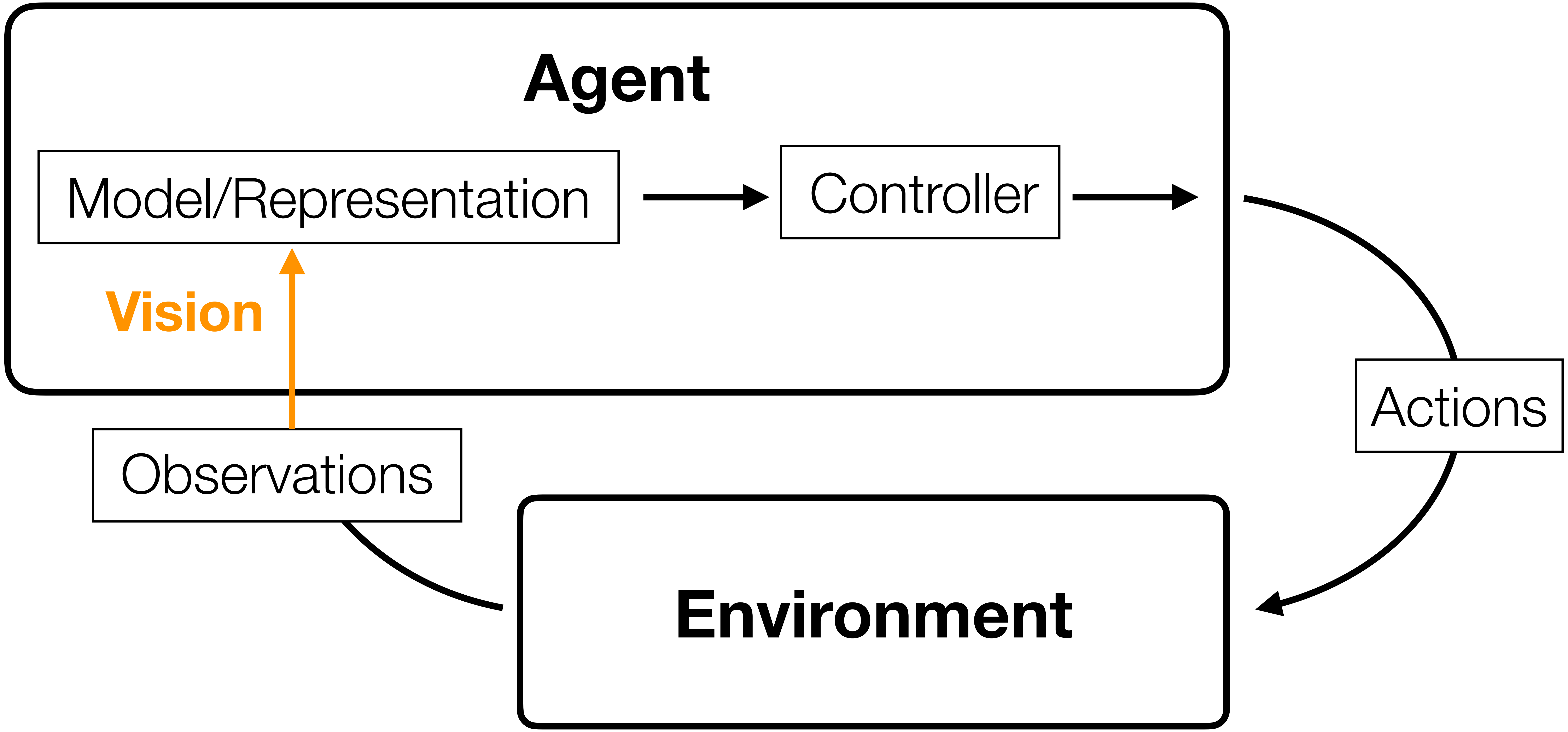




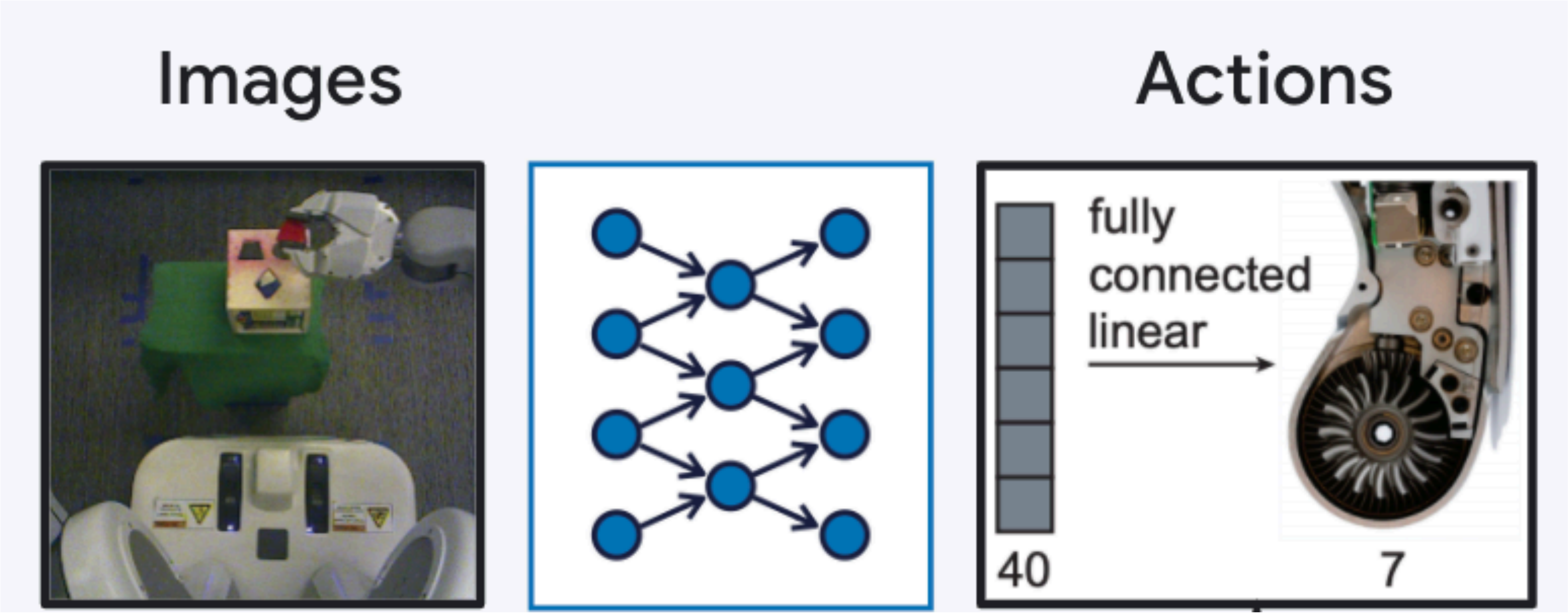




Intelligent agents



End-to-end Deep Reinforcement Learning

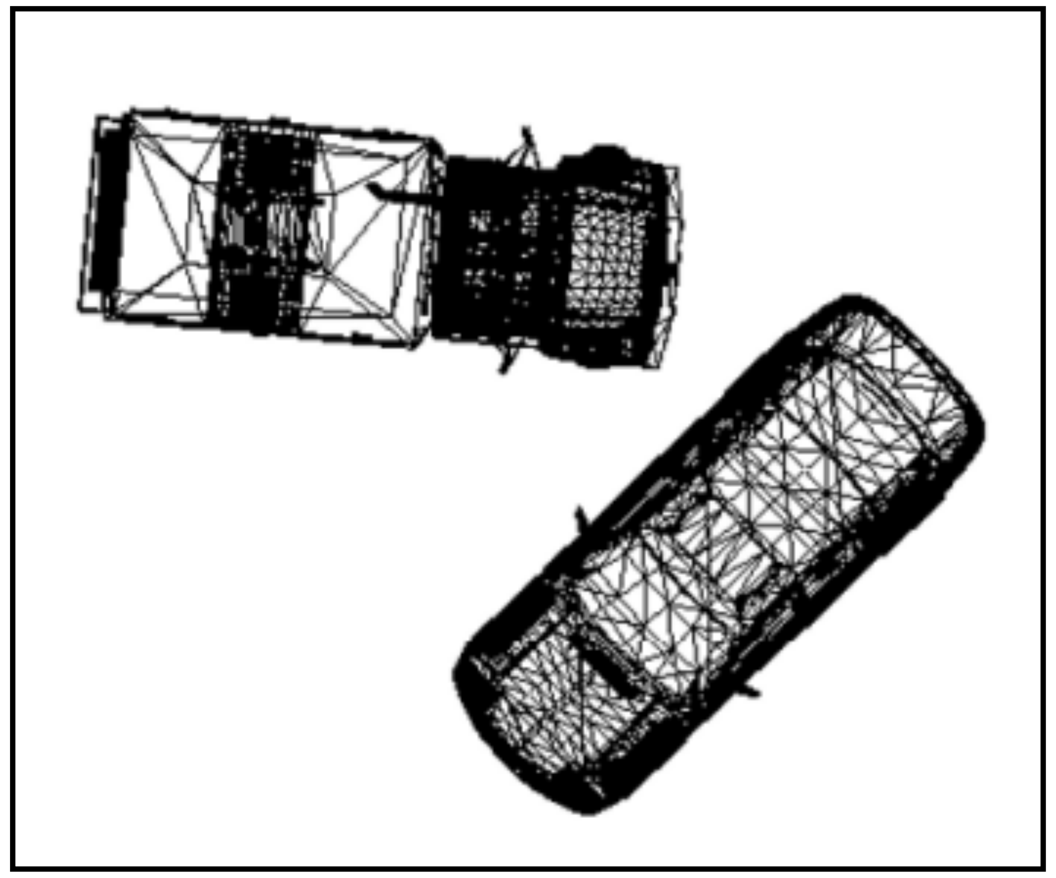


e.g., [Levine, Finn, et al. JMLR 2017]

[Slide adapted from Yen-Chen Lin]

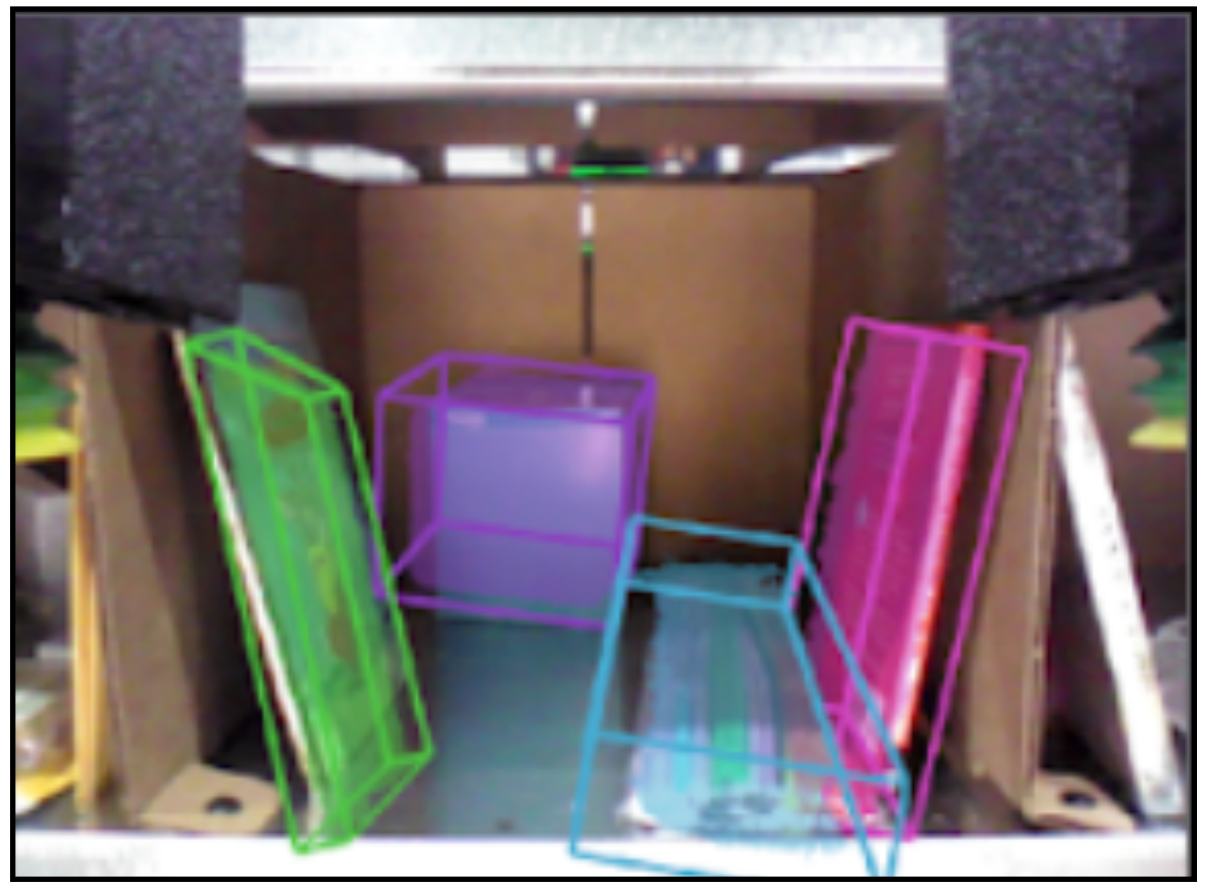
Object models for interaction

Meshes



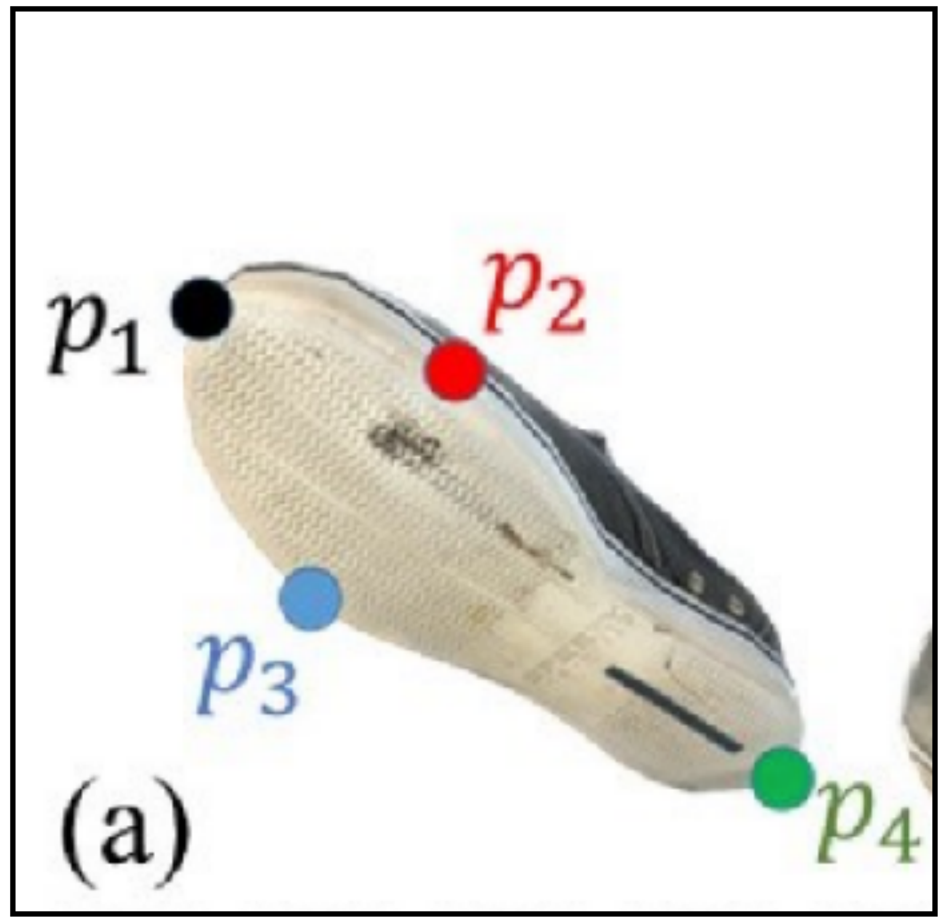
Palazzi et al. ECCV 2018

Pose
(3D bounding box)



Zeng et al. ICRA 2017

Keypoints



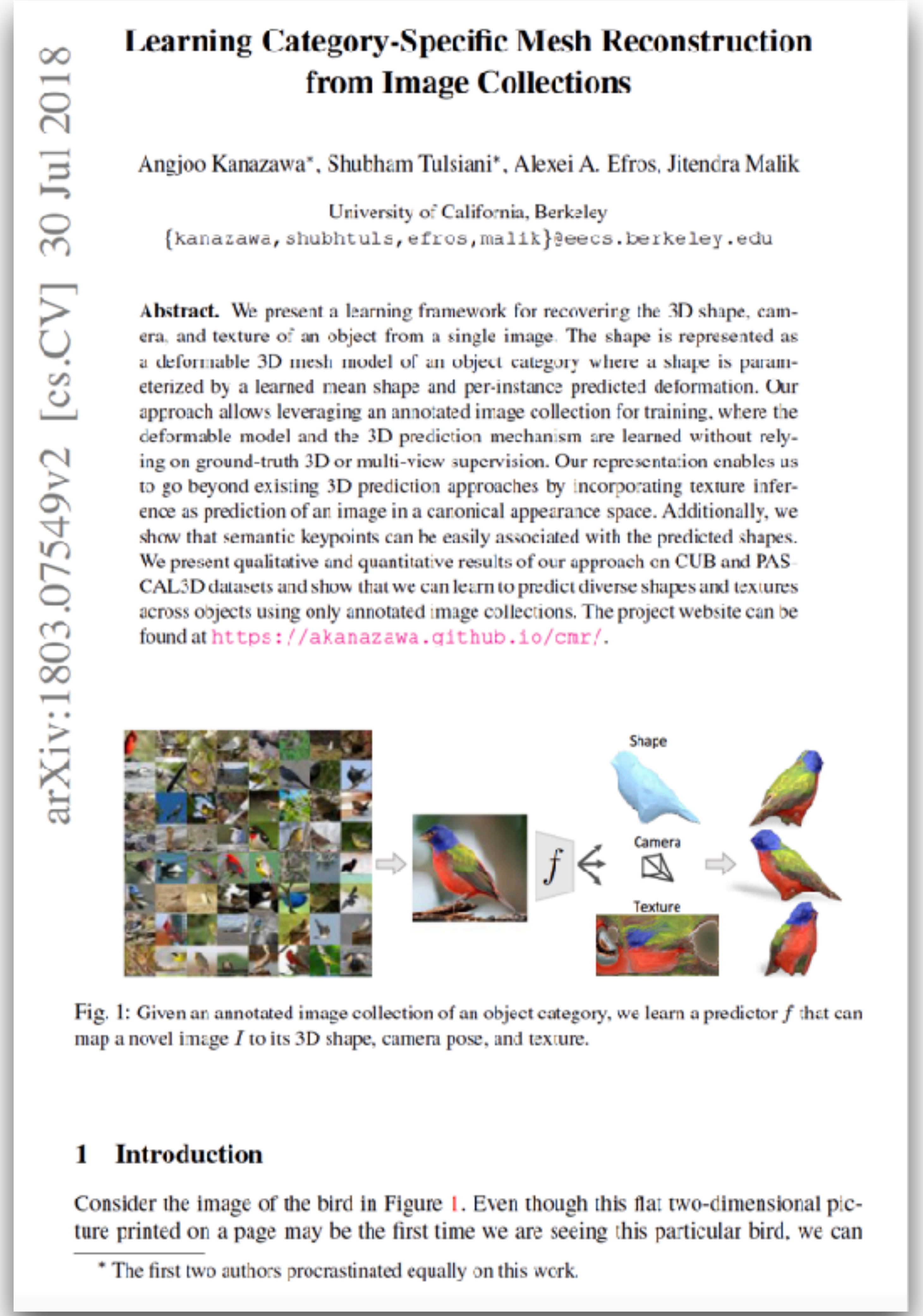
Manuelli et al. ISRR 2019

Dense descriptors

Florence et al. CoRL 2018

[Slide adapted from Yen-Chen Lin]

Learning 3D mesh models from photographs



arXiv:1803.07549v2 [cs.CV] 30 Jul 2018

Learning Category-Specific Mesh Reconstruction from Image Collections

Angjoo Kanazawa*, Shubham Tulsiani*, Alexei A. Efros, Jitendra Malik

University of California, Berkeley
{kanazawa,shubhtuls,efros,malik}@eecs.berkeley.edu

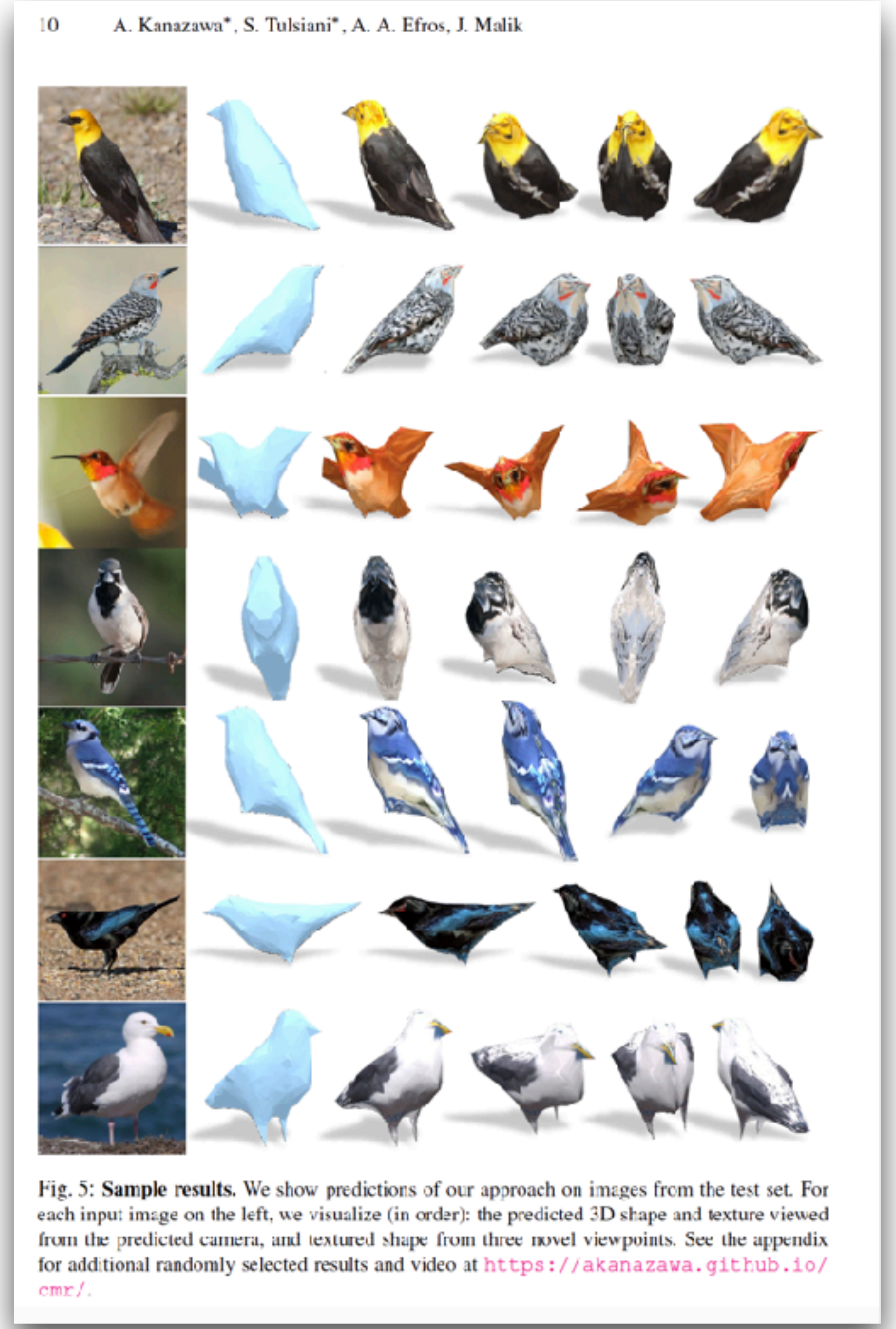
Abstract. We present a learning framework for recovering the 3D shape, camera, and texture of an object from a single image. The shape is represented as a deformable 3D mesh model of an object category where a shape is parameterized by a learned mean shape and per-instance predicted deformation. Our approach allows leveraging an annotated image collection for training, where the deformable model and the 3D prediction mechanism are learned without relying on ground-truth 3D or multi-view supervision. Our representation enables us to go beyond existing 3D prediction approaches by incorporating texture inference as prediction of an image in a canonical appearance space. Additionally, we show that semantic keypoints can be easily associated with the predicted shapes. We present qualitative and quantitative results of our approach on CUB and PASCAL3D datasets and show that we can learn to predict diverse shapes and textures across objects using only annotated image collections. The project website can be found at https://akanazawa.github.io/cmr/.

Fig. 1: Given an annotated image collection of an object category, we learn a predictor f that can map a novel image I to its 3D shape, camera pose, and texture.

1 Introduction

Consider the image of the bird in Figure 1. Even though this flat two-dimensional picture printed on a page may be the first time we are seeing this particular bird, we can

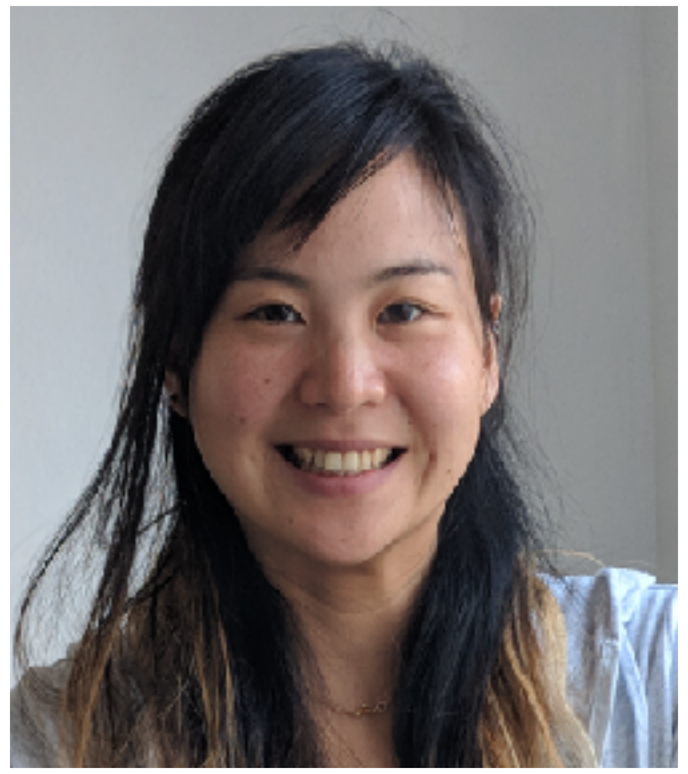
* The first two authors procrastinated equally on this work.



10 A. Kanazawa*, S. Tulsiani*, A. A. Efros, J. Malik

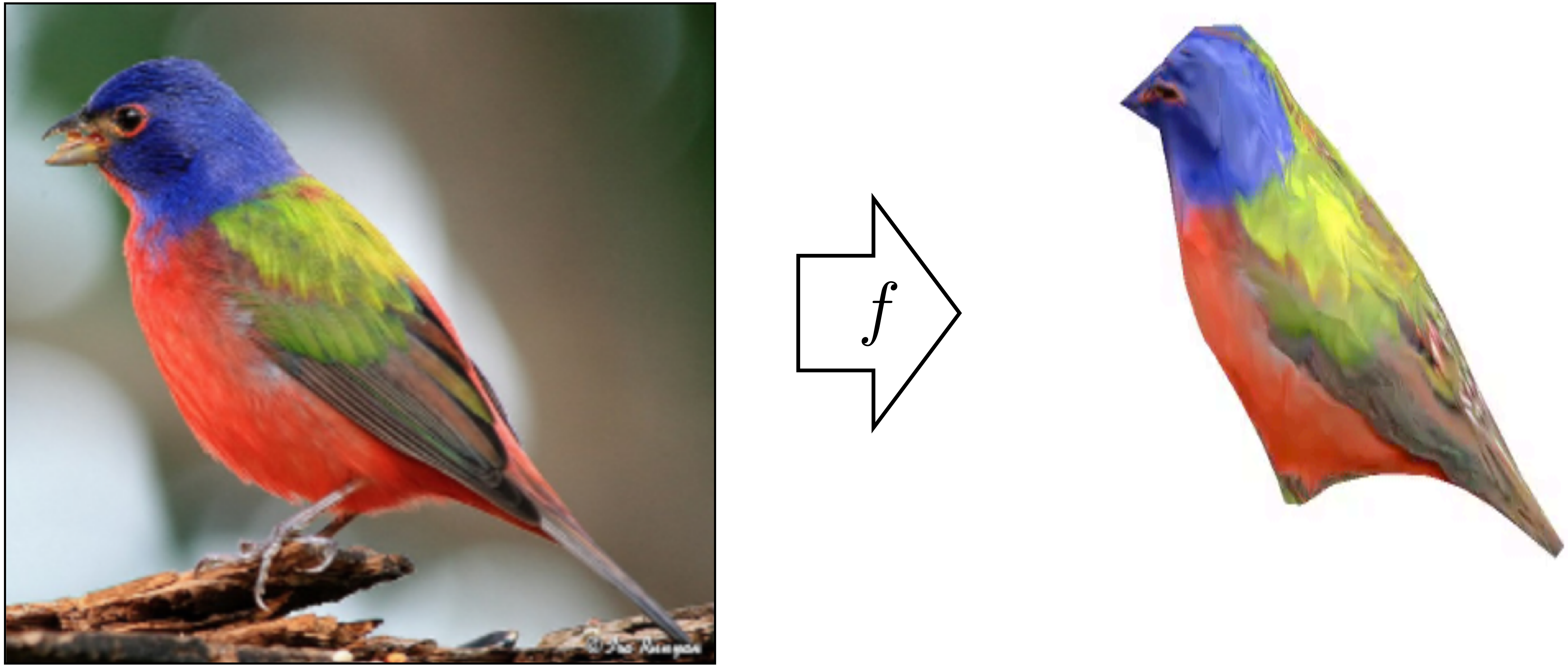
Fig. 5: **Sample results.** We show predictions of our approach on images from the test set. For each input image on the left, we visualize (in order): the predicted 3D shape and texture viewed from the predicted camera, and textured shape from three novel viewpoints. See the appendix for additional randomly selected results and video at https://akanazawa.github.io/cmr/.

ECCV, 2018



Angjoo Kanazawa

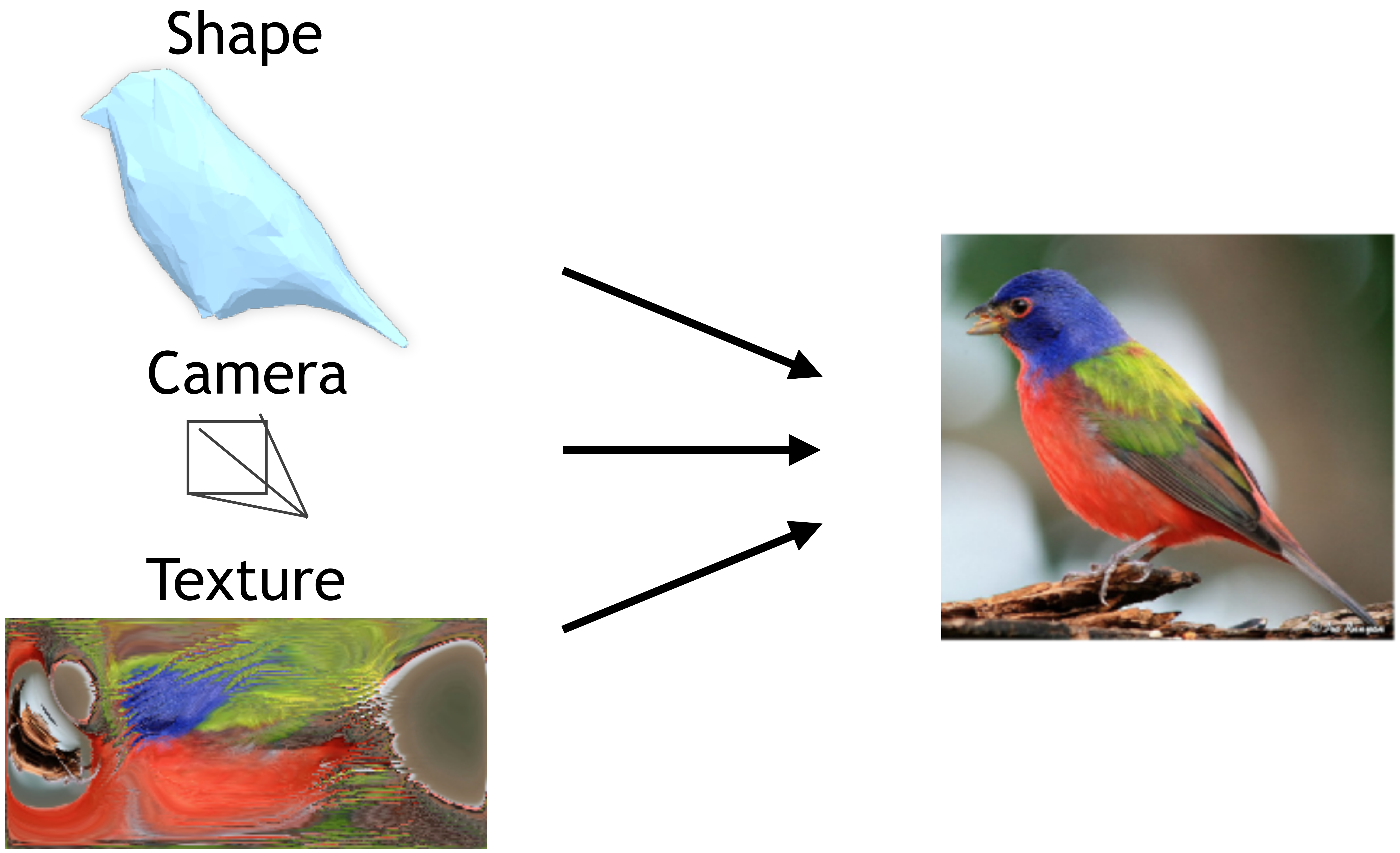
Object modeling



[Kanazawa, Tulsiani, et al., ECCV 2018]

Analysis by synthesis

Find a [shape, camera, texture] combination (analysis) that renders to the image (synthesis).

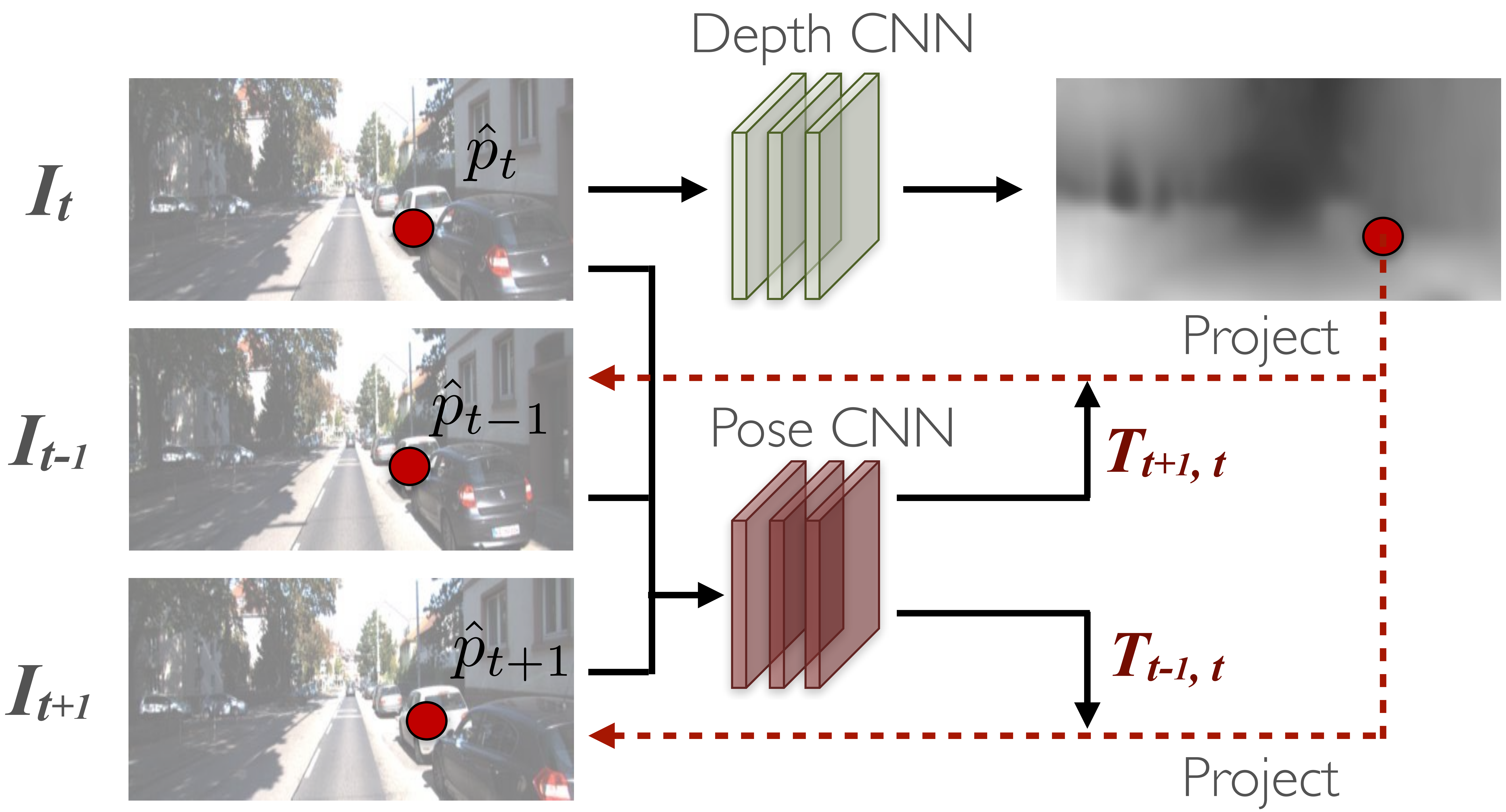


Recall: “Unsupervised” camera motion and monocular depth

Depth estimation

Pose estimation

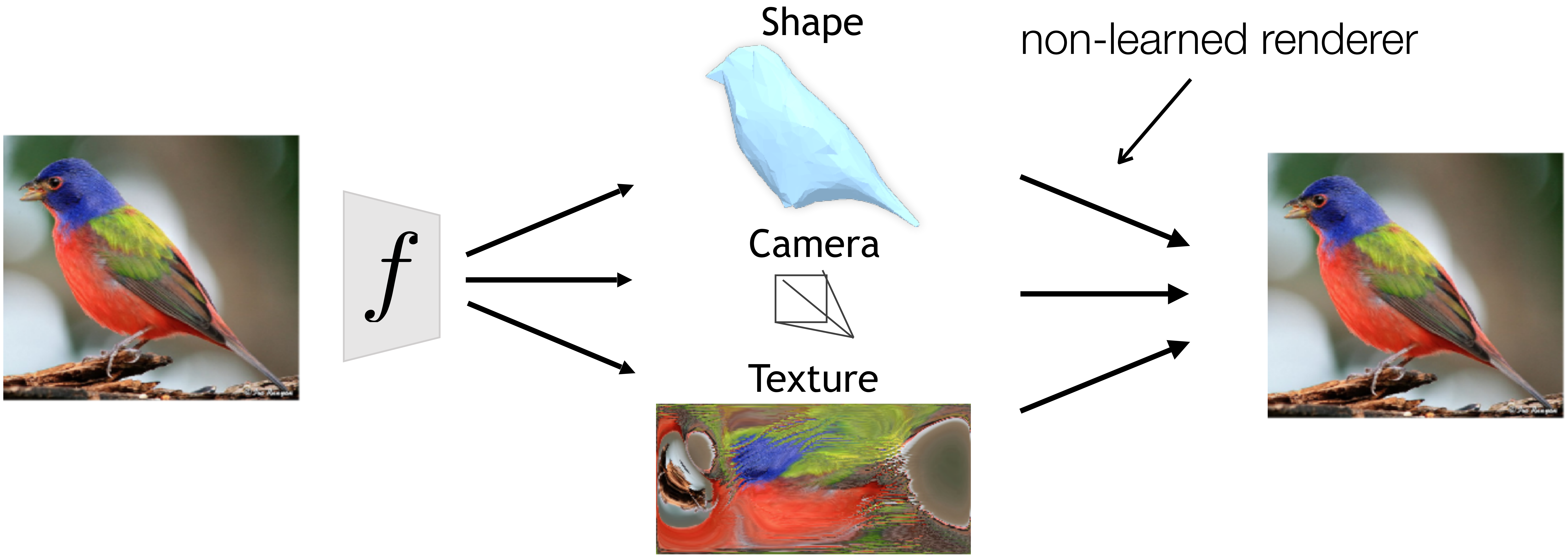
View synthesis



Training Loss: $$\mathcal{L}_{vs} = \sum_{s\in\{\text{nearby frames}\}} \sum_{p} |I_t(p_t) - I_s(\hat{p}_s)|$$

Funny looking autoencoder

Find a [shape, camera, texture] combination that renders to the image.



Training

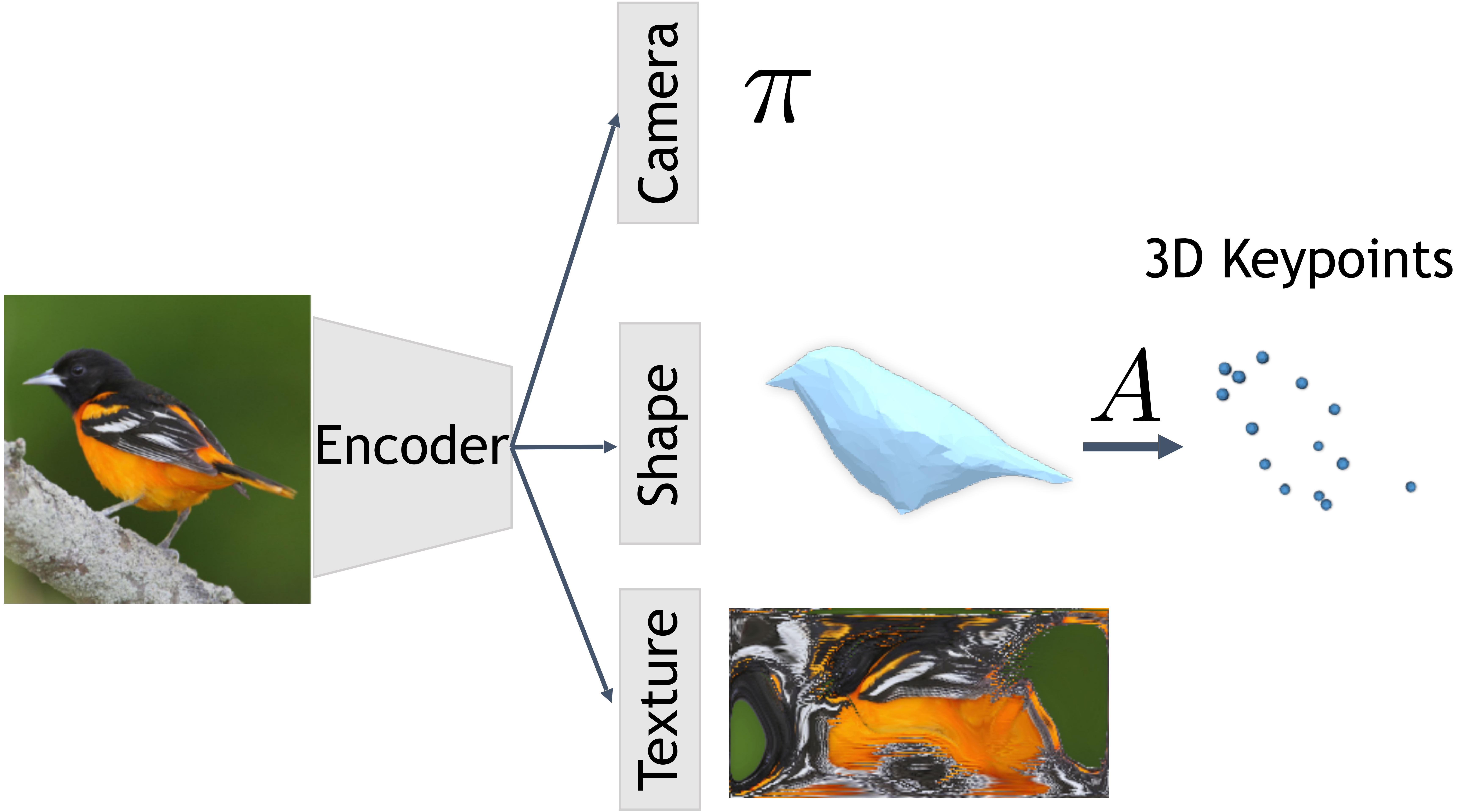
Keypoints

Segmentation mask

Learn 3D only from 2D image-based annotations

Many images are only seen under a single view point

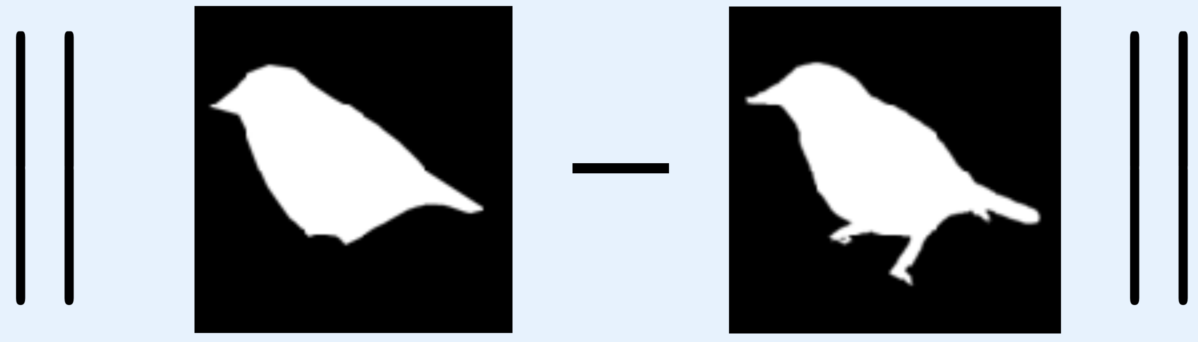
Approach



Losses:
Predicted, GT

Texture:

Mask:

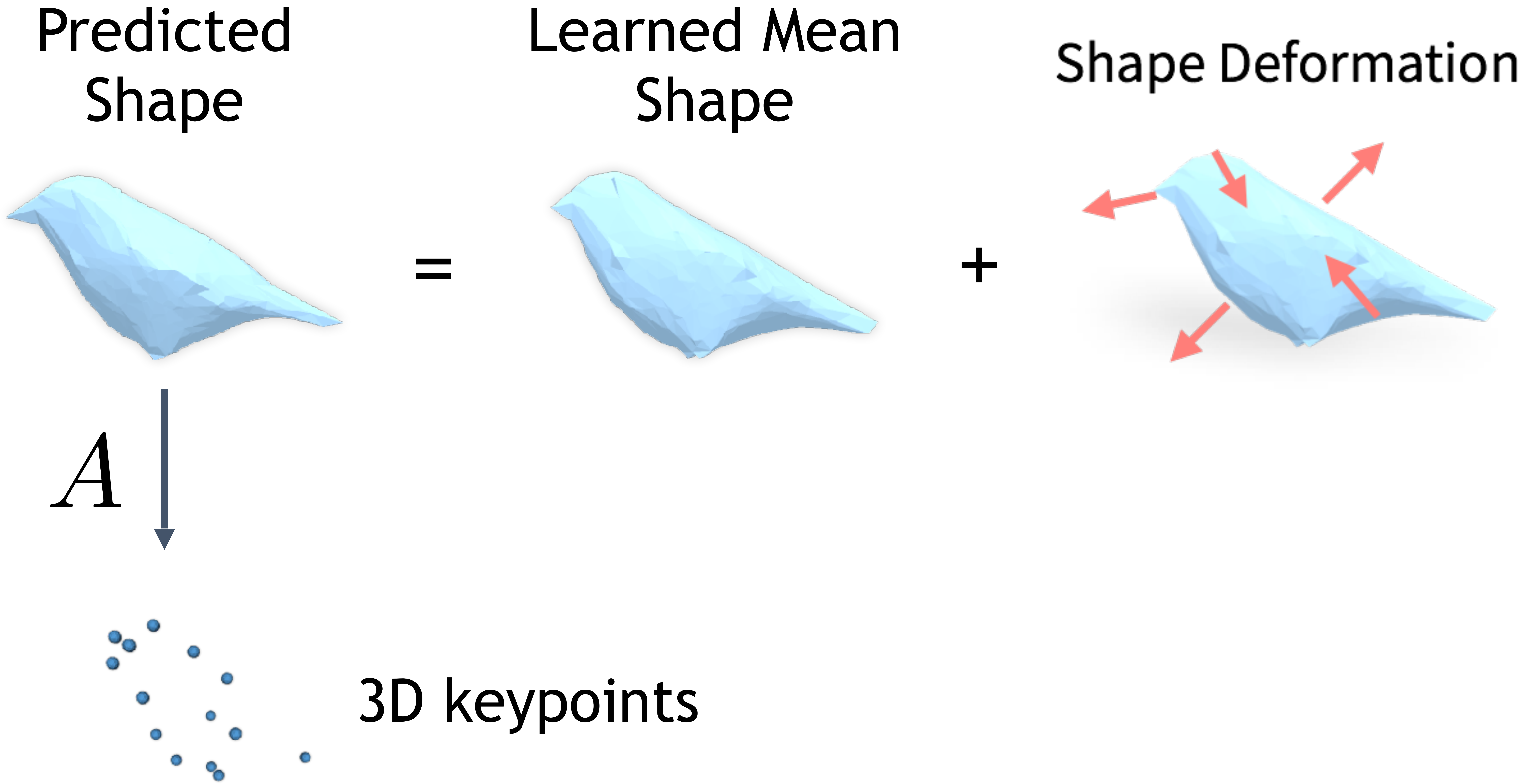


SfM Camera:

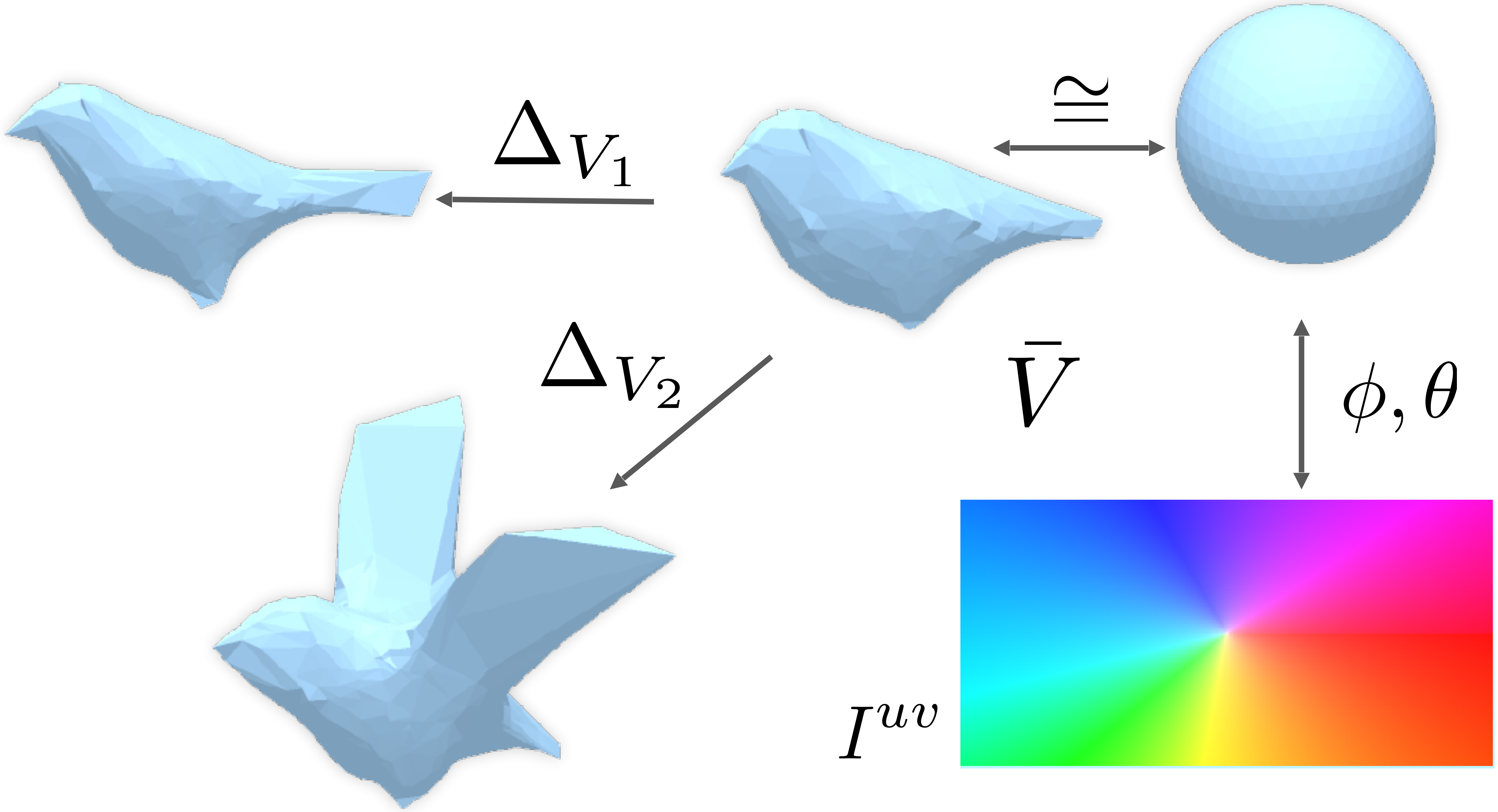
$$\left\| \pi - \pi^{\mathrm{sfm}} \right\|$$

Keypoints:

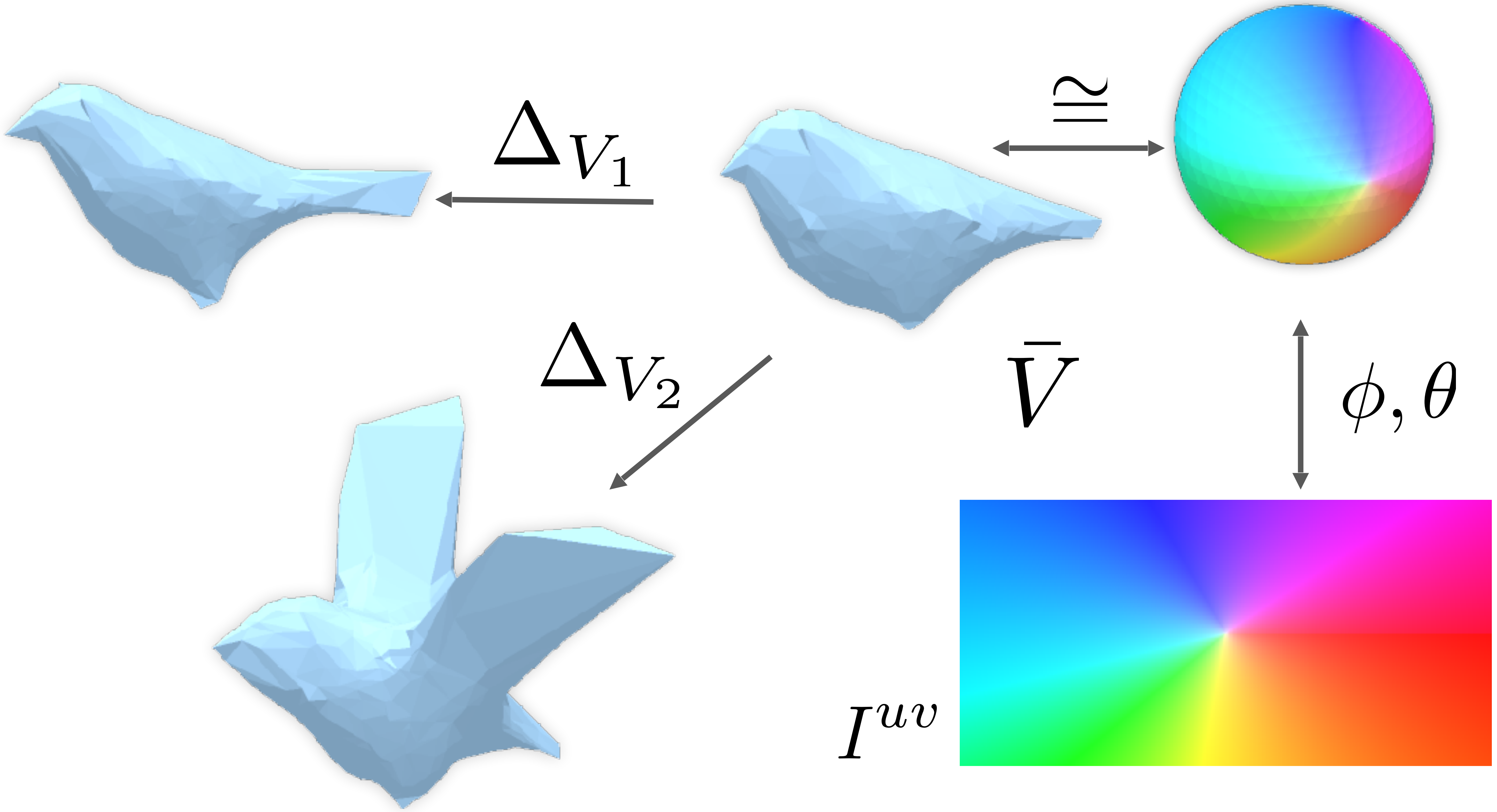
Shape Representation



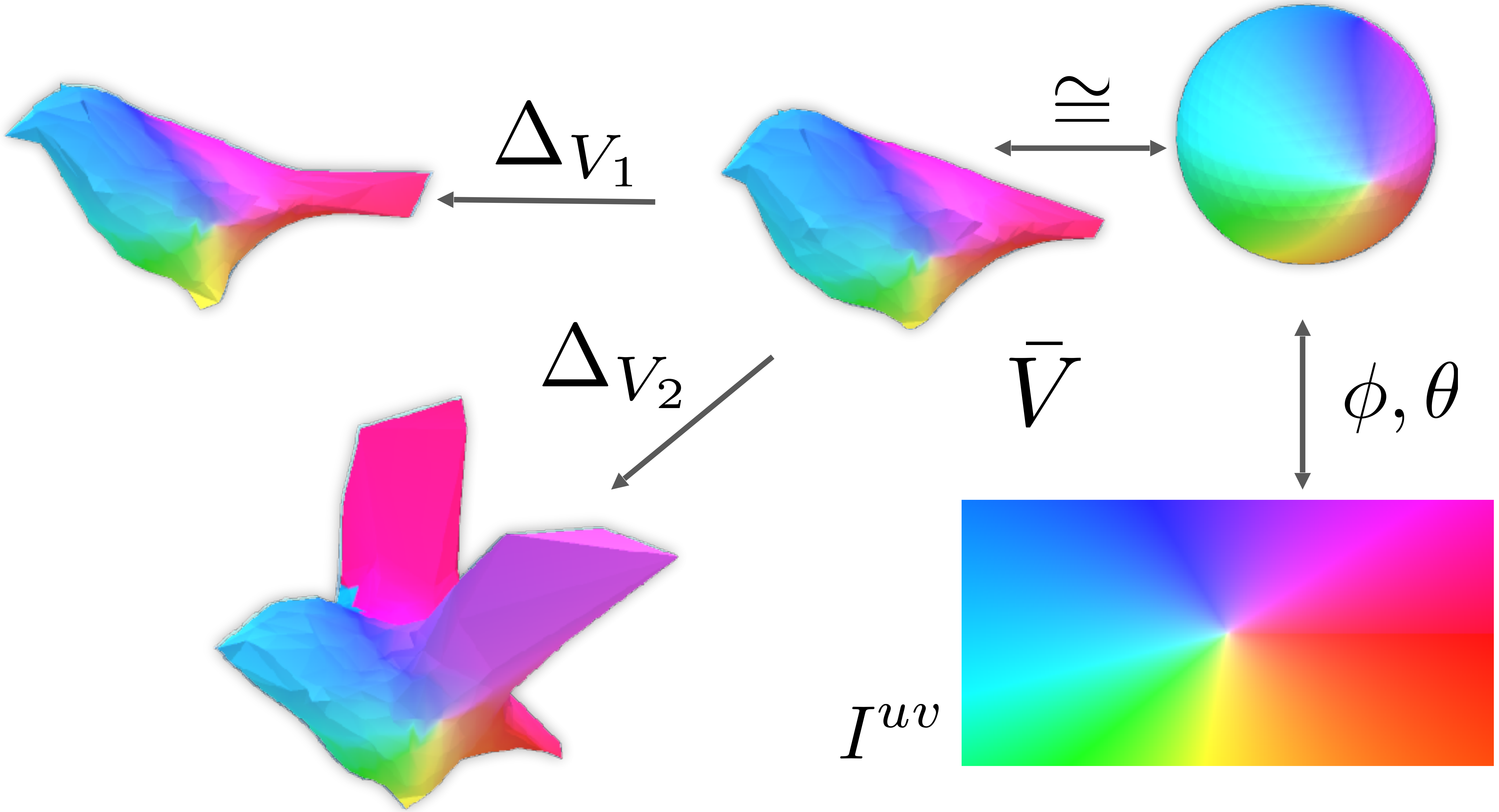
Texture Representation



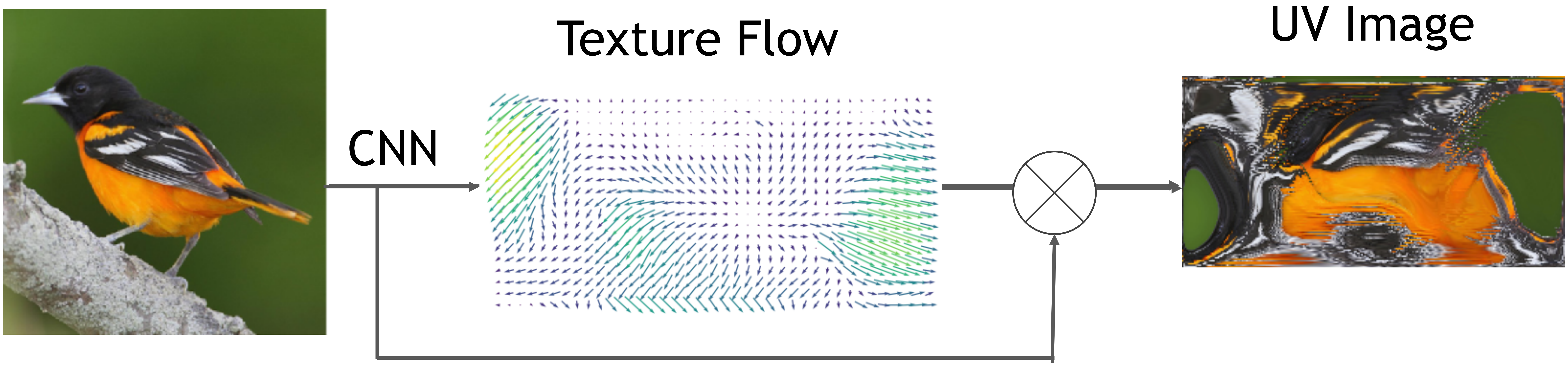
Texture Representation



Texture Representation



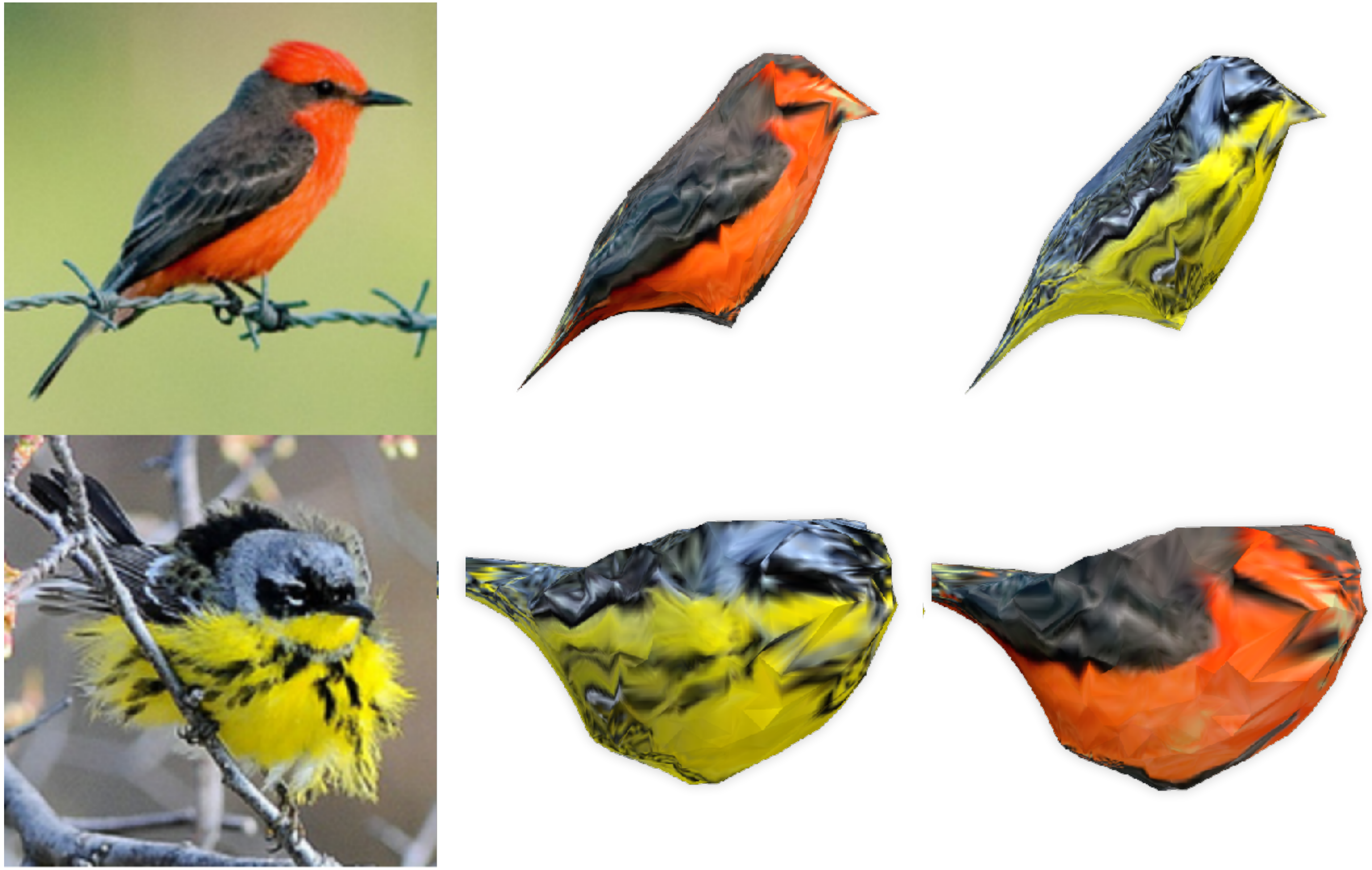
Texture as UV Image Prediction



Results

Texture Transfer

Texture Transfer



Dense object descriptors

arXiv:1806.08756v2 [cs.RO] 7 Sep 2018

Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation

Peter R. Florence*, Lucas Manuelli*, Russ Tedrake
CSAIL, Massachusetts Institute of Technology
{peteflo,manuelli,russt}@csail.mit.edu
**These authors contributed equally to this work.*

Abstract:
What is the right object representation for manipulation? We would like robots to visually perceive scenes and learn an understanding of the objects in them that (i) is task-agnostic and can be used as a building block for a variety of manipulation tasks, (ii) is generally applicable to both rigid and non-rigid objects, (iii) takes advantage of the strong priors provided by 3D vision, and (iv) is entirely learned from self-supervision. This is hard to achieve with previous methods: much recent work in grasping does not extend to grasping specific objects or other tasks, whereas task-specific learning may require many trials to generalize well across object configurations or other tasks. In this paper we present Dense Object Nets, which build on recent developments in self-supervised dense descriptor learning, as a consistent object representation for visual understanding and manipulation. We demonstrate they can be trained quickly (approximately 20 minutes) for a wide variety of previously unseen and potentially non-rigid objects. We additionally present novel contributions to enable multi-object descriptor learning, and show that by modifying our training procedure, we can either acquire descriptors which generalize across classes of objects, or descriptors that are distinct for each object instance. Finally, we demonstrate the novel application of learned dense descriptors to robotic manipulation. We demonstrate grasping of specific points on an object across potentially deformed object configurations, and demonstrate using class general descriptors to transfer specific grasps across objects in a class.

Keywords: Visual Descriptor Learning, Self-Supervision, Robot Manipulation

1 Introduction

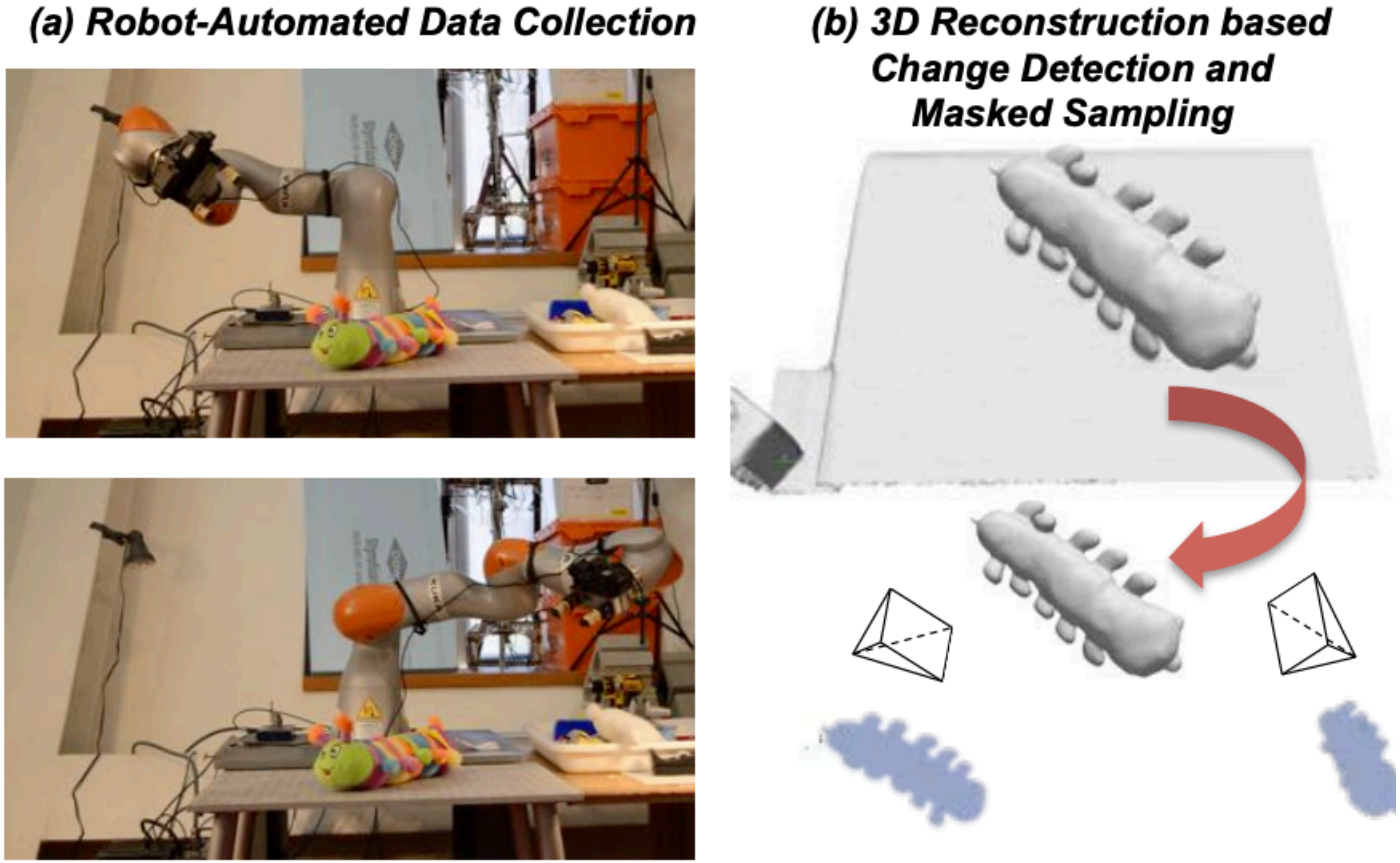
What is the right object representation for manipulation? While task-specific reinforcement learning can achieve impressively dexterous skills for a given specific task [1], it is unclear which is the best route to efficiently achieving many different tasks. Other recent work [2, 3] can provide very general grasping functionality but does not address specificity. Achieving specificity, the ability to accomplish specific tasks with specific objects, may require solving the data association problem. At a coarse level the task of identifying and manipulating individual objects can be solved by instance segmentation, as demonstrated in the Amazon Robotics Challenge (ARC) [4, 5] or [6]. Whole-object-level segmentation, however, does not provide any information on the rich structure of the objects themselves, and hence may not be an appropriate representation for solving more complex tasks. While not previously applied to the robotic manipulation domain, recent work has demonstrated advances in learning dense pixel level data association [7, 8], including self-supervision from raw RGBD data [8], which inspired our present work.

In this paper, we propose and demonstrate using dense visual description as a representation for robotic manipulation. We demonstrate the first autonomous system that can entirely self-supervise to learn consistent dense visual representations of objects, and ours is the first system we know of that is capable of performing the manipulation demonstrations we provide. Specifically, with no human supervision during training, our system can grasp specific locations on deformable objects, grasp semantically corresponding locations on instances in a class, and grasp specific locations on specific instances in clutter. Towards this goal, we also provide practical contributions to dense visual descriptor learning with general computer

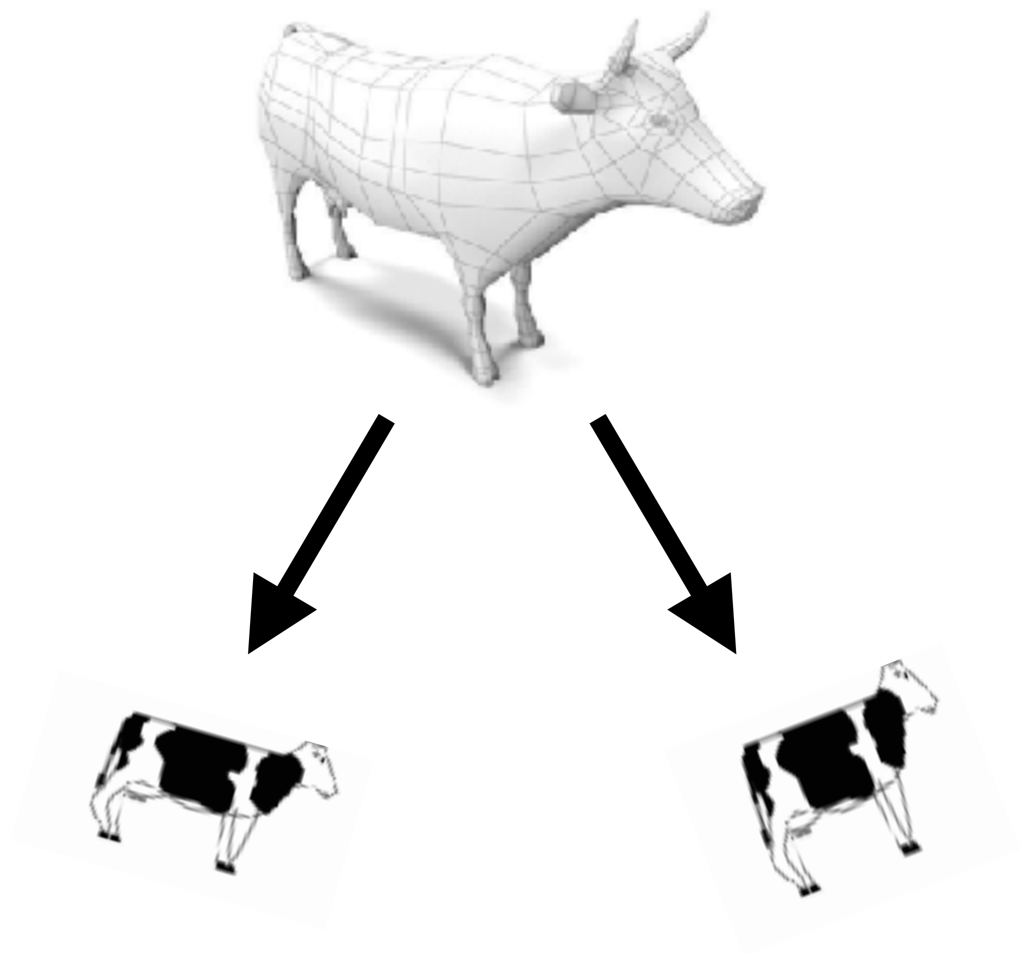
Video and source code available at https://youtu.be/L5UW1VapKNE and https://github.com/RobotLocomotion/pytorch-dense-correspondence.

Lucas Manuelli

Every pixel is a learned feature vector!


(a) Robot-Automated Data Collection
(b) 3D Reconstruction based Change Detection and Masked Sampling

Multiview self-supervised **contrastive** learning



Data

$\{\mathbf{x}^{(i)}\}_{i=1}^{N} \quad \rightarrow$

Learner

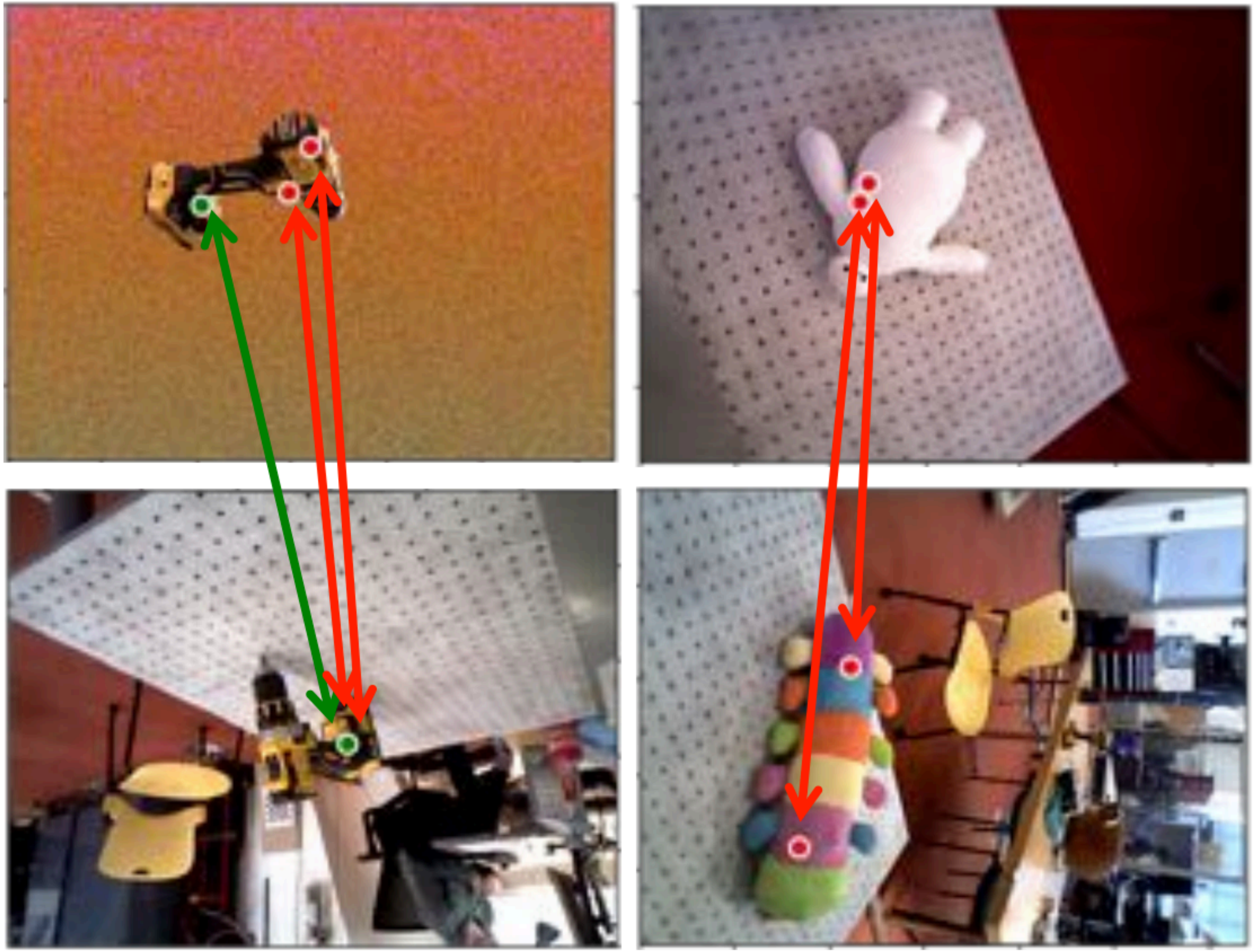
Objective

$$\arg\min_{f} \sum_{i,j} D(f_1(g(\mathbf{x}^{\boxed{(i)}})), f_2(h(\mathbf{x}^{\boxed{(i)}}))+$$
$$-D(f_1(g(\mathbf{x}^{\boxed{(i)}})), f_2(h(\mathbf{x}^{\boxed{(j)}})))$$

$\rightarrow \; f_1, f_2$

Distance function

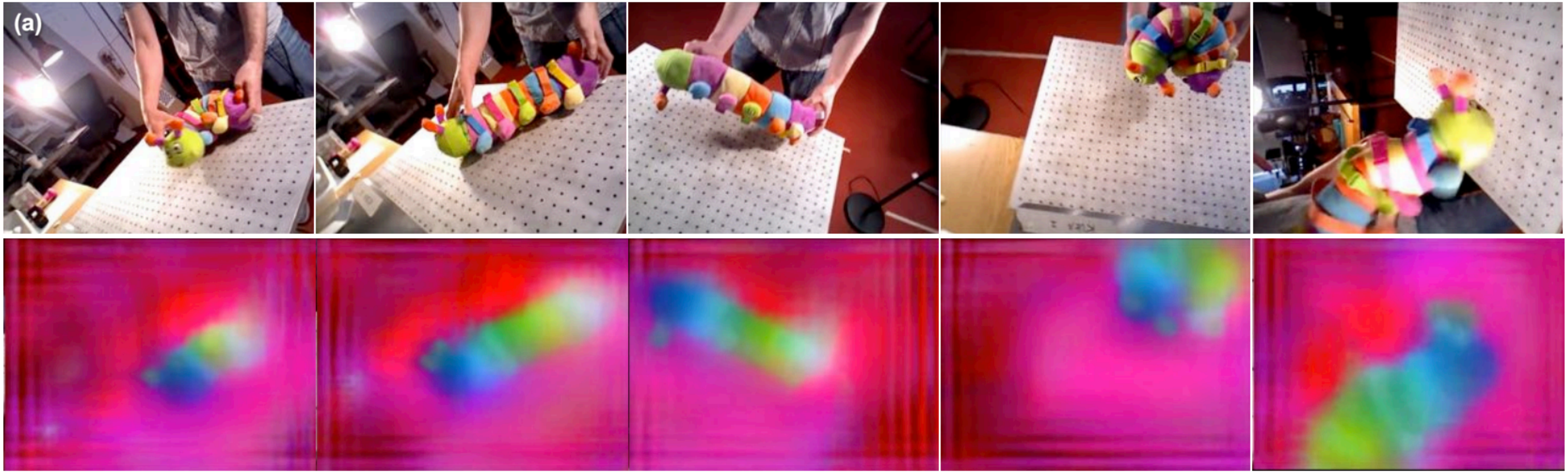
g and h are two **views** of the data x,
e.g., two different camera views



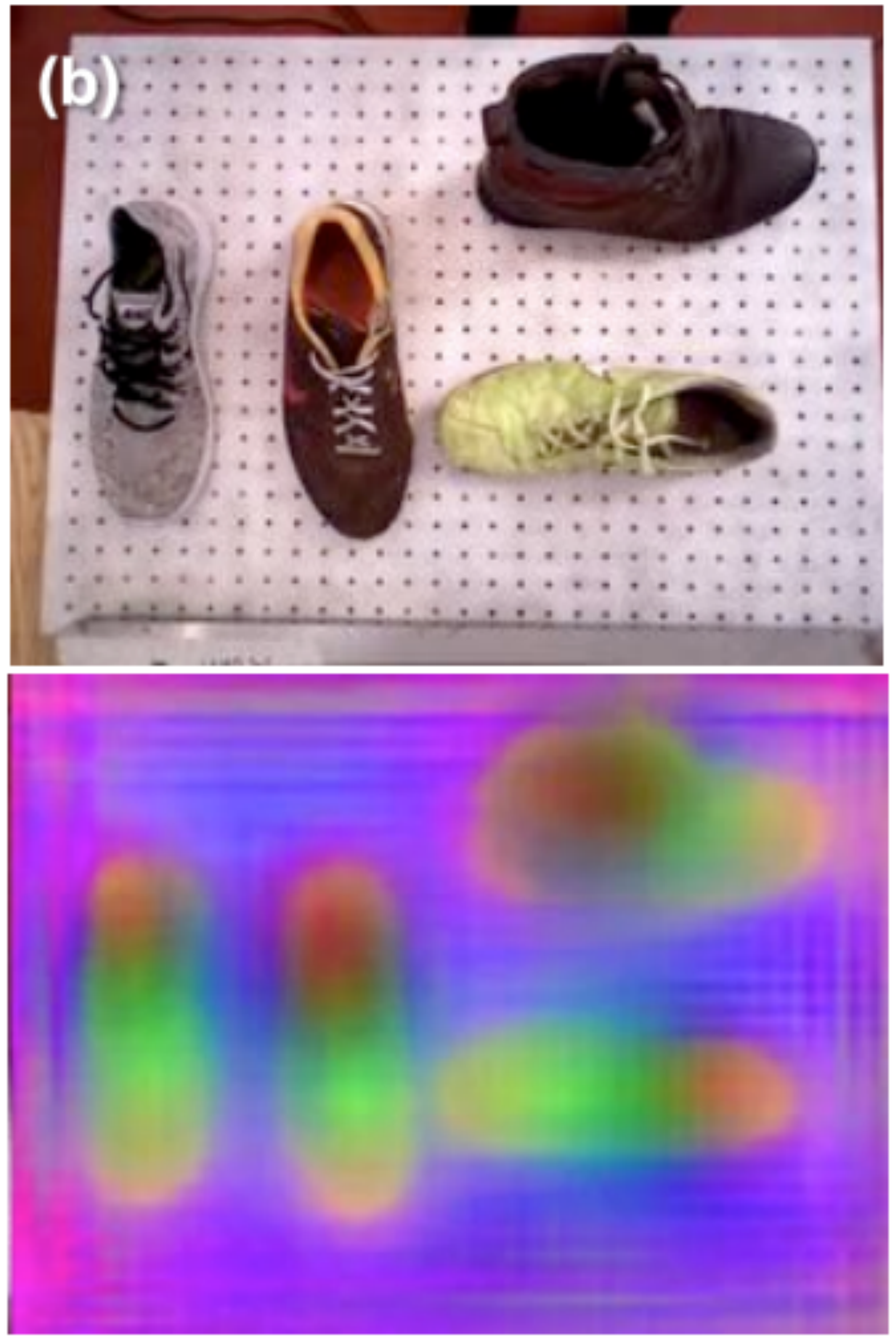
Matching points
(two views of the same piece of the world)

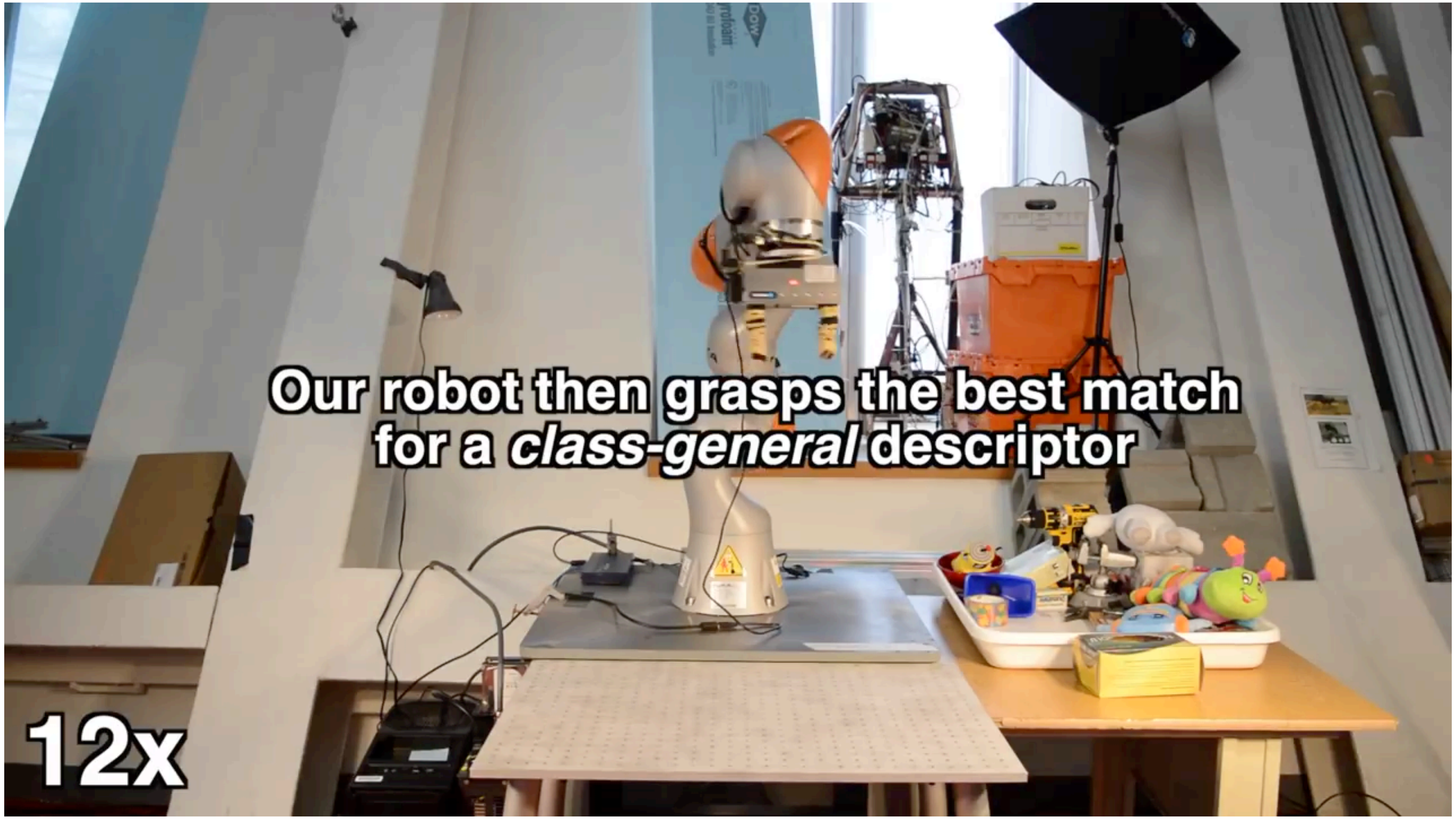
Non-matching points
(two views of the two different pieces of the world)

Learn an embedding such that matching points have the same vector representation and unmatching have different vector representations



Colors capture the *intrinsic geometry* of the object, invariant to transformations





Consistent representation across instances

16. Vision for Embodied Agents

- Formalisms for intelligent agents *(environment, state, action, policy)*
- Imitation learning
- Reinforcement learning
    - Policy gradient method
- Object representations for interaction
    - 3D meshes
    - Dense descriptors