

Unsupervised Perceptual Rewards for Imitation Learning

Pierre Sermanet* Kelvin Xu*† Sergey Levine

sermanet, kelvinxx, slevine@google.com

Google Brain

Abstract—Reward function design and exploration time are arguably the biggest obstacles to the deployment of reinforcement learning (RL) agents in the real world. In many real-world tasks, designing a reward function takes considerable hand engineering and often requires additional and potentially visible sensors to be installed just to measure whether the task has been executed successfully. Furthermore, many interesting tasks consist of multiple implicit intermediate steps that must be executed in sequence. Even when the final outcome can be measured, it does not necessarily provide feedback on these intermediate steps or sub-goals. To address these issues, we propose leveraging the abstraction power of intermediate visual representations learned by deep models to quickly infer perceptual reward functions from small numbers of demonstrations. We present a method that is able to identify key intermediate steps of a task from only a handful of demonstration sequences, and automatically identify the most discriminative features for identifying these steps. This method makes use of the features in a pre-trained deep model, but does not require any explicit specification of sub-goals. The resulting reward functions, which are dense and smooth, can then be used by an RL agent to learn to perform the task in real-world settings. To evaluate the learned reward functions, we present qualitative results on two real-world tasks and a quantitative evaluation against a human-designed reward function. We also demonstrate that our method can be used to learn a complex real-world door opening skill using a real robot, even when the demonstration used for reward learning is provided by a human using their own hand. To our knowledge, these are the first results showing that complex robotic manipulation skills can be learned directly and without supervised labels from a video of a human performing the task. Supplementary material and dataset are available at sermanet.github.io/rewards

I. INTRODUCTION

Social learning, such as imitation, plays a critical role in allowing humans and animals to acquire complex skills in the real world. Humans can use this weak form of supervision to acquire behaviors from very small numbers of demonstrations, in sharp contrast to deep reinforcement learning (RL) methods, which typically require extensive training data. In this work, we make use of two ideas to develop a scalable and efficient imitation learning method: first, imitation makes use of extensive prior knowledge to quickly glean the “gist” of a new task from even a small number of demonstrations; second, imitation involves both observation and trial-and-error learning (RL). Building on these ideas, we propose a reward learning method for understanding the intent of a user demonstration through the use of pre-trained visual features, which provide

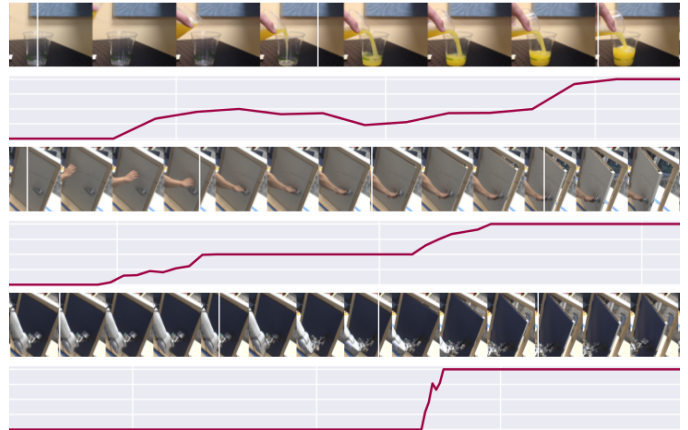


Fig. 1: **Perceptual reward functions learned from unsupervised observation of human demonstrations.** For each video strip, we show its corresponding learned reward function below with the range $[0, 4]$ on the vertical axis, where 4 corresponds to the maximum reward for completing the demonstrated task. We show rewards learned from human demonstrations for a pouring task (top) and door opening task (middle) and use the door opening reward to successfully train a robot to perform the task (bottom).

the “prior knowledge” for efficient imitation. Our algorithm aims to discover not only the high-level goal of a task, but also the implicit sub-goals and steps that comprise more complex behaviors. Extracting such sub-goals can allow the agent to make maximal use of information contained in a demonstration. Once the reward function has been extracted, the agent can use its own experience at the task to determine the physical structure of the behavior, even when the reward is provided by an agent with a substantially different embodiment (e.g. a human providing a demonstration for a robot).

To our knowledge, our method is the first reward learning technique that learns generalizable vision-based reward functions for complex robotic manipulation skills from only a few demonstrations provided directly by a human. Although prior methods have demonstrated reward learning with vision for real-world robotic tasks, they have either required kinesthetic demonstrations with robot state for reward learning [12], or else required low-dimensional state spaces and numerous demonstrations [33]. The contributions of this paper are:

- A method for perceptual reward learning from only a **few demonstrations of real-world tasks**. Reward functions are dense and incremental, with automated **unsupervised**

* equal contribution

† Google Brain Residency program (g.co/brainresidency)

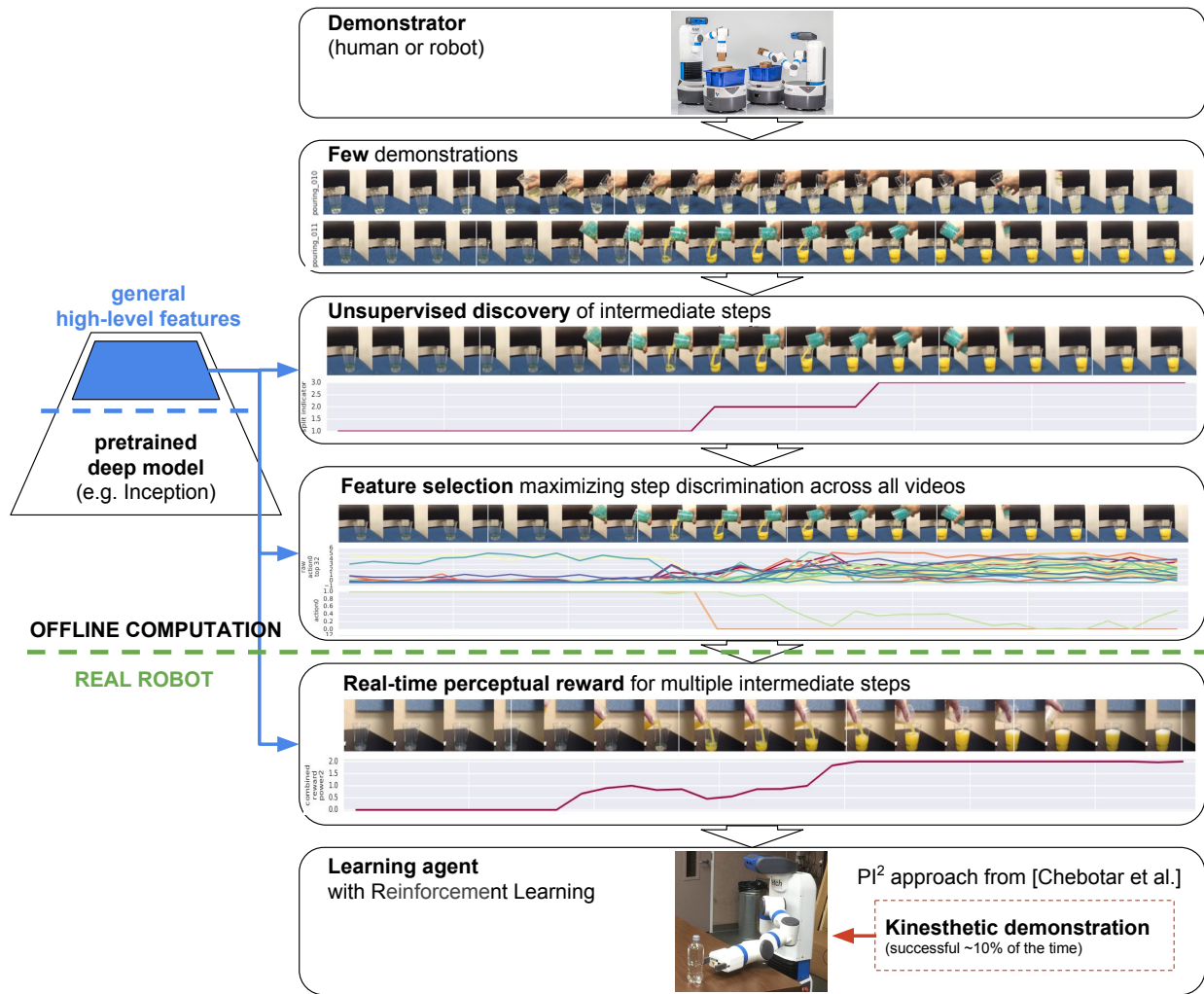


Fig. 2: **Method overview.** Given a few demonstration videos of the same action, our method discovers intermediate steps, then trains a classifier for each step on top of the mid and high-level representations of a pre-trained deep model (in this work, we use all activations starting from the first “mixed” layer that follows the first 5 convolutional layers). The step classifiers are then combined to produce a single reward function per step prior to learning. These intermediate rewards are combined into a single reward function. The reward function is then used by a real robot to learn the perform the demonstrated task as shown in Sec. III-B.

discovery of intermediate steps.

- **The first vision-based reward learning method** that can learn a complex robotic manipulation task from a few human demonstrations in **real-world robotic experiments**.
- A set of empirical experiments that show that the learned visual representations inside a pre-trained deep model are general enough to be directly used to represent goals and sub-goals for manipulation skills in new scenes **without retraining**.

A. Related Work

Our method can be viewed as an instance of learning from demonstration (LfD), which has long been studied as a strategy to overcome the challenging problem of learning robotic tasks from scratch [7] (also see [3] for a survey). Previous work in LfD has successfully replicated gestures [8] and dual-arm manipulation [4] on real robotic systems. LfD can however

be challenging when the human demonstrator and robot do not share the same “embodiment”. A mismatch between the robot’s sensor/actions and the recorded demonstration can lead to suboptimal behaviour. Previous work has addressed this issue by either providing additional feedback [32], or alternatively providing demonstrations which only give a coarse solutions which can be iteratively refined [24]. Our method of using perceptual rewards can be seen as an attempt to circumvent the need for shared embodiment in the general problem of learning from humans. To accomplish this, we first learn a visual reward from human demonstrations, and then learn *how* to perform the task using the robot’s own experience via reinforcement learning.

Our approach can be seen as an instance of the more general inverse reinforcement learning framework [20]. Inverse reinforcement learning can be performed with a variety of algorithms, ranging from margin-based methods [2, 27] to

methods based on probabilistic models [25, 35]. Most similar in spirit to our method is the recently proposed SWIRL algorithm [1] and non-parametric reward segmentation approaches [26, 19, 21], which like our method attempts to decompose the task into sub-tasks to learn a sequence of local rewards. These approaches differs in that they consider only problems with low-dimensional state spaces while our method can be applied to raw pixels. Our experimental evaluation shows that our method can learn a complex real-world door opening skill using visual rewards from a human demonstration.

We use visual features obtained from training a deep convolutional network on a standard image classification task to represent our visual rewards. Deep learning has been used in a range of applications within robotic perception and learning [28, 23, 6]. In work that has specifically examined the problem of learning reward functions from images, one common approach to image-based reward functions has been to directly specify a “target image” by showing the learner the raw pixels of a successful task completion state, and then using distance to that image (or its latent representation) as a reward function [18, 12, 31]. However, this approach has several shortcomings. First, the use of a target image presupposes that the system can achieve a substantially similar visual state, which precludes generalization to semantically similar but visually distinct situations. Second, the use of a target image does not provide the learner with information about which facet of the image is more or less important for task success, which might result in the learner excessively emphasizing irrelevant factors of variation (such as the color of a door due to light and shadow) at the expense of relevant factors (such as whether or not the door is open or closed).

A few recently proposed IRL algorithms have sought to combine IRL with vision and deep network representations [14, 33]. However, scaling IRL to high-dimensional systems and open-ended reward representations is very challenging. The previous work closest to ours used images together with robot state information (joint angles and end effector pose), with demonstrations provided through kinesthetic teaching [14]. The approach we propose in this work, which can be interpreted as a simple and efficient approximation to IRL, can use demonstrations that consist of videos of a human performing the task using their own body, and can acquire reward functions with intermediate sub-goals using just a few examples. This kind of efficient vision-based reward learning from videos of humans has not been demonstrated in prior IRL work. The idea of perceptual reward functions using raw pixels was also explored by [11] which, while sharing the same spirit as this work, was limited to synthetic tasks and used single images as perceptual goals rather than multiple demonstration videos.

II. SIMPLE INVERSE REINFORCEMENT LEARNING WITH VISUAL FEATURES

The key insight in our approach is that we can exploit the semantically meaningful and powerful features in a pre-trained deep neural network to infer task goals and sub-

goals using a very simple approximate inverse reinforcement learning method. The pre-trained network effectively transfers prior knowledge about the visual world to make imitation learning fast and robust. Our approach can be interpreted as a simple approximation to inverse reinforcement learning under a particular choice of system dynamics, as discussed in Section II-A. While this approximation is somewhat simplistic, it affords an efficient and scalable learning rule that minimizes overfitting even when trained on a small number of demonstrations. As depicted in Fig. 2, our algorithm first segments the demonstrations into segments based on perceptual similarity, as described in Section II-B. Intuitively, the resulting segments correspond to sub-goals or steps of the task. The segments can then be used as a supervision signal for learning steps classifiers, described in Section II-C, which produces a single perception reward function for each step of the task. The combined reward function can then be used with a reinforcement learning algorithm to learn the demonstrated behavior. Although this method for extracting reward functions is exceedingly simple, its power comes from the use of highly general and robust pre-trained visual features, and our key empirical result is that such features are sufficient to acquire effective and generalizable reward functions for real-world manipulation skills.

We use the Inception network [29] pre-trained for ImageNet classification [10] to obtain the visual features for representing the learned rewards. It is well known that visual features in such networks are quite general and can be reused for other visual tasks. However, it is less clear if sparse subsets of such features can be used directly to represent goals and sub-goals for real-world manipulation skills. Our experimental evaluation suggests that indeed they can, and that the resulting reward representations are robust and reliable enough for real-world robotic learning without any finetuning of the features. In this work, we use all activations starting from the first “mixed” layer that follows the first 5 convolutional layers (this layer’s activation map is of size 35x35x256 given a 299x299 input). While this paper focuses on visual perception, the approach could be generalized to other modalities (e.g. audio and tactile).

A. Inverse Reinforcement Learning with Time-Independent Gaussian Models

In this work, we use a very simple approximation to the MaxEnt IRL model [35], a popular probabilistic approach to IRL. We will use s_t to denote the visual feature activations at time t , which constitute the state, s_{it} to denote the i^{th} feature at time t , and $\tau = \{s_1, \dots, s_T\}$ to denote a sequence or trajectory of these activations in a video. In MaxEnt IRL, the demonstrated trajectories τ are assumed to be drawn from a Boltzmann distribution according to:

$$p(\tau) = p(s_1, \dots, s_T) = \frac{1}{Z} \exp \left(\sum_{t=1}^T R(s_t) \right), \quad (1)$$

where $R(s_t)$ is the unknown reward function. The principal computational challenge in MaxEnt IRL is to approximate Z , since the states at each time step are not independent, but are constrained by the system dynamics. In deterministic systems, where $s_{t+1} = f(s_t, a_t)$ for actions a_t and dynamics f , the dynamics impose constraints on which trajectories τ are feasible. In stochastic systems, where $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$, we must also account for the dynamics distribution in Equation (1), as discussed by [35]. Prior work has addressed this using dynamic programming to exactly compute Z in small, discrete systems [35], or by using sampling to estimate Z for large, continuous domains [16, 5, 14]. Since our state representation corresponds to a large vector of visual features, exact dynamic programming is infeasible. Sample-based approximation requires running a large number of trials to estimate Z and, as shown in recent work [13], corresponds to a variant of generative adversarial networks (GANs), with all of the accompanying stability and optimization challenges. Furthermore, the corresponding model for the reward function is complex, making it prone to overfitting when only a small number of demonstrations is available.

When faced with a difficult learning problem in extremely low-data regimes, a standard solution is to resort to simple, biased models, so as to minimize overfitting. We adopt precisely this approach in our work: instead of approximating the complex posterior distribution over trajectories under nonlinear dynamics, we use a simple biased model that affords efficient learning and minimizes overfitting. Specifically, we assume that all trajectories are dynamically feasible, and that the distribution over each activation at each time step is independent of all other activations and all other time steps. This corresponds to the IRL equivalent of a naïve Bayes model: in the same way that naïve Bayes uses an independence assumption to mitigate overfitting in high-dimensional feature spaces, we use independence between both time steps and features to learn from very small numbers of demonstrations. Under this assumption, the probability of a trajectory τ factorizes according to

$$p(\tau) = \prod_{t=1}^T \prod_{i=1}^N p(s_{it}) = \prod_{t=1}^T \prod_{i=1}^N \frac{1}{Z_{it}} \exp(R_i(s_{it})),$$

which corresponds to a reward function of the form $R_t(s_t) = \sum_{i=1}^N R_i(s_{it})$. We can then simply choose a form for $R_i(s_{it})$ that can be normalized analytically, such as one which is quadratic in s_{it} which yields a Gaussian distribution. While this approximation is quite drastic, it yields an exceedingly simple learning rule: in the most basic version, we have only to fit the mean and variance of each feature distribution, and then use the log of the resulting Gaussian as the reward.

B. Discovery of Intermediate Steps

The simple IRL model in the previous section can be used to acquire a single quadratic reward function in terms of the visual features s_t . However, for complex multi-stage tasks, this model can be too coarse, making task learning slow and

Algorithm 1 Recursive similarity maximization, where $AverageStd()$ is a function that computes the average standard deviation over a set of frames or over a set of values, $Join()$ is a function that joins values or lists together into a single list, n is the number of splits desired and min_size is the minimum size of a split.

```

function SPLIT(video, start, end, n, min_size, prev_std = [])
  if n = 1 then return [], [AVERAGESTD(video[start : end])]
  end if
  min_std ← None
  min_std_list ← []
  min_split ← []
  for i ← start + min_size to end - ((n - 1) * min_size) do
    std1 ← [AVERAGESTD(video[start : i])]
    splits2, std2 ← SPLIT(video, i, end, n - 1, min_size,
                        std1 + prev_std)
    avg_std ← AVERAGESTD(JOIN(prev_std, std1, std2))
    if min_std = None or avg_std < min_std then
      min_std ← avg_std
      min_std_list ← JOIN(std1, std2)
      min_split ← JOIN(i, splits2)
    end if
  end for
  return min_split, min_std_list
end function

```

difficult. We therefore instead fit multiple quadratic reward functions, with one reward function per intermediate step or goal. These steps are discovered automatically in the first stage of our method, which is performed independently on each demonstration. If multiple demonstrations are available, they are pooled together in the feature selection step discussed in the next section, and could in principle be combined at the segmentation stage as well, though we found this to be unnecessary in our prototype. The intermediate steps model extends the simple independent Gaussian model in the previous section by assuming that

$$p(\tau) = \prod_{t=1}^T \prod_{i=1}^N \frac{1}{Z_{it}} \exp(R_{i_{g_t}}(s_{it})),$$

where g_t is the index of the goal or step corresponding to time step t . Learning then corresponds to identifying the boundaries of the steps in the demonstration, and fitting independent Gaussian feature distributions at each step. Note that this corresponds exactly to segmenting the demonstration such that the variance of each feature within each segment is minimized.

In this work, we employ a simple recursive video segmentation algorithm as described in Algorithm 1. Intuitively, this method breaks down a sequence in a way that each frame in a segment is abstractly similar to each other frame in that segment. The number of segments is treated like a hyperparameter in this approach. There exists a body of unsupervised video segmentation methods [34, 17] which would likely enable a less constrained set of demonstrations to be used. While this is an important avenue of future work, we show that our simple approach is sufficient to demonstrate the efficacy of our method on a realistic set of demonstrations. We also investigate how to reduce the search space of similar feature patterns

across videos in section II-C. This would render discovery of video alignments tractable for an optimization method such as the one used in [15] for video co-localization.

The complexity of Algorithm 1 is $\mathcal{O}(n^m)$ where n is the number of frames in a sequence and m the number of splits. We also experiment with a greedy binary version of this algorithm (Algorithm 2 detailed in Appendix A): first split the entire sequence in two, then recursively split each new segment in two. While not exactly minimizing the variance across all segments, it is significantly more efficient ($\mathcal{O}(n^2 \log m)$) and yields qualitatively sensible results.

C. Steps Classification

In this section we explore learning a steps classifier on top of the pre-trained deep model using a regular linear classifier and a custom feature selection classifier.

One could consider a naive Bayes classifier here, however we show in Fig. 3 that it overfits badly when using all features. Other options are to select a small set of features, or use an alternative classification approach. We first discuss a feature selection approach that is efficient and simple. Then we discuss the alternative logistic regression approach which is simple but less principled. As our experimental results in Table II show, the performance slightly exceeds the selection method, though requires using all of the features.

Intent understanding requires identifying highly discriminative features of a specific goal while remaining invariant to unrelated variation (e.g. lighting, color, viewpoint). The relevant discriminative features may be very diverse and more or less abstract, which motivates our intuition to tap into the activations of deep models at different depths. Deep models cover a large set of representations that can be useful, from spatially dense and simple features in the lower layers (e.g. large collection of detected edges) to gradually more spatially sparse and abstract features (e.g. few object classes).

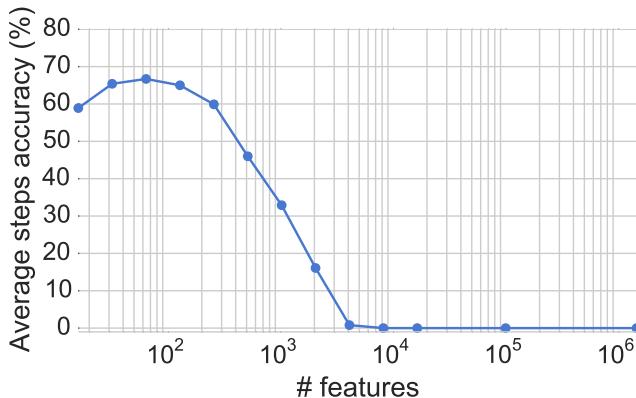


Fig. 3: **Feature selection classification accuracy** on the pouring validation set for 2-steps classification. By varying the number of features n selected, we show that the method yields good results in the region $n = [32, 64]$, but collapses to 0% accuracy starting at $n = 8192$.

Within this high-dimensional representation, we hypothesize that there exists a subset of mid to high-level features that can

readily and compactly discriminate previously unseen inputs. We investigate that hypothesis using a simple feature selection method described in Appendix C. The existence of a small subset of discriminative features can be useful for reducing overfitting in low data regimes, but more importantly can allow drastic reduction of the search space for the unsupervised steps discovery. Indeed since each frame is described by millions of features, finding patterns of feature correlations across videos leads to a combinatorial explosion. However, the problem may become tractable if there exists a low-dimensional subset of features that leads to reasonably accurate steps classification. We test and discuss that hypothesis in Section III-A2.

We also train a simple linear layer which takes as input the same mid to high level activations used for steps discovery and outputs a score for each step. This linear layer is trained using logistic regression and the underlying deep model is not fine-tuned. Despite the large input (1,453,824 units) and the low data regime (11 to 19 videos of 30 to 50 frames each), we show that this model does not severely overfit to the training data and perform slightly better than the feature selection method as described in Section III-A2.

D. Using Perceptual Rewards for Robotic Learning

In order to use our learned perceptual reward functions in a complete skill learning system, we must also choose a reinforcement learning algorithm and a policy representation. While in principle any reinforcement learning algorithm could be suitable for this task, we chose a method that is efficient for evaluation on real-world robotic systems in order to validate our approach. The method we use is based on the PI² reinforcement learning algorithm [30]. Our implementation, which is discussed in more detail in Appendix D, uses a relatively simple linear-Gaussian parameterization of the policy, which corresponds to a sequence of open-loop torque commands with fixed linear feedback to correct for perturbations. This method also requires initialization from example demonstrations to learn complex manipulation tasks efficiently. A more complex neural network policy could also be used [9], and more sophisticated RL algorithms could also learn skills without demonstration initialization. However, since the main purpose of this component is to validate the learned reward functions, we used this simple approach to test our rewards quickly and efficiently.

III. EXPERIMENTS

In this section, we discuss our empirical evaluation, starting with an analysis of the learned reward functions in terms of both qualitative reward structure and quantitative segmentation accuracy. We then present results for a real-world validation of our method on robotic door opening.

A. Perceptual Rewards Evaluation

We report results on two demonstrated tasks: door opening and liquid pouring. We collected about a dozen training videos for each task using a smart phone. As an example, Fig. 4 shows the entire training set used for the pouring task.

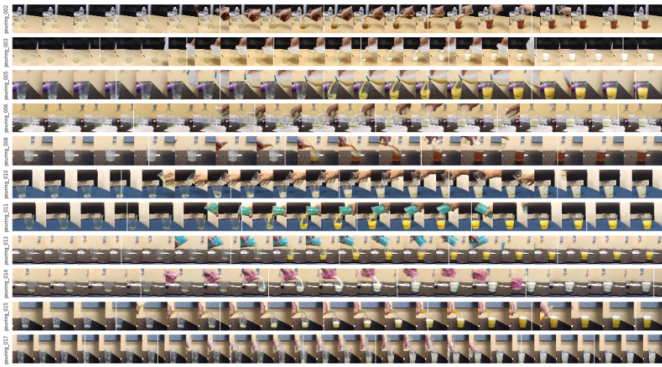


Fig. 4: Entire training set for the pouring task (11 demonstrations).

1) *Qualitative Analysis:* While a door opening sensor can be engineered using sensors hidden in the door, measuring pouring or container tilting would be quite complicated, would visually alter the scene, and is unrealistic for learning in the wild. Visual reward functions are therefore an excellent choice for complex physical phenomena such as liquid pouring. In Fig. 5, we present the combined reward functions for test videos on the pouring task, and Fig. 11 shows the intermediate rewards for each sub-goal. We plot the predicted reward functions for both successful and failed task executions in Fig. 12. We observe that for “missed” executions where the task is only partially performed, the intermediate steps are correctly classified. Fig. 10 details qualitative results of unsupervised step segmentation for the door opening and pouring tasks. For the door task, the 2-segments splits are often quite in line with what one can expect, while a 3-segments split is less accurate. We also observe that the method is robust to the presence or absence of the handle on the door, as well as its opening direction. We find that for the pouring task, the 4-segments split often yields the most sensible break down. It is interesting to note that the 2-segment split usually occurs when the glass is about half full.

Failure Cases

The intermediate reward function for the door opening task which corresponds to a human hand manipulating the door handle seems rather noisy or wrong in Fig. 11b, 11c and 11e (“action1” on the y-axis of the plots). The reward function in Fig. 12f remains flat while liquid is being poured into the glass. The liquid being somewhat transparent, we suspect that it looks too similar to the transparent glass for the function to fire.

2) *Quantitative Analysis:* We evaluate the quantitative accuracy of the unsupervised steps discovery in Table I (see Table III for more details), while Table II presents quantitative generalization results for the learned reward on a test video of each task. For each video, ground truth intermediate steps were provided by human supervision for the purpose of evaluation. While this ground truth is subjective, since each task can be broken down in multiple ways, it is consistent for the simple tasks in our experiments. We use the Jaccard similarity measure (intersection over union) to indicate how much a

detected step overlaps with its corresponding ground truth.

TABLE I: **Unsupervised steps discovery accuracy** (Jaccard overlap on training sets) versus the ordered random steps baseline.

dataset (training)	method	2 steps average	3 steps average
door	ordered random steps	52.5%	55.4%
	unsupervised steps	76.1%	66.9%
pouring	ordered random steps	65.9%	52.9%
	unsupervised steps	91.6%	58.8%

In Table I, we compare our method against a random baseline. Because we assume the same step order in all demonstrations, we also order the random steps in time to provide a fair baseline. Note that the random baseline performs fairly well because the steps are distributed somewhat uniformly in time. Should the steps be much less temporally uniform, the random baseline would be expected to perform very poorly, while our method should maintain similar performance. We compare splitting between 2 and 3 steps and find that, for both tasks, 2 steps are easier to discover, probably because these tasks exhibit one strong visual change each while the other steps are more subtle. Note that our unsupervised segmentation only works when full sequences are available while our learned reward functions can be used in real-time without accessing future frames. Hence in these experiments we evaluate the unsupervised segmentation on the training set only and evaluate the reward functions on the test set.

TABLE II: **Reward functions accuracy** by steps (Jaccard overlap on test sets).

dataset (testing)	classification method	2 steps average	3 steps average
door	random baseline	33.6% ± 1.6	25.5% ± 1.6
	feature selection	72.4% ± 0.0	52.9% ± 0.0
	linear classifier	75.0% ± 5.5	53.6% ± 4.7
pouring	random baseline	31.1% ± 3.4	25.1% ± 0.1
	feature selection	65.4% ± 0.0	40.0% ± 0.0
	linear classifier	69.2% ± 2.0	49.6% ± 8.0

In Table II, we evaluate the reward functions individually for each step on the test set. For that purpose, we binarize the reward function using a threshold of 0.5. The random baseline simply outputs true or false at each timestep. We observe that the learned feature selection and linear classifier functions outperform the baseline by about a factor of 2. It is not clear exactly what the minimum level of accuracy is required to successfully learn to perform these tasks, but we show in section III-B2 that the reward accuracy on the door task is sufficient to reach 100% success rate with a real robot. Individual steps accuracy details can be found in Table IV.

Surprisingly, the linear classifier performs well and does not appear to overfit on our relatively small training set. Although the feature selection algorithm is rather close to the linear classifier compared to the baseline, using feature selection to avoid overfitting is not necessary. However the idea that a small subset of features (32 in this case) can lead to reasonable

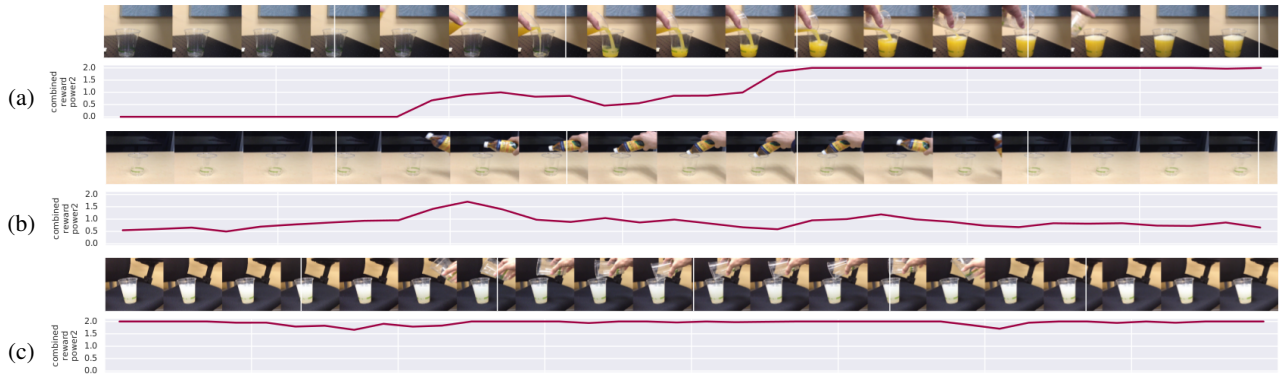


Fig. 5: **Examples of "pouring" reward functions.** We show here a few successful examples, see Fig. 12 for results on the entire test set. In Fig. 5a we observe a continuous and incremental reward as the task progresses and saturating as it is completed. Fig. 5b increases as the bottle appears but successfully detects that the task is not completed, while in Fig. 5c it successfully detects that the action is already completed from the start.

classification accuracy is verified and an important piece of information for drastically reducing the search space for future work in unsupervised steps discovery. Additionally, we show in Fig. 3 that the feature selection approach works well when the number of features n is in the region $[32, 64]$ but collapses to 0% accuracy when $n > 8192$.

B. Real-World Robotic Door Opening

In this section, we aim to answer the question of whether our previously visualized reward function can be used to learn a real-world robotic motion skill. We experiment on a door opening skill, where we adapt a demonstrated door opening to a novel configuration, such as different position or orientation of the door. Following the experimental protocol in prior work [9], we adapt an imperfect kinesthetic demonstration which we ensure succeeds occasionally (about 10% of the time). These demonstrations consist only of robot poses, and do not include images. We then use a variety of different video demonstrations, which contain images but not robot poses, to learn the reward function. These videos include demonstrations with other doors, and even demonstrations provided by a human using their own body, rather than through kinesthetic teaching with the robot. Across all experiments, we use a total of 11 videos. We provide videos of experiments in ¹.

Figure 6 shows the experimental setup. We use a 7-DoF robotic arm with a two-finger gripper, and a camera placed above the shoulder, which provides monocular RGB images. For our baseline PI^2 policy, we closely follow the setup of [9] which uses an IMU sensor in the door handle to provide both a cost and feedback as part of the state of the controller. In contrast, in our approach we remove this sensor both from the state representation provided to PI^2 and in our reward replace the target IMU state with the output of a deep neural network.

1) *Data:* We experiment with a range of different demonstrations from which we derive our reward function, varying both the source demo (human vs robotic), the number of

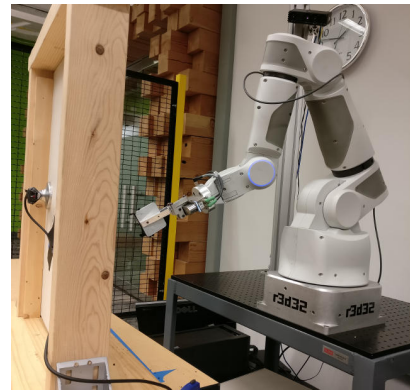


Fig. 6: **Robot arm setup.** Note that our method does not make use of the sensor on the back handle of the door, but it is used in our comparison to train a baseline method with the ground truth reward.

subgoals we extract, and the appearance of the door. We record monocular RGB images on a camera placed above the shoulder of the arm. The door is cropped from the images, and then the resulting image is re-sized such that the shortest side is 299 dimensional with preserved aspect ratio. The input into our convolutional feature extractor [29] is the 299x299 center crop.

2) *Qualitative Analysis:* We evaluate our reward functions qualitatively by plotting our perceptual reward functions below the demonstrations with a variety of door types and demonstrators (e.g robot or human). As can be seen in Fig. 7 and in real experiments Fig. 8, we show that the reward functions are useful to a robotic arm while only showing human demonstrations as depicted in Fig. 13. Moreover we exhibit robustness variations in appearance.

3) *Quantitative Analysis:* In comparing the success rate of visual reward versus a baseline PI^2 method that use s the ground truth reward function obtained by instrumenting the door with an IMU. We run PI^2 for 11 iterations with 10 sampled trajectories at each iteration. As can be seen in Fig. 8, we obtain similar convergence speeds to our baseline

¹sermanet.github.io/rewards

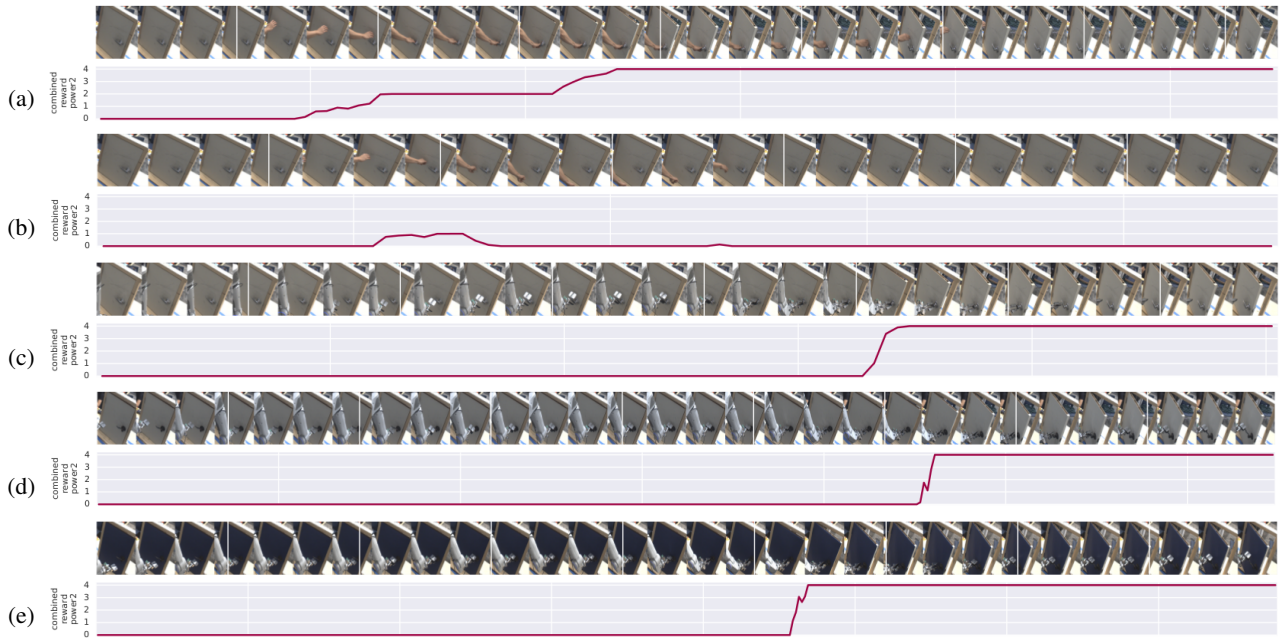


Fig. 7: **Rewards from human demonstration only.** Here we show the rewards produced when trained on humans only (see Fig. 13). In 7a, we show the reward on a human test video. In 7b, we show what the reward produces when the human hands misses opening the door. In 7c, we show the reward successfully saturates when the robot opens the door even though it has not seen a robot arm before. Similarly in 7d and 7e we show it still works with some amount of variation of the door which was not seen during training (white door and black handle, blue door, rotations of the door).

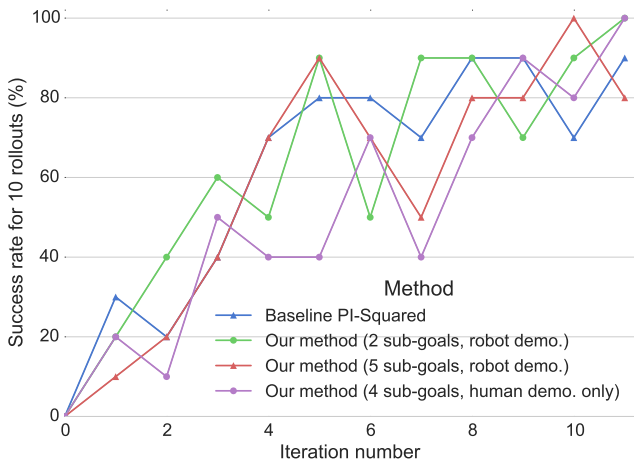


Fig. 8: **Door opening success rate at each iteration of learning on the real robot.** The PI^2 baseline method uses a ground truth reward function obtained by instrumenting the door. Note that rewards learned by our method, even from videos of humans or different doors, learn comparably or faster when compared to the ground truth reward.

model, with our method also able to open the door consistently. Since our local policy is able to obtain high reward candidate trajectories, this is strong evidence that a perceptual reward could be used to train a global policy in same manner as [9].

IV. CONCLUSION

In this paper, we present a method for automatically identifying important intermediate goal given a few visual demonstrations of a task. By leveraging the general features learned from pre-trained deep models, we propose a method for rapidly learning an incremental reward function from human demonstrations which we successfully demonstrate on a real robotic learning task.

We show that pre-trained models are general enough to be used without retraining. We also show there exists a small subset of pre-trained features that are highly discriminative even for previously unseen scenes and which can be used to reduce the search space for future work in unsupervised subgoal discovery.

In this work, we studied imitation in a setting where the viewpoint of the robot/demonstrator is fixed. An interesting avenue of future work would be to analyze the impact of changes in viewpoint. The ability to learn from a learn from broad diverse sets of experience also ties into the goal of lifelong robotic learning. Continuous learning using unsupervised rewards promises to substantially increase the variety and diversity of experiences, resulting in more robust and general robotic skills.

Acknowledgments: We thank Vincent Vanhoucke for helpful discussions and feedback, Mrinal Kalakrishnan and Ali Yahya for indispensable guidance throughout this project and Yevgen Chebotar for his PI^2 implementation. We also thank the anonymous reviewers for their feedback and constructive comments.

REFERENCES

- [1] Swirl: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards.
- [2] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. *ICML*, 2004.
- [3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 2009.
- [4] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 2008.
- [5] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. *AISTATS*, 2011.
- [6] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. *ICRA*, 2017.
- [7] Sylvain Calinon. Robot programming by demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer, 2008.
- [8] Sylvain Calinon and Aude Billard. Incremental learning of gestures by imitation in a humanoid robot. 2007.
- [9] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. *ICRA*, 2017.
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.
- [11] Ashley Edwards, Charles Isbell, and Atsuo Takanishi. Perceptual reward functions. *arXiv preprint arXiv:1608.03824*, 2016.
- [12] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. *arXiv preprint arXiv:1509.06113*, 2015.
- [13] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.
- [14] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *ICML*, 2016.
- [15] A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. *ECCV*, 2014.
- [16] Mrinal Kalakrishnan, Evangelos Theodorou, and Stefan Schaal. Inverse reinforcement learning with pi 2. 2010.
- [17] Oliver Kroemer, Herke Van Hoof, Gerhard Neumann, and Jan Peters. Learning to predict phases of manipulation tasks as hidden states. *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014.
- [18] Sascha Lange, Martin Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. *IJCNN*, 2012.
- [19] Bernard Michini, Mark Cutler, and Jonathan P How. Scalable reward learning from demonstration. *ICRA*, 2013.
- [20] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. *ICML*, 2000.
- [21] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 2015.
- [22] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. *AAAI*, 2010.
- [23] Llerel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *ICRA*, 2016.
- [24] Urbain Prieur, Véronique Perdureau, and Alexandre Bernardino. Modeling and planning high-level in-hand manipulation actions from human knowledge and active learning from demonstration. *IROS*, 2012.
- [25] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.
- [26] Pravesh Ranchod, Benjamin Rosman, and George Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. *IROS*, 2015.
- [27] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. *ICML*, 2006.
- [28] Jaeyong Sung, Seok Hyun Jin, and Ashutosh Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. *arXiv preprint arXiv:1504.03071*, 2015.
- [29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CVPR*, 2016.
- [30] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. 2010.
- [31] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *NIPS*, 2015.
- [32] Sebastian Wrede, Christian Emmerich, Ricarda Grünberg, Arne Nordmann, Agnes Swadzba, and Jochen Steil. A user study on kinesthetic teaching of redundant robots in task and configuration space. *Journal of Human-Robot Interaction*, 2013.
- [33] Markus Wulfmeier, Dominic Zeng Wang, and Ingmar Posner. Watch This: Scalable Cost-Function Learning for Path Planning in Urban Environments .
- [34] Jinhui Yuan, Huiyi Wang, Lan Xiao, Wujie Zheng, Jianmin Li, Fuzong Lin, and Bo Zhang. A formal study of shot boundary detection. *IEEE transactions on circuits and systems for video technology*, 2007.
- [35] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *AAAI*, 2008.

A. Binary Segmentation Algorithm

Algorithm 2 Greedy and binary algorithm similar to and utilizing Algorithm 1, where $AverageStd()$ is a function that computes the average standard deviation over a set of frames or over a set of values, $Join()$ is a function that joins values or lists together into a single list, n is the number of splits desired and min_size is the minimum size of a split.

```

function BINARYSPLIT(video, start, end, n, min_size,
prev_std = [])
  if n = 1 then return [], []
  end if
  splits0, std0 ← SPLIT(video, start, end, 2, min_size)
  if n = 2 then return splits0, std0
  end if
  splits1, std1 ← BINARYSPLIT(video, start, splits0[0],
                               CEIL(n/2), min_size)
  splits2, std2 ← BINARYSPLIT(video, splits0[0]+1, end,
                               FLOOR(n/2), min_size)

  all_splits = []
  all_std = []
  if splits1 ≠ [] then
    JOIN(all_splits, splits1)
    JOIN(all_std, std1)
  else
    JOIN(all_std, std0[0])
  end if
  if splits0 ≠ [] then
    JOIN(all_splits, splits0[0])
  end if
  if splits2 ≠ [] then
    JOIN(all_splits, splits2)
    JOIN(all_std, std2)
  else
    JOIN(all_std, std0[1])
  end if
  return all_splits, all_std
end function

```

B. Combining Intermediate Rewards

From the two previous sections, we obtain one reward function per intermediate step discovered by the unsupervised algorithm. These need to be combined so that the RL algorithm uses a single reward function which partially rewards intermediate steps but emphasizes later rewards. The initial step is ignored as it is assumed to be the initial starting state in the demonstrations. We opt for the maximum range of each reward to be twice the maximum range of its preceding reward, summing them as follow:

$$R(a) = \sum_{i=2}^n R_i(a) * 2^{(i-1)} \quad (2)$$

where n is the number of intermediate rewards detected and a an activations vector. An example of this combination is shown in Fig. 9.

C. Feature Selection Algorithm

Here we describe the feature selection algorithm we use to investigate the presence of a small subset of discriminative features in mid to high level layers of a pre-trained deep network. To select the most discriminative features, we use a simple scoring heuristic. Each feature i is first normalized by subtracting the mean and dividing by the standard deviation of all training sequences. We then rank

them for each sub-goal according to their distance z_i to the average statistics of the sets of positive and negative frames for a given goal:

$$z_i = \alpha \left| \mu_i^+ - \mu_i^- \right| - (\sigma_i^+ + \sigma_i^-), \quad (3)$$

where μ_i^+ and σ_i^+ are the mean and standard deviation of all “positive” frames and the μ_i^- and σ_i^- of all “negative” frames (the frames that do not contain the sub-goal). Only the top- M features are retained to form the reward function $R_g()$ for the sub-goal g , which is given by the log-probability of an independent Gaussian distribution over the relevant features:

$$R_g(s_t) = \frac{1}{n} \sum_j^M \frac{(s_{i_j t} - \mu_{i_j t}^+)^2}{\sigma_{i_j t}^+{}^2}, \quad (4)$$

where i_j indexes the top- M selected features. We empirically choose $\alpha = 5.0$ and $M = 32$ for our subsequent experiments. At test time, we do not know when the system transitions from one goal to another, so instead of time-indexing the goals, we instead combine all of the goals into a single time-invariant reward function, where later steps yield higher reward than earlier steps, as described in Appendix B.

D. PI² Reinforcement Learning Algorithm

We chose the PI² reinforcement learning algorithm [30] for our experiments, with the particular implementation of the method based on a recently proposed deep reinforcement learning variant [9]. Since our aim is mainly to validate that our learned reward functions capture the goals of the task well enough for learning, we employ a relatively simple linear-Gaussian parameterization of the policy, which corresponds to a sequence of open-loop torque commands with fixed linear feedback to correct for perturbations, as in the work of [9]. This policy has the form $\pi(\mathbf{u}_t | \mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t, \Sigma_t)$, where \mathbf{K}_t is a fixed stabilizing feedback matrix, and \mathbf{k}_t is a learned control. In this case, the state \mathbf{x}_t corresponds to the joint angles and angular velocities of a robot, and \mathbf{u}_t corresponds to the joint torques. Since the reward function is evaluated from camera images, we assume that the image is a (potentially stochastic) consequence of the robot’s state, so that we can evaluate the state reward $r(\mathbf{x}_t)$ by taking the image \mathbf{I}_t observed at time t , and computing the corresponding activations a_t . Overloading the notation, we can write $a_t = f(\mathbf{I}_t(\mathbf{x}_t))$, where f is the network we use for visual features. Then, we have $r(\mathbf{x}_t) = R(f(\mathbf{I}_t(\mathbf{x}_t)))$.

The PI² algorithm is an episodic policy improvement algorithm that uses the reward $r(\mathbf{x}_t)$ to iteratively improve the policy. The trust-region variant of PI² that we use [9], which is also similar to the REPS algorithm [22], updates the policy at iteration n by sampling from the time-varying linear-Gaussian policy $\pi(\mathbf{u}_t | \mathbf{x}_t)$ to obtain samples $\{(\mathbf{x}_t^{(i)}, \mathbf{u}_t^{(i)})\}$, and updating the controls \mathbf{k}_t at each time step according to

$$\mathbf{k}_t \leftarrow \left[\sum_i \mathbf{u}_t^{(i)} \exp \left(\beta_t \sum_{t'=t}^T r(\mathbf{x}_{t'}^{(i)}) \right) \right] / \left[\sum_i \exp \left(\beta_t \sum_{t'=t}^T r(\mathbf{x}_{t'}^{(i)}) \right) \right],$$

where the temperature β_t is chosen to bound the KL-divergence between the new policy $\pi(\mathbf{u}_t | \mathbf{x}_t)$ and the previous policy $\bar{\pi}(\mathbf{u}_t | \mathbf{x}_t)$, such that $D_{KL}(\pi(\mathbf{u}_t | \mathbf{x}_t) || \bar{\pi}(\mathbf{u}_t | \mathbf{x}_t)) \leq \epsilon$ for a step size epsilon. Further details and a complete derivation are provided in prior work [30, 22, 9].

The PI² algorithm is a local policy search method that performs best when provided with demonstrations to bootstrap the policy. In our experiments, we use this method together with our learned reward functions to learn a door opening skill with a real physical robot, as discussed in Section III-B. Demonstration are provided with kinesthetic teaching, which results in a sequence of reference steps $\hat{\mathbf{x}}_t$, and initial controls \mathbf{k}_t are given by $\mathbf{k}_t = -\mathbf{K}_t \hat{\mathbf{x}}_t$, such that the mean of the initial controller is $\mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t)$, corresponding to

TABLE III: **Per-step details of unsupervised steps discovery accuracy** (Jaccard overlap on training sets) versus the ordered random steps baseline.

dataset (training)	method	2 steps			3 steps			
		step 1	step 2	average	step 1	step 2	step 3	average
door	ordered random steps	59.4%	45.6%	52.5%	48.0%	58.1%	60.1%	55.4%
	unsupervised steps	84.0%	68.1%	76.1%	57.6%	75.1%	68.1%	66.9%
pouring	ordered random steps	65.2%	66.6%	65.9%	46.2%	46.3%	66.3%	52.9%
	unsupervised steps	92.3%	90.5%	91.6%	79.7%	48.0%	48.6%	58.8%



Fig. 9: **Combining intermediate rewards into a single reward function.** From top to bottom, we show the combined reward function (with range $[0,2]$) followed by the reward function of each individual steps (with range $[0,1]$). The first step reward corresponds to the initial resting state of the demonstration and is ignored in the reward function. The second step corresponds to the pouring action and the third step corresponds to the glass full of liquid.

TABLE IV: **Reward functions accuracy** by steps (Jaccard overlap on test sets).

dataset (testing)	method	steps			
		step 1	step 2	step 3	average
door	random baseline	40.8% \pm 1.0	26.3% \pm 4.1	-	33.6% \pm 1.6
	feature selection	85.1% \pm 0.0	59.7% \pm 0.0	-	72.4% \pm 0.0
	linear classifier	79.7% \pm 6.0	70.4% \pm 5.0	-	75.0% \pm 5.5
door	random baseline	20.8% \pm 1.1	31.8% \pm 1.6	23.8% \pm 2.3	25.5% \pm 1.6
	feature selection	56.9% \pm 0.0	47.7% \pm 0.0	54.1% \pm 0.0	52.9% \pm 0.0
	linear classifier	46%6.9 \pm	47.5% \pm 4.2	67.2% \pm 3.3	53.6% \pm 4.7
pouring	random baseline	39.2% \pm 2.9	22.9% \pm 3.9	-	31.1% \pm 3.4
	feature selection	76.2% \pm 0.0	54.6% \pm 0.0	-	65.4% \pm 0.0
	linear classifier	78.2% \pm 2.4	60.2% \pm 1.7	-	69.2% \pm 2.0
pouring	random baseline	22.5% \pm 0.6	38.8% \pm 0.8	13.9% \pm 0.1	25.1% \pm 0.1
	feature selection	32.9% \pm 0.0	55.2% \pm 0.0	32.2% \pm 0.0	40.0% \pm 0.0
	linear classifier	72.5% \pm 10.5	37.2% \pm 11.0	39.1% \pm 6.8	49.6% \pm 8.0

a trajectory-following initialization. This initial controller is rarely successful consistently, but the occasional successes it achieves provide a learning signal to the algorithm. The use of demonstrations enables PI^2 to be used to quickly and efficiently learn complex robotic manipulation skills.

Although this particular RL algorithm requires demonstrations to begin learning, it can still provide a useful starting point for real-world learning with a real robotic system. As shown by previous work[9], the initial set of demonstrations can be expanded into a generalizable policy by iteratively “growing” the effective region where the policy succeeds. For example, if the robot is provided with a demonstration of opening a door in one position, additional learning can expand the policy to succeed in nearby positions, and the application of a suitable curriculum can grow the region of door poses in which the policy succeeds progressively. However, as with all RL algorithms, this process requires knowledge of the reward function. Using the method described in this paper, we can learn such a reward function from either the initial demonstrations or even from other demonstration videos provided by a human. Armed with this learned reward function, the robot could continue to improve its

policy through real-world experience, iteratively increasing its region of competence through lifelong learning.

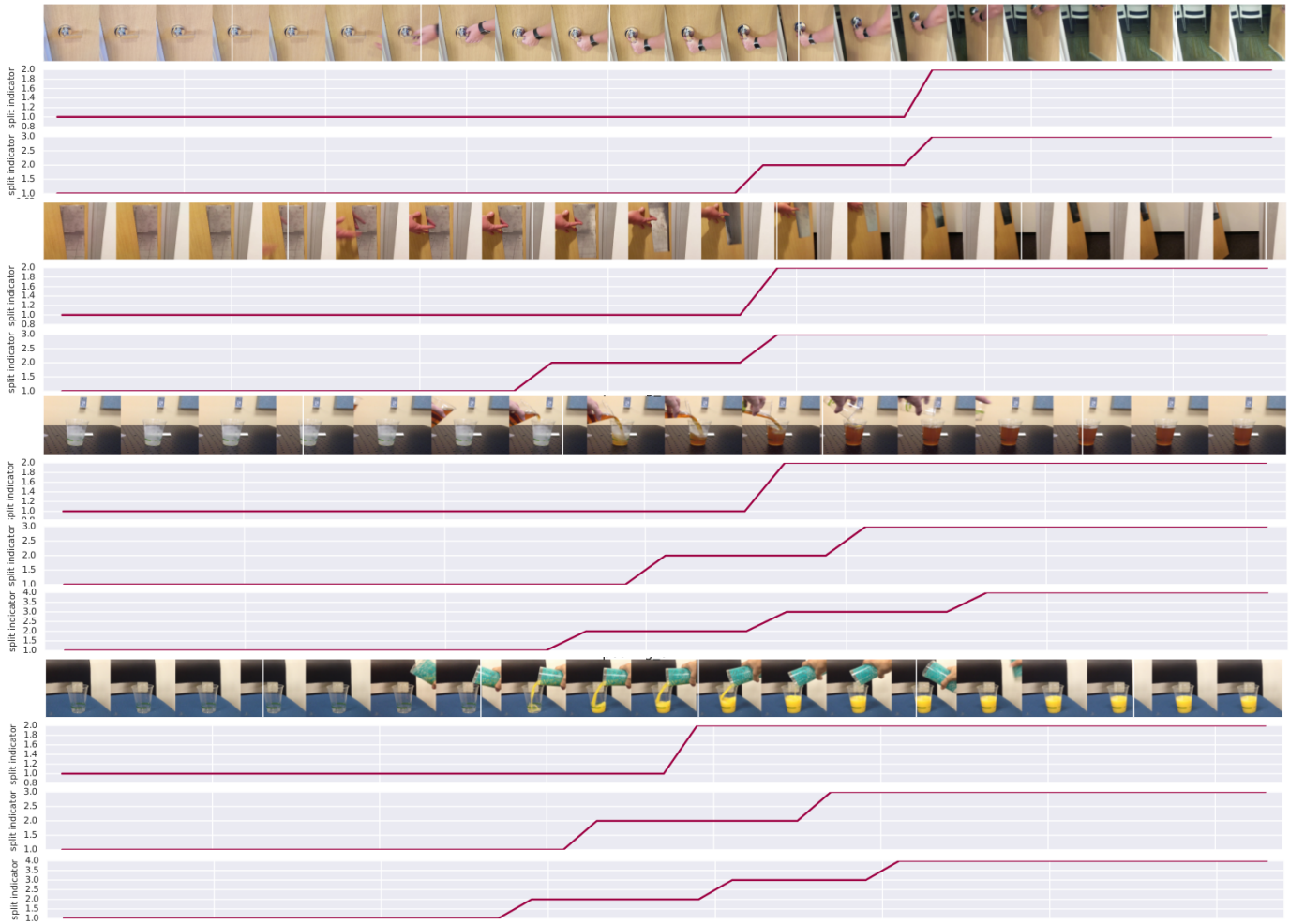


Fig. 10: **Qualitative examples of unsupervised discovery of steps for door and pouring tasks** in training videos. For each video, we show the detected splits when splitting in 2, 3 or 4 segments. Each segment is delimited by a different value on the vertical axis of the curves.

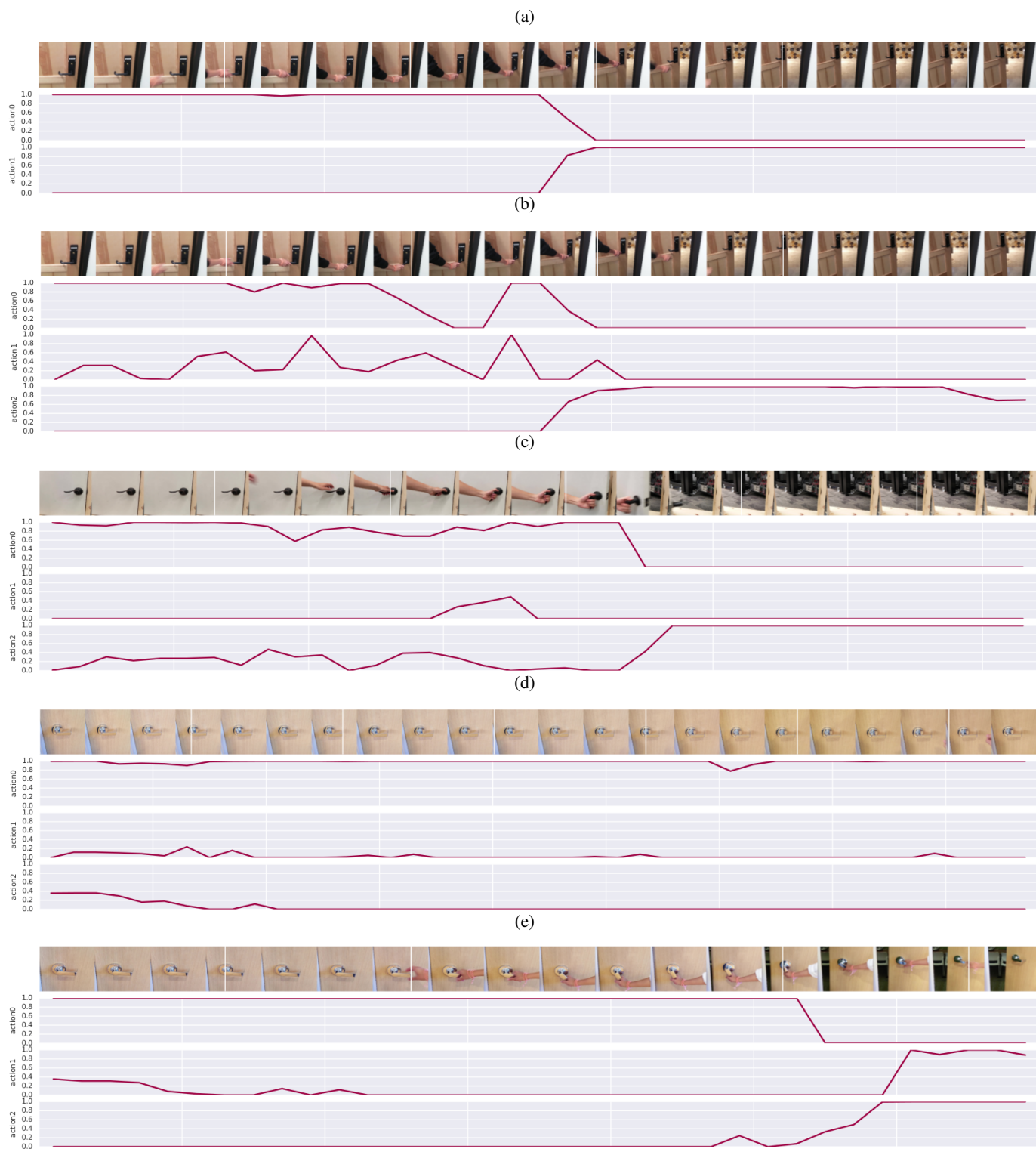


Fig. 11: **Qualitative examples of reward functions for the door task** in testing videos. These plots show the individual sub-goal rewards for either 2 or 3 goals splits. The "open" or "closed" door reward functions are firing quite reliably in all plots, the "hand on handle" step however can be a weaker and noisier signal as seen in 11b and 11c, or incorrect as shown in 11e. 11d demonstrates how a "missed" action is correctly recognized.

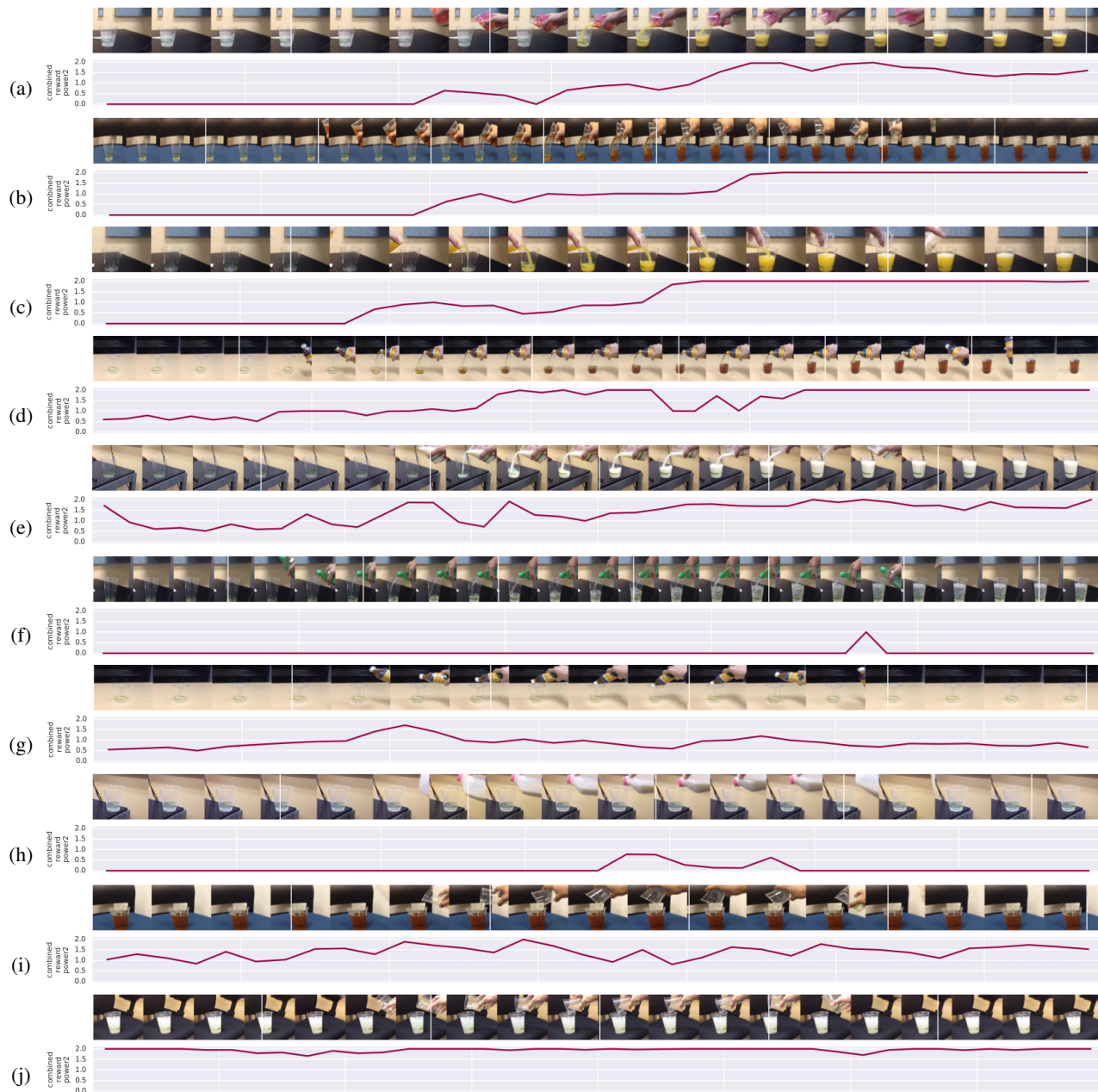


Fig. 12: **Entire testing set of "pouring" reward functions.** This testing set is designed to be more challenging than the training set by including ambiguous cases such as pouring into an already full glass (12i and 12j) or pouring with a closed bottle (12g and 12h). Despite the ambiguous inputs, the reward functions do produce reasonably low or high reward based on how full the glass is. 12a, 12b, 12b and 12d are not strictly monotonically increasing but do overall demonstrate a reasonable progression as the pouring is executed to a saturated maximum reward when the glass is full. 12e also correctly trends upwards but starts with a high reward with an empty glass. 12f is a failure case where the somewhat transparent liquid is not detected.

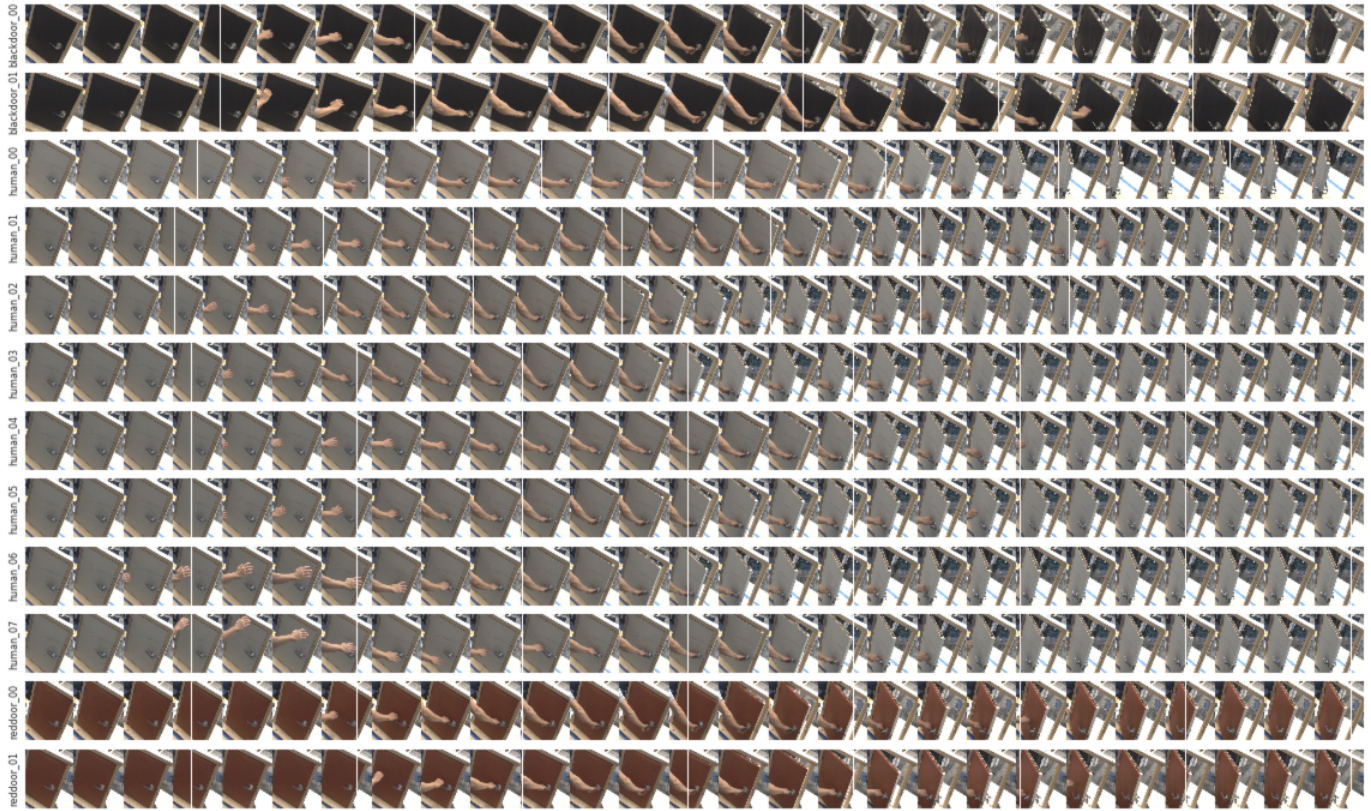


Fig. 13: Entire training set of human demonstrations.