

RECON: Reducing Causal Confusion with Human-Placed Markers

Robert Ramirez Sanchez¹, Heramb Nemlekar¹, Shahabedin Sagheb¹, Cara M. Nunez², and Dylan P. Losey¹

Abstract—Imitation learning enables robots to learn new tasks from human examples. One current fundamental limitation while learning from humans is *causal confusion*. Causal confusion occurs when the robot’s observations include both task-relevant and extraneous information: for instance, a robot’s camera might see not only the intended goal, but also clutter and changes in lighting within its environment. Because the robot does not know which aspects of its observations are important *a priori*, it often misinterprets the human’s examples and fails to learn the desired task. To address this issue, we highlight that — while the robot learner may not know what to focus on — the human teacher does. In this paper we propose that the human proactively marks key parts of their task with small, lightweight *beacons*. Under our framework the human attaches these beacons to task-relevant objects before providing demonstrations: as the human shows examples of the task, beacons track the position of marked objects. We then harness this offline beacon data to train a task-relevant state embedding. Specifically, we embed the robot’s observations to a latent state that is correlated with the measured beacon readings: in practice, this causes the robot to autonomously filter out extraneous observations and make decisions based on features learned from the beacon data. Our simulations and a real robot experiment suggest that this framework for human-placed beacons mitigates causal confusion and enables robots to learn the desired task from fewer demonstrations. See videos here: <https://youtu.be/oy85xJvtLSU>

I. INTRODUCTION

Imagine a human teaching their personal robot arm to pick up bread and drop it on a plate (see Figure 1). To learn this new task, the robot observes its environment and records the human’s demonstrations. But not all the information it observes is relevant: in addition to the bread and plate, the robot’s camera also sees a bowl, assorted objects in the background, and even changes in the lighting conditions. These high-dimensional and complex real-world observations make it challenging for the robot to learn the desired task. For instance, if the position of the bowl changes, should the robot’s actions change? What about the objects in the background or the scene’s lighting — should these features affect the robot’s behavior? Without additional information, the robot does not know *a priori* which aspects of its observations contain task-critical information. Unintentionally focusing on the wrong thing can result in incorrect learning: e.g., moving the bread to a bowl in the background instead of the plate. More generally, imitation learning with real-world observations can produce *causal confusion* [1] and *compounding errors* [2], leading to robots that fail to learn the demonstrated task.

This work was supported by NSF Grants #2337884 and #2246446

¹H. Nemlekar, R. R. Sanchez, S. Sagheb, and D. Losey are with the Collaborative Robotics Lab (Collab), Dept. of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061.

²C. M. Nunez is with the Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14853.

Email: hnemlekar@vt.edu

Existing research often tries to mitigate these issues by developing purely algorithmic solutions. Current methods leverage attention networks, autoencoders, and object-centric priors to try and autonomously extract a task-relevant state from the robot’s high-dimensional observations [3]–[9]. For example, in Figure 1 the state learned by these approaches could include the pixel location of all objects that the robot detects. While today’s algorithmic methods can learn compact representations of the robot’s observations, this process requires data [10]: the human needs to spend more time giving additional examples so that the robot can indirectly infer what parts of the environment are relevant for the task.

In this paper we propose to mitigate causal confusion and efficiently learn task-relevant representations by directly leveraging the human teacher’s knowledge of the key task items. Specifically, we enable the human to add *beacons* (Bluetooth tracking devices) to the environment before giving demonstrations. Beacons are small, lightweight, and inexpensive. As the human demonstrates the task, these beacons automatically record the position of their marked items (e.g., the bread and the plate). Our intuition is that:

Humans will attach beacons to task-relevant objects.

Put another way, we hypothesize that the beacons are correlated with aspects of the observation the robot should focus on. We apply our insight to develop **RECON: Reducing Causal Confusion with Human-Placed Markers**, an imitation learning algorithm that (a) learns a state representation to embed the beacon data, and then (b) learns a policy that maps this compact state representation to robot actions. Returning to our kitchen example: when the human teacher adds beacons to the bread and plate, the robot learns to extract features involving the bread and plate from its camera images, and then take actions based on those features.

Overall, we make the following contributions:

Leveraging Beacons. We extend imitation learning from offline human demonstrations to include user-placed beacons. Under this formalism, beacons provide an added source of supervision for extracting the task-relevant state (i.e., the features) from robot observations.

RECON. We introduce our RECON algorithm for learning a robot policy from human demonstrations. The robot (a) minimizes its divergence from the human’s demonstrated actions and (b) maximizes mutual information between the features and beacon readings. One key advantage of our algorithm is that beacons are not required after training.

Experiments. We conduct multiple simulations and a robot arm experiment in visual imitation learning settings. When the human places the beacon(s) on relevant objects, we find that our RECON algorithm learns the desired task more accurately than prior imitation learning approaches.

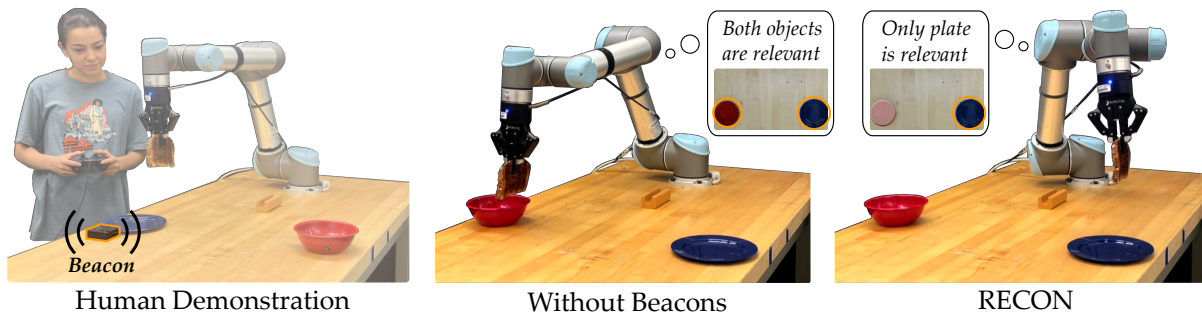


Fig. 1. (Left) Human teaching a robot arm to place bread on a blue plate. The environment also includes a red bowl that is not relevant to the task. The robot observes this scene from a top-down camera. We enable humans to mark task-relevant objects (i.e., the plate) with *beacons* — lightweight devices that track the position of the marked objects while the human is demonstrating the task. (Center) A robot trained to imitate the task with only visual information incorrectly infers that both objects are relevant and delivers the bread to the wrong location. (Right) A robot trained with beacons using our RECON algorithm is able to mitigate this confusion and learns to correctly drop the bread onto the plate, even after the beacons are removed.

II. RELATED WORK

Our research explores imitation learning settings where the human teacher adds markers to the environment to measure the key aspects of their task demonstrations.

Imitation Learning. Within imitation learning, the robot uses examples from a human expert to extrapolate the desired mapping between observations and actions [3]. When the observations are high-dimensional (e.g., images), it is often challenging for the robot to determine which aspects of those observations should affect its actions [1]. Accordingly, recent works that learn from human demonstrations have tried to extract succinct, task-critical representations of visual observations. For instance, object-centric approaches autonomously detect items in the scene (e.g., the bread, the toaster, the bowl), and then condition the robot’s policy on those objects [4]–[6]. More generally, robots can leverage attention networks [7] and representation learning [8] to learn the key features of an image given multiple demonstrations. Overall, these methods attempt to accelerate the robot’s learning by *indirectly* estimating task-critical data from high-dimensional observations. But the robot could learn more efficiently if it had *direct* access to this information (e.g., if the robot knew the bread and plate positions).

Additional Sensors. To more directly measure the relevant components of the state, other imitation learning research explores additional sensors and interfaces [9]. Consider our motivating example of putting bread onto a plate; by tracking where the human looks when teaching the system (e.g., by detecting that the human gazes at the bread and then the plate), the robot can determine which features of the environment it should focus on. More generally, related works have developed multiple types of sensors and interfaces: these include gaze tracking [11], [12], augmented reality [13]–[15], crowdsourced labels [16], and manually specified waypoints [17], [18]. Most relevant to our proposed approach are physical sensors that the human uses or wears when interacting with the environment. In [19], [20] the human utilizes an instrumented grabber to manipulate objects, in [21] the human wears an articulated glove while providing demonstrations, and in [22] the human moves around a position and optical tracker to trace their desired trajectory. These existing sensors focus on the human, and

often must be present during both training and task execution. By contrast, we propose to instrument the *environment* only when collecting demonstrations, and then *remove* those markers when the robot acts autonomously.

III. PROBLEM STATEMENT

We consider contexts where a robot arm is learning a new task from human demonstrations. The robot observes its environment (e.g., using a camera), and the human teacher kinesthetically guides or teleoperates the robot arm through instances of the desired task. The robot learner does not know *a priori* which aspects of the environment it should focus on when learning the task. To accelerate the robot’s learning, we enable the human to instrument the environment by adding beacons to mark key items. Overall, the robot’s objective is to leverage these beacons to efficiently learn a policy that maps its environment observations to actions. Below we formalize each aspect of this problem:

Robot. The robotic system consists of a robot arm and RGB camera. The state of the robot arm is $x \in \mathbb{R}^n$, and the robot takes actions $u \in \mathbb{R}^n$. For instance, within our motivating example, x is the robot’s joint position, and u is the robot’s joint velocity. We emphasize that x only captures information about the robot arm; in order to observe the state of the environment, the robot uses its RGB camera. Let $y \in \mathbb{R}^m$ be the current camera reading (e.g., an image of the toast, bowl, and plate). Overall, at every timestep the robot observes a high-dimensional vector (x, y) and takes action u .

Features. The observed state (x, y) contains the relevant information for learning the task. But this high-dimensional state may also include unnecessary data that should not affect the robot’s behavior. Consider our kitchen example: the camera sees the bowl in image y , but the position of that bowl should have no impact on how the robot picks up and moves the bread. More formally, let $\phi = f(x, y)$ be a minimal representation of the observed state that is sufficient for the desired task. We instantiate $\phi \in \mathbb{R}^k$ as a feature vector of length k , where this feature dimension k is less than the image dimension m . Returning to our example, ϕ could contain the location of the toast and the location of the plate. The robot does not have access to feature vector ϕ ; the robot observes x and y , but it does not know *a priori* which aspects of these observations are features of the desired task.

Beacons. We enable the human teacher to intuitively show the robot which aspects of its observations are important by introducing physical beacons. These beacons are markers: the human can attach them to objects in the environment, and each beacon will continuously relay its real-time position and distance relative to the other beacons. In our experiments, we use beacons with the same technology as Apple AirTags [23]. Specifically, we use Qorvo’s DWM1001-DEV ultra-wideband (UWB) transceiver development boards [24]. These beacons utilize a combination of Bluetooth and UWB to measure position despite of visual occlusions (i.e., the robot does not need to see the beacons). More generally, our algorithmic framework is *not tied to any specific type of beacon*. We simply assume that the beacons provide measurements of the form $b = g(x, y)$, where $b \in \mathbb{R}^d$ consists of information relayed from one or multiple beacons. Vector b depends on the robot state x , the observed environment y , and the locations where the human places the beacon(s).

Policy. The human marks the key objects in the environment at the start of the teaching process and then demonstrates the desired behavior. During these demonstrations the robot records a dataset of observed states, beacon measurements, and expert actions: $\mathcal{D} = \{(x, y, b, u)\}$. After the demonstrations are complete, the human removes the beacons. The robot’s objective is to learn to autonomously perform the demonstrated task. Specifically, the robot should learn a policy $u = \pi(x, y)$ that maps observations to actions. We note that this policy does not depend on beacon readings b — the robot only uses beacons to scaffold its learning.

IV. LEVERAGING BEACONS TO SUPERVISE LEARNING

In this section we present our approach for using demonstrations with human-placed beacons to efficiently bootstrap robot policies. We build upon our underlying hypothesis: beacon readings are correlated with the parts of the observation the robot should focus on. Utilizing this insight, in Section IV-A we first outline a model structure that connects observations, features, beacons, and actions. Next, in Section IV-B we derive loss functions that cause the learned policy to match the human’s demonstrations (minimizing divergence), while also correlating the learned features with the measured beacons (maximizing mutual information). In Section IV-C we finally combine these loss functions to reach RECON, an imitation learning algorithm that reduces causal confusion by biasing the robot’s state representation towards task-relevant features. Once trained, *the robot’s policy can be used without any beacons in the environment*.

A. Model Structure

An outline of our proposed model structure is shown in Figure 2. Below we explain the three component networks:

Policy Network. At training time the robot has access to the offline dataset of task demonstrations $\mathcal{D} = \{(x, y, b, u)\}$. Given this dataset, the robot’s objective is to learn a policy that imitates the human expert and maps observations (x, y) to actions u . We recognize that not all of the information in

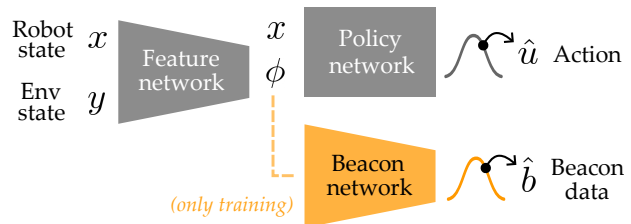


Fig. 2. Our proposed model architecture consists of a *feature network* that maps observations (x, y) to task-relevant features ϕ , a *beacon network* that relates the features to beacon data b , and a *policy network* that estimates actions u based on the robot state and learned features. The beacon network is only utilized at training time to supervise the task-relevant features.

the observed state (x, y) is relevant for the desired task; to correctly imitate the human, the robot only needs to reason over the task-relevant features $\phi = f(x, y)$. Accordingly, we propose to learn a *policy network* of the form: $\pi_{\theta_1}(u | x, \phi)$. This policy is a network with weights θ_1 . In practice, π determines the probability of action u given robot’s current state x and the estimated features ϕ .

Feature Network. The robot learner does not know the features of the task *a priori*. Instead, the robot must learn to extract these features ϕ from its state observations (x, y) . Consistent with our definitions from Section III, we therefore introduce a *feature network* $\phi = f_{\theta_2}(x, y)$ with weights θ_2 . This network inputs observations x and y and deterministically outputs a feature vector ϕ .

Beacon Network. The final component of our model structure connects the features and beacons. Here we apply our insight that the beacon data b should be correlated with the unknown features ϕ . In order to correlate $b = g(x, y)$ and $\phi = f(x, y)$, we add a *beacon network* $h_{\theta_3}(b | \phi)$. This network with weights θ_3 outputs the probability of a beacon measurement $b \in \mathbb{R}^d$ given that the true features are $\phi \in \mathbb{R}^k$. Here the choice of dimension k is important. Although we assume that b is correlated with the task-relevant features, we recognize that the beacon readings may not capture all aspects of the unknown ϕ . Hence, we constrain the designer to select $k > d$, so that the learned vector ϕ can contain additional information beyond the beacon readings b .

B. Loss Functions

We use two sources of supervision to train our model: the expert actions and beacon readings. Similar to behavior cloning, we update the policy and feature networks so that the robot’s learned actions match the actions provided by the human expert [25]. But we also apply the beacon readings to directly supervise feature extraction; because we hypothesize that b and ϕ should be correlated, we can leverage b to guide the robot towards ϕ . Below we derive the loss functions that result from both types of supervision:

Action Supervision. Let $\pi^*(u | x, y)$ be the ideal, unknown policy that the human teacher wants the robot to follow. The robot should learn to match this policy. More concretely, we seek to minimize the Kullback–Leibler (KL) divergence between π^* and π across the demonstration dataset:

$$D_{KL}(\pi^* || \pi) = -\mathbb{E}_{(x, y, u) \sim \mathcal{D}} [\log \pi_{\theta_1}(u | x, \phi)] + C_1 \quad (1)$$

Here $(x, y, u) \in \mathcal{D}$ are observation-action samples from the ideal policy π^* that the human provides during their demonstrations. The constant C_1 in Equation (1) is the entropy of π^* , and this term does not depend on π . Hence, to minimize the KL divergence and match the ideal policy, our model should learn to minimize the following loss function:

$$\mathcal{L}_1(\theta_1, \theta_2) = -\mathbb{E}_{(x,y,b,u) \sim \mathcal{D}} \left[\log \pi_{\theta_1}(u | x, f_{\theta_2}(x, y)) \right] \quad (2)$$

When moving from Equation (1) to Equation (2) we have substituted in our feature network $\phi = f_{\theta_2}(x, y)$. We have also added a marginalization over b — without affecting the loss — because this term is not used within the expectation.

Intuitively, minimizing Equation (2) means that the system has learned a policy π and feature encoding f that causes the robot to mimic the human teacher’s actions across dataset \mathcal{D} . But we do not know how this robot will perform when it encounters new states at run time [26].

Beacon Supervision. To facilitate learning — and better infer what aspects of the state observations the robot should focus on — we introduce a second loss function. This loss is based on the correlation between beacon readings b and learned features ϕ . Specifically, we seek to maximize the mutual information between b and ϕ across dataset \mathcal{D} :

$$I(b; \phi) = \mathbb{E}_{(x,y,b) \sim \mathcal{D}} \left[\log \frac{P(b | \phi)}{P(b)} \right] \quad (3)$$

Increasing mutual information $I(b; \phi)$ means that, if we observe variable b , we gain information about ϕ (and vice versa). We can rewrite this mutual information in terms of our networks. Substituting in the beacon network $P(b | \phi) = h_{\theta_3}(b | \phi)$ and the feature network $\phi = f_{\theta_2}(x, y)$, we reach:

$$I(b; \phi) = \mathbb{E}_{(x,y,b) \sim \mathcal{D}} \left[\log h_{\theta_3}(b | f_{\theta_2}(x, y)) \right] + C_2 \quad (4)$$

Similar to C_1 in Equation (1), here constant C_2 does not depend on our model and can be dropped from the analysis. By remembering to flip the sign (because we want to maximize mutual information gain but minimize loss) and then adding in a marginalization over action u , we obtain our second loss function for supervising the features:

$$\mathcal{L}_2(\theta_2, \theta_3) = -\mathbb{E}_{(x,y,b,u) \sim \mathcal{D}} \left[\log h_{\theta_3}(b | f_{\theta_2}(x, y)) \right] \quad (5)$$

In practice, minimizing Equation (5) causes the robot to learn a feature vector ϕ that can be decoded to recover the beacon signal b . This does not mean that ϕ is equal to b — instead, we obtain a function h that maps from ϕ to b .

C. RECON Algorithm

Now that we have defined the model structure (Section IV-A) and loss functions (Section IV-B), we are ready to introduce our RECON algorithm. RECON is an imitation learning approach that extrapolates from the human’s offline dataset. The human teacher first instruments the environment by adding beacons, and then the human provides task demonstrations \mathcal{D} . The robot trains its policy network π_{θ_1} , feature network f_{θ_2} , and beacon network h_{θ_3} to minimize

the combined loss across the human’s dataset:

$$\mathcal{L}(\theta_1, \theta_2, \theta_3) = -\mathbb{E}_{(x,y,b,u) \sim \mathcal{D}} \left[\log \pi_{\theta_1}(u | x, f_{\theta_2}(x, y)) + \log h_{\theta_3}(b | f_{\theta_2}(x, y)) \right] \quad (6)$$

where Equation (6) is the sum of the action supervision in Equation (2) and the beacon supervision in Equation (5). All three networks that compose our model are trained simultaneously using this loss function. Minimizing Equation (6) trains the model so that (a) we can predict the beacon reading from our learned features ϕ , and (b) we can reconstruct expert actions given x and ϕ . As compared to state-of-the-art imitation learning baselines, the key novelty of RECON is the additional supervision provided by the beacon markers. This supervision guides the robot’s feature extraction such that the learned features are correlated with the human’s marked measurements, resulting in a state representation ϕ that is aligned with the desired task. Returning to our kitchen example: because ϕ is trained to capture the position of the bread and plate from environment images y , the robot learns to make decisions based on these two objects, and ignores the color and location of the extraneous bowl.

Implementation. In our experiments, the feature network f was a convolutional neural network (CNN) that mapped from image y to vector ϕ . Both the beacon network h and the policy π were multilayer perceptrons (MLPs) with fully connected linear layers and rectified linear unit (ReLU) activation functions. The networks were trained using the Adam optimizer at a learning rate of 0.0001. Our code can be found here: <https://github.com/VT-Collab/RECON>

V. SIMULATION EXPERIMENTS

We start by evaluating our proposed algorithm in simulated tasks. Here we test whether robots can learn the desired tasks more accurately from expert demonstrations by leveraging task-relevant beacon information.

Environments. We evaluate our algorithm in three environments: *Static 2D*, *Robosuite*, and *Dynamic 2D*. The *Static 2D* environment consists of three objects placed on a line as shown in Figure 3. The robot’s goal is to reach and grasp the object in the center. In each instantiation of the task, the objects are shuffled and positioned randomly. The *Robosuite* environment has two bins. The left bin contains four objects that are randomly positioned at the start of the task. The robot’s goal is to pick up the right-most object and transfer it to the other bin. In both these environments the objects are static until grasped by the robot. In contrast, the objects in the *Dynamic 2D* environment move around in a circle (see Figure 4-Left). The robot’s goal is to move away from the red object while disregarding the other objects.

A. Imitation Learning with Task-Relevant Beacons

First we assess whether using beacons to mark the relevant objects during training can enable the robot to mitigate causal confusion and learn a better policy from task demonstrations. In these experiments we test our algorithm in the *Static 2D*

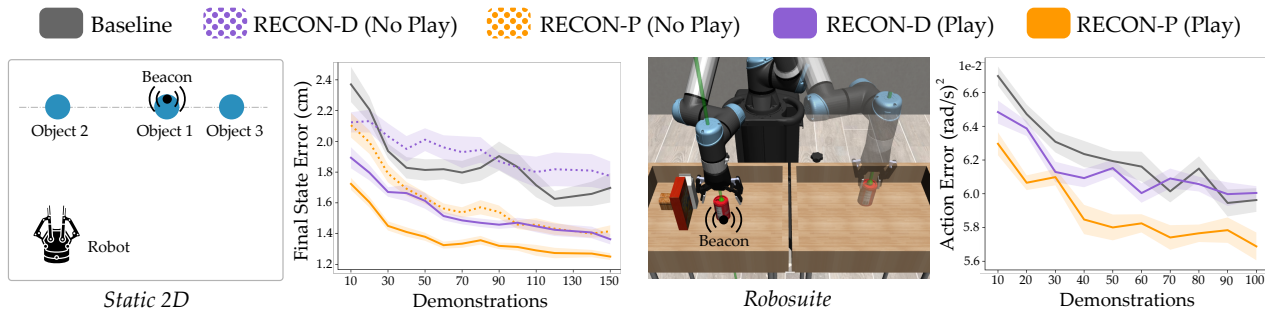


Fig. 3. Environments and results for our simulations in Section V-A averaged over 20 training and testing runs. (Left) In the *Static 2D* environment, the robot must reach the object in the center. Here a robot trained with position beacons (RECON-P) reaches closer to the target than a robot trained without beacons (Baseline). However, when using distance beacons (RECON-D) we require additional Play data (i.e., state and beacon pairs) to properly calibrate the feature network and ensure that the robot performs better than the baseline. (Right) In the *Robosuited* environment, the robot must transfer the right-most object from one bin to the other. Here the robot imitates the expert policy more closely when trained with position beacons and play data (RECON-P (Play)), while training with distance beacons and play data (RECON-D (Play)) produces results similar to the Baseline.

and *Robosuited* environments, and assume that beacons are correctly placed on the relevant objects, i.e., the central object in *Static 2D* and the right-most object in *Robosuited*.

Demonstrations. We simulate the robot’s actions for 10 time steps in *Static 2D* and until task completion in *Robosuited*, after which the environment is reset. At each step, we record the robot state x , the environment state y , the beacon reading b , and the action u . The robot’s state is its 2D position in *Static 2D* and 6D joint angles in *Robosuited*. The environment state includes the positions of all objects.

Play data. Since the objects in our environments are static, the beacon readings stay constant throughout the task. As a result, the beacon data recorded with the demonstrations can be insufficient for training the feature and beacon networks. To account for this, we collect an additional dataset of (y, b) pairs by randomizing the environment and placing beacons on the task-relevant objects. Note that we do not collect extra demonstrations. In practice, play data could be collected once to calibrate the beacons and used for teaching multiple tasks.

Independent variables. Our beacons transmit two types of data: distance and position. While distance requires less instrumentation, position is more informative. We separately evaluate our algorithm with both data types: **RECON-Distance** and **RECON-Position**. We also evaluate training with and without play data: **RECON (Play)** and **RECON (No Play)**. As a baseline, we compare RECON to an imitation learning approach that learns a latent representation of the observations [8] and trains a policy conditioned on the robot state and latent values [27]. Similar to RECON, the **Baseline** includes the policy and feature networks (grey blocks in Figure 2) and trains them by minimizing the loss in Equation (2). However, it does not use any beacon data. We train all methods with the same set of demonstrations.

Results. Our results are summarized in Figure 3. In *Static 2D*, we test the learned policies in 100 new task configurations by rolling out the actions for 10 time steps and measuring the final distance between the robot and the target object. In *Robosuited*, we measure the mean squared error between the actions of an expert policy and the actions of the learned policies in 110 task configurations.

In *Static 2D*, we find that RECON-P achieves a lower

final state error than the Baseline, with and without training on the play data. Meanwhile, RECON-D (No Play) performs similar to the Baseline, only surpassing it after training on the play data. RECON-P performs well even without play data because the mapping from the environment state to beacon position, $y \mapsto \phi \mapsto b$, is straightforward — since y already includes the position of the tagged object. As a result, ϕ can easily learn to b from just the demonstration data. On the other hand, the mapping from y to distance is indirect and requires additional play data to train ϕ effectively.

In *Robosuited*, we find that only RECON-P (Play) is able to outperform the Baseline. Unlike *Static 2D*, where the objects are aligned in a straight line, the objects in this environment are scattered within a bin. Hence, the distance to an object does not always correlate with its position, which explains why the performance does not improve when using beacons that transmit distance instead of position.

Overall these results suggest that when beacon data aligns perfectly with the task-relevant features, our algorithm learns better policies with the same amount of demonstrations compared to learning solely from the environment observations.

B. Learning from Imperfect Beacon Data

In our previous simulations we assumed that the beacons were accurately placed on the task-relevant objects. However, in real-world settings, users can make mistakes, e.g., placing beacons on irrelevant items. Moreover, these beacon readings can be noisy in practice. How does our algorithm perform with imperfect beacon data?

Independent variables. We compare the same **Baseline** to learning from beacons placed on all relevant items (**Exact**), half of the relevant items (**Partial**), other less relevant items (**Other**), and random items (**Random**). For instance, in **Exact** placement the beacon transmits the X-Y position of the task-relevant object (e.g., the red object), while in **Partial**, the beacon only transmits its position along the X or Y axis. In **Other**, the beacon is placed on an object adjacent to the red object and transmits its 2D position. Lastly, in **Random**, the beacon is placed on a randomly chosen object before each demonstration. We also compare with combinations of the above: **Exact+Other** and **Exact+Random**. In addition to evaluating our approach with alternative beacon placements,

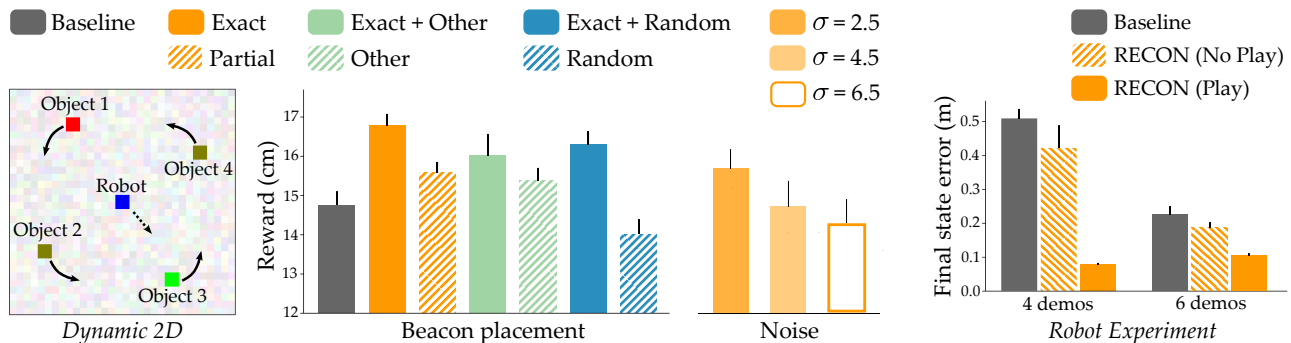


Fig. 4. Results for simulations in the *Dynamic 2D* environment and the real-world robot experiment. (Left) Simulation results for our experiments in Section V-B averaged over 15 training and testing runs. We find that robots trained with beacons attached to the task-relevant objects (Exact) achieve the highest rewards while random beacon placement results in the lowest. Though Partial and Other (less relevant) beacon readings also perform worse than Exact, they still outperform the Baseline (which does not use beacons). Exact remains better than the Baseline even with moderate noise ($\sigma = 2.5$), only performing worse at high noise levels ($\sigma \geq 6.5$). (Right) Results for our robot experiment in Section VI averaged over 3 end-to-end runs. With just 4 demonstrations, a robot trained without beacons (Baseline) fails to recognize the task-relevant objects (blue plate), resulting in poor performance during testing. In contrast, a robot trained with beacon readings (RECON (Play)) learns to focus on the position of the plate and deliver the toast accurately.

we test the performance of our algorithm with noisy beacons: we add a zero mean Gaussian noise with standard deviations $\sigma = [2.5, 4.5, 6.5]$ to the **Exact** beacon readings.

Demonstrations. We collect demonstrations in the Dynamic 2D environment by randomly initializing the objects and simulating the robot’s actions for 10 time steps. As opposed to the static environments where the observations were vectors of object positions, the observations in this environment are RGB images. Because the objects in this environment move at each time step, we obtain diverse beacon readings in the demonstration data itself. Hence, we do not require any play data for calibrating the beacons for this experiment.

Results. Our results are summarized in Figure 4-Left. We train robot policies using RECON for each beacon placement and noise level. All policies are trained with 10 demonstrations. We then test the learned policies in 100 random task configurations by rolling out the actions for 10 time steps and measuring the final distance between the robot and the red object. Since the robot’s goal is to move away from the red object, the final distance corresponds to its **Reward**.

As expected, the rewards decrease with increasing noise in the beacon readings. The robot achieves the highest rewards when its policy is trained with perfect beacon data, i.e., exact placement with zero noise. The performance reduces as we use partially or less relevant beacon data, with the lowest rewards for policies trained using randomly placed beacons. Overall, all beacon placements except Random result in higher rewards than the Baseline, which does not use any beacons. These results indicate that the improvement in performance from training with beacons depends on the degree of correlation between all or part of the beacon readings and the task-relevant features.

VI. ROBOT EXPERIMENT

In this section we evaluate our algorithm in a real-world environment. Specifically, we train a UR5 robot arm to pick up a piece of toast and drop it on a plate. We compare our RECON approach to the same Baseline as in our simulations.

Experimental setup. The environment includes a blue plate, a red bowl, and a piece of toast placed on a table as shown in

Figure 1. The blue plate can lie in two locations: on the left or right edge of the table, while the red bowl is placed anywhere at random. An expert human teacher teleoperates the robot from the toast to the blue plate using a joystick. We collect 2 examples each for 3 different plate and bowl configurations — 6 demonstrations in total. During each demonstration we record the robot joint states x , joint velocities u , images from a top-down camera y , and position readings from a beacon placed near the blue plate b . We also collect play data where we attach the beacon to the plate and record (y, b) pairs for various plate and bowl configurations.

Results. Our results are displayed in Figure 4-Right. We train the robot policy with the collected demonstrations and test by executing the learned policy in the real world for 20 different plate and bowl positions.

After training with only 4 demonstrations, the Baseline learns to associate the robot’s actions with the positions of both objects. As a result, it fails to generalize when the position of the bowl changes during testing. By contrast, RECON learns to ignore the bowl and focus on the position of the plate by leveraging the beacon readings (especially after training with play data). With more demonstrations, the Baseline gradually learns that the bowl is irrelevant to the task, bringing its performance closer to that of RECON (Play). Overall, these results are consistent with the findings from our simulation experiments and highlight the benefit of incorporating human-placed beacons when teaching robots.

VII. CONCLUSION

In this paper we equipped an expert human teacher with position tracking beacons. The human attached these beacons to relevant objects, and we developed an imitation learning approach that synthesized both the human’s demonstrations and the beacon data to learn the desired task. Specifically, we trained the robot to encode its observations (e.g., camera images) into a compact state representation that maximized mutual information with the beacon data. Our simulations and experiment suggest that this approach mitigates causal confusion, helping the robot to efficiently learn the demonstrated task even when the beacons are removed.

REFERENCES

- [1] P. De Haan, D. Jayaraman, and S. Levine, “Causal confusion in imitation learning,” in *Advances in Neural Information Processing Systems*, 2019.
- [2] J. Spencer, S. Choudhury, A. Venkatraman, B. Ziebart, and J. A. Bagnell, “Feedback in imitation learning: The three regimes of covariate shift,” *arXiv preprint arXiv:2102.02872*, 2021.
- [3] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al., “An algorithmic perspective on imitation learning,” *Foundations and Trends in Robotics*, 2018.
- [4] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, “VIOLA: Imitation learning for vision-based manipulation with object proposal priors,” in *Conference on Robot Learning*, 2023.
- [5] J. Gao, Z. Tao, N. Jaquier, and T. Asfour, “K-VIL: Keypoints-based visual imitation learning,” *IEEE Transactions on Robotics*, 2023.
- [6] A. Jonnavittula, S. Parekh, and D. P. Losey, “VIEW: Visual imitation learning with waypoints,” *arXiv preprint arXiv:2404.17906*, 2024.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing*, 2017.
- [8] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto, “The surprising effectiveness of representation learning for visual imitation,” in *Robotics: Science and Systems*, 2021.
- [9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual Review of Control, Robotics, and Autonomous Systems*, 2020.
- [10] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, “State representation learning for control: An overview,” *Neural Networks*, vol. 108, pp. 379–392, 2018.
- [11] A. Biswas, B. A. Pardhi, C. Chuck, J. Holtz, S. Niekum, H. Admoni, and A. Allievi, “Gaze supervision for mitigating causal confusion in driving agents,” in *International Conference on Autonomous Agents and Multiagent Systems*, 2024.
- [12] A. Saran, R. Zhang, E. Short, and S. Niekum, “Efficiently guiding imitation learning algorithms with human gaze,” in *International Conference on Autonomous Agents and Multiagent Systems*, 2021.
- [13] X. Jiang, P. Mattes, X. Jia, N. Schreiber, G. Neumann, and R. Litoutikov, “A comprehensive user study on augmented reality-based data collection interfaces for robot learning,” in *ACM/IEEE International Conference on Human-Robot Interaction*, 2024.
- [14] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft, “Robot programming through augmented trajectories in augmented reality,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.
- [15] M. B. Luebbbers, C. Brooks, C. L. Mueller, D. Szafir, and B. Hayes, “ARC-LFD: Using augmented reality for interactive long-term robot skill maintenance via constrained learning from demonstration,” in *IEEE International Conference on Robotics and Automation*, 2021.
- [16] A. Allevato, E. Short, M. Pryor, and A. Thomaz, “Learning labeled robot affordance models using simulations and crowdsourcing,” in *Robotics: Science and Systems*, 2020.
- [17] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, “Keyframe-based learning from demonstration: Method and evaluation,” *International Journal of Social Robotics*, 2012.
- [18] S. Belkhale, Y. Cui, and D. Sadigh, “Hydra: Hybrid robot actions for imitation learning,” in *Conference on Robot Learning*, 2023.
- [19] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, “Visual imitation made easy,” in *Conference on Robot Learning*, 2021.
- [20] S. Song, A. Zeng, J. Lee, and T. Funkhouser, “Grasping in the wild: Learning 6DoF closed-loop grasping from low-cost demonstrations,” *IEEE Robotics and Automation Letters*, 2020.
- [21] D. Wei and H. Xu, “A wearable robotic hand for hand-over-hand imitation learning,” in *IEEE International Conference on Robotics and Automation*, 2024.
- [22] K.-T. Song, B.-Y. Li, and S.-C. Ou, “Data alignment design for robotic programming by demonstration based on IMU and optical tracker,” *IEEE Transactions on Instrumentation and Measurement*, 2023.
- [23] “AirTag - Apple.” <https://www.apple.com/airtag/>, 2024.
- [24] “DWM1001-DEV Ultra-Wideband (UWB) Transceiver Development Board - Qorvo.” <https://www.qorvo.com/products/p/DWM1001-DEV>, 2024.
- [25] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa, “Expert intervention learning: An online framework for robot learning from explicit and implicit human feedback,” *Autonomous Robots*, 2022.
- [26] S. A. Mehta, Y. U. Ciftci, B. Ramachandran, S. Bansal, and D. P. Losey, “Stable-BC: Controlling covariate shift with stable behavior cloning,” *arXiv preprint arXiv:2408.06246*, 2024.
- [27] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-shot visual imitation learning via meta-learning,” in *Conference on robot learning*, pp. 357–368, 2017.