Reward Machine Inference for Robotic Manipulation

Mattijs Baert[1,2], **Elias Malomgré**[1,2], **Sam Leroux**[1,2], **Pieter Simoens**[1,2]
{mattijs.baert,elias.malomgre,sam.leroux,pieter.simoens}@ugent.be

[1]**IDLab, Department of Information Technology at Ghent University**
[2]**imec**

Abstract

Learning from Demonstration (LfD) and Reinforcement Learning (RL) have enabled robot agents to accomplish complex tasks. Reward Machines (RMs) enhance RL's capability to train policies over extended time horizons by structuring high-level task information. In this work, we introduce a novel LfD approach for learning RMs directly from visual demonstrations of robotic manipulation tasks. Unlike previous methods, our approach requires no predefined propositions or prior knowledge of the underlying sparse reward signals. Instead, it jointly learns the RM structure and identifies key high-level events that drive transitions between RM states. We validate our method on vision-based manipulation tasks, showing that the inferred RM accurately captures task structure and enables an RL agent to effectively learn an optimal policy.

1 Introduction

Combining Learning from Demonstration (LfD) with Reinforcement Learning (RL) has empowered artificial agents to learn complex tasks from human examples in areas such as video games Hester et al. (2018), board games Silver et al. (2016), and robotics Argall et al. (2009); Abbeel et al. (2010). LfD enhances RL by improving sample efficiency, which accelerates learning, and by reducing the effort required from human developers to manually specify precise task objectives. However, many LfD approaches struggle with long-duration tasks as they focus on directly learning a control policy from demonstrations Zhang et al. (2018); Mandlekar et al. (2020) or inferring a reward function Ho & Ermon (2016); Abbeel & Ng (2004); Baert et al. (2023). Integrating abstract task structures, by decomposing complex tasks into manageable sub-tasks and providing structured guidance to the agent, has enabled RL to address long-horizon tasks more effectively Baert et al. (2024b); Camacho et al. (2021).

Reward machines (RMs) Icarte et al. (2022) provide a framework for defining such abstract task structures by encoding high-level task objectives in a structured, automata-inspired format. Originally RMs were developed to extend RL for environments with non-Markovian rewards, by enhancing the agent's state space with an abstract state layer that allow agents to retain memory of past actions. However, recent research Camacho et al. (2021) has demonstrated that RMs can also enhance performance in fully Markovian tasks by converting sparse task rewards into a denser reward signal, thereby offering consistent guidance as the policy advances through abstract states. Despite their advantages, most current approaches require RMs to be manually specified by domain experts. In this work, we address this gap by focusing on learning reward machines directly from visual demonstrations in robotic manipulation tasks. The RM framework depends on a set of propositions that encode high-level properties of the environment and a labeling function L that assigns truth values to these propositions based on the current environment state. Unlike prior approaches, which assume these propositions or feature detectors are predefined Toro Icarte et al. (2019); Xu et al. (2021); Verginis et al. (2024); Camacho et al. (2021), our approach jointly learns the RM structure and the mapping from environment states to Boolean propositions. Our method begins by capturing visual demonstrations. We leverage the observation that sub-goals are visited more frequently in

expert demonstrations compared to other states Ghazanfari et al. (2020); Baert et al. (2024a). To identify these sub-goals, each frame is mapped to a low-dimensional feature vector. Through clustering, similar states are grouped and prototypical states (representing a certain sub-goal) can be identified. Finally, an RM is constructed capturing the sequential task structure from the demonstrations. We evaluate our method on a set of vision-based robotic manipulation tasks. Leveraging the inherent interpretability of the RM structure and the ability to visualize sub-goals as prototypical states, we show that our method successfully recovers an RM that accurately captures the demonstrated task. Furthermore, we demonstrate that the learned RM can be effectively used to train an optimal RL policy.

2 Related Work

There has been extensive research on integrating formal methods into RL. For example, several works adopt temporal logics to specify complex, long-horizon tasks Li et al. (2017); Kuo et al. (2020); Xiong et al. (2022); Voloshin et al. (2022), while others investigate the use of automata, such as RMs, to formalize task specifications Araki et al. (2019; 2021); Icarte et al. (2022). A common assumption in these approaches is that high-level knowledge, in the form of automata or temporal logic specifications, is available a priori. However, in real-world scenarios, this knowledge is often implicit and must be inferred from data.

Regarding the inference of temporal logics from data, many methods focus on learning temporal logic formulas using both positive and negative examples Kong et al. (2016); Bombara & Belta (2021). However, in the context of LfD, only positive examples (i.e., demonstrations) are typically available. Some works address learning linear temporal logic (LTL) from only positive examples Shah et al. (2018); Roy et al. (2023), but these approaches usually assume that for each time step in the provided trajectories, the truth valuation of a set of Boolean propositions is known. More recent efforts aim to learn both a mapping from states to atomic propositions and the formula structure Baert et al. (2024a), although applying this in continuous state spaces remains challenging.

In the domain of learning reward machines, most approaches focus on jointly learning both the RM and the policy through environment interaction. For example, discrete optimization can be used to learn an RM that decomposes the task into subproblems, such that combining their optimal memoryless policies yields an optimal solution for the original task Toro Icarte et al. (2019). Xu et al. (2020) propose an iterative method that alternates between automaton inference, used to hypothesize RMs, and RL to optimize policies based on the current RM candidate. Inconsistencies between the hypothesis RM and observed trajectories then trigger re-learning. Verginis et al. (2024) extend this approach to settings with partially known semantics. Additionally, Xu et al. (2021) and Dohmen et al. (2022) leverage active automata inference algorithms, such as the L* algorithm, to learn RMs. Although these approaches do not rely on expert demonstrations, they assume access to the reward function of the underlying MDP as well as the labeling function that maps states to propositions. Closest to our work is the method proposed by Camacho et al. (2021), which learns RMs for vision-based robotic manipulation tasks. However, this approach also assumes access to a predefined mapping between low-level states and high-level propositions, limiting its applicability in more general settings where such a mapping is not readily available.

3 Background

3.1 Reinforcement Learning

A Markov decision process (MDP) Bellman (1957) models sequential decision-making with the following components: a state space $\mathcal{S}$, an action space $\mathcal{A}$, a discount factor $\gamma \in [0, 1]$, a transition distribution $p(s' \mid s, a)$ describing the probability of reaching state s' from state s when taking action a, an initial state distribution $\mathcal{I}(s)$, and a reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, which defines a scalar reward for each state-action pair. An agent interacts with the environment at discrete timesteps t, generating trajectories $\tau = (s_0, ..., s_{T-1})$ of length T. The RL objective is to find an optimal policy

π that maximizes the expected sum of discounted rewards: $\max_\pi \mathbb{E}_\pi \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$.
In this work, we use off-policy deep Q-learning (DQN) Mnih et al. (2015) to train a policy π that selects actions by maximizing a Q-function: $\arg\max_{a \in \mathcal{A}} Q_\theta(s, a)$. Here, $Q_\theta(s, a)$, a neural network with parameters θ, estimates the expected cumulative reward for taking action a in state s. DQN updates θ by minimizing the temporal-difference (TD) error between the current Q-value estimate and a target value y_t. For each training step, transitions (s_t, a_t, r_t, s_{t+1}) are sampled from a replay buffer. The loss function used is the Huber loss Huber (1992):

$$\mathcal{L}(\theta) = \text{Huber}(Q_\theta(s_t, a_t) - y_t) \tag{1}$$

where

$$y_t = r(s_t, a_t, s_{t+1}) + \gamma \max_{a'} Q_\theta(s_{t+1}, a'). \tag{2}$$

Here, a' represents the set of all available actions.

3.2 Reward Machines

A Reward Machine (RM) Icarte et al. (2022) is defined as a tuple $\mathcal{R}_{AP,\mathcal{S},\mathcal{A}} = \langle U, u_0, \delta_u, \delta_r \rangle$, given a set of atomic propositions *AP*, a set of environment states $\mathcal{S}$, and a set of actions $\mathcal{A}$. Each proposition $p \in AP$ has a truth value of either true or false and represents a specific piece of information about the environment, such as object properties, agent statuses, or environmental conditions. The negation of a proposition p is denoted as $\neg p$. Propositions can be combined into more complex logical expressions using conjunctions ($\wedge$) and disjunctions ($\vee$). U is a finite set of states, with $u_0 \in U$ representing the initial state. The state-transition function δ_u maps pairs (u, AP) to new states in U, while the reward-transition function δ_r maps state transitions (u, u') to real-valued rewards in $\mathbb{R}$. At each time step t, the RM receives a truth assignment $\mathbf{p}_t$, which includes the propositions in *AP* that are true in the current environment state s_t. We can replace the standard reward in an MDP by an RM, creating a MDPRM. This requires a labeling function $L : \mathcal{S} \to 2^{|AP|}$ that assigns truth values to the propositions in *AP* based on the environment state. The state of the RM is updated every time step of the environment. If the RM is in state u and the agent takes action a to transition from environment state s to s', the RM transitions to state $u' = \delta_u(u, L(s'))$ and the agent receives a reward $r = \delta_r(u, u')$. A policy $\pi(s, u)$ for an MDPRM is conditioned both on the environment state and the RM state. This setup enables the modeling of non-Markovian reward functions within an MDPRM, as different histories of environment states can be distinguished by elements of a finite set of regular expressions over *AP*. Consequently, RMs can yield different rewards for identical environment transitions (s, a, s'), depending on the agent's prior state history.

4 Reward Machine Inference

In this section, we describe the process of inferring an RM from a set of high-dimensional demonstrations. The method consists of four steps: (1) capturing demonstrations, (2) extracting feature representations, (3) inferring sub-goals through clustering, and (4) constructing the reward machine. An overview of this process, applied to the task of building a predefined pyramid, is depicted in Fig. 1.

4.1 Capturing Demonstrations

The first step is to capture a set of demonstrations from an expert performing the task. A camera with full observability of the workspace records these demonstrations, which results in a set of captured trajectories, denoted as $\mathcal{D} = \{\tau_0, \tau_1, \dots\}$, where each trajectory τ_i represents a sequence of observed states over time. Since an image frame contains all the necessary information for the agent to make decisions, we define a state $s_t \in \mathcal{S}$ as the image frame at time step t. Each state is represented as a tensor in $\mathbb{R}^{256 \times 256 \times 3}$, where the dimensions refer to the height, width, and RGB color channels of the image. Additionally, we capture the same demonstrations using a top-down

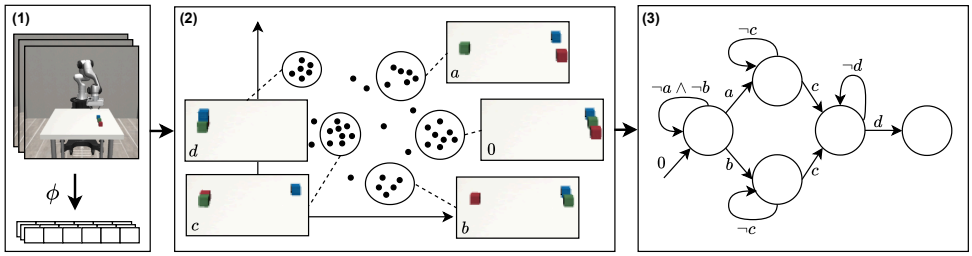


Figure 1: Overview of the proposed method applied to the task of building a predefined pyramid. (1) Visual demonstrations are captured, and feature embeddings are extracted using a pre-trained model ϕ. (2) Sub-goals are inferred by clustering the feature vectors obtained from the demonstrations. (3) An RM is constructed, capturing the valid temporal ordering and transitions between the inferred sub-goals. The 0-prototype corresponds with the initial RM state.

camera, however, these are only utilized during the policy training phase (as described in the next section).

4.2 Extracting Feature Representations

To process the demonstrations, each frame s_t is cropped to retain only the most relevant portion of the workspace, such as the tabletop area where task-relevant objects are located (see Fig. 1). After cropping, we extract a feature representation from each frame. This process transforms the image frames into a lower-dimensional feature space that retains the essential visual information. The extracted feature representation of a state s_t is denoted by $\phi(s_t)$.

4.3 Inferring Sub-Goals through Clustering

To identify sub-goals, we leverage the insight that true sub-goals occur more frequently in expert demonstrations than other states Ghazanfari et al. (2020); Baert et al. (2024a). Consequently, clustering can be used to detect high-density regions in the state space. Sub-goals are inferred by clustering the feature vectors obtained from the previous step, using the DBSCAN algorithm Ester et al. (1996). DBSCAN is robust to noise and does not require prior knowledge of the number of clusters, which is crucial as the number of sub-goals in the task is unknown. Let $\phi(\mathcal{D}) = \phi(s) \mid s \in \tau, \tau \in \mathcal{D}$ represent the set of all feature vectors extracted from all demonstrations. DBSCAN is applied to $\phi(\mathcal{D})$, yielding a set of k clusters $\{C_0, C_1, \ldots, C_{k-1}\}$. The cluster center for each cluster C_i is defined as μ_i:

$$\mu_i = \frac{1}{\mid C_i \mid} \sum_{\phi(s) \in C_i} \phi(s). \tag{3}$$

For each cluster C_i, we identify a prototypical state s_i^*, which is the state whose feature vector is closest to the cluster center:

$$s_i^* = \arg\min_{s \in C_i} \mid\mid \phi(s) - \mu_i \mid\mid_2 . \tag{4}$$

Here, $\mid\mid \cdot \mid\mid_2$ represents the Euclidean norm. Although clustering is performed in feature space, each prototypical state corresponds to an interpretable image, providing a human-understandable representation of each sub-goal.

4.4 Constructing the Reward Machine

The set of prototypical states forms the basis for defining the reward machine states. Let $U = \{u_0, u_1, \ldots, u_{k-1}\}$ represent the set of RM states, where each state u_i corresponds to a prototypical

Algorithm 1 Inferring the state transition function δ_u from the set of abstract demonstrations $\hat{\mathcal{D}}$

Require: set of abstract demonstrations $\hat{\mathcal{D}}$
  $\delta_u(u, \cdot) \leftarrow$ undefined $\forall u \in U$
  for $\hat{d} \in \hat{\mathcal{D}}$ **do**
    $u \leftarrow u_0$
    for $\mathbf{p} \in \hat{d}$ **do**
      for $p_i \in \mathbf{p}$ **do**
        if $\delta_u(u, p_i) =$ undefined **then**
          $\delta_u(\hat{u}, \hat{p}) \leftarrow \begin{cases} u_i & \text{if } \hat{u} = u \text{ and } \hat{p} = p_i \\ \delta_u(\hat{u}, \hat{p}) & \text{otherwise} \end{cases}$
        end if
        $u \leftarrow \delta_u(u, p_i)$
      end for
    end for
  end for
  return δ_u

state s_i^*. We build the set of atomic propositions *AP* by defining a proposition p_i for each prototypical state s_i^*. The truth evaluation of each proposition $p_i \in AP$ is determined based on the Euclidean distance between the feature representation of the current state $\phi(s)$ and the feature representation of the corresponding prototypical state $\phi(s_i^*)$. Formally, for a given proposition $p_i \in AP$, corresponding to the sub-goal represented by prototypical state s_i^* and RM state u_i, the proposition is true if the Euclidean distance between $\phi(s)$ and $\phi(s_i^*)$ is less than a predefined threshold κ. The labeling function can then be expressed as:

$$L(s) = \{p_i \mid ||\phi(s) - \phi(s_i^*)||_2 < \kappa, \forall p_i \in AP\}. \tag{5}$$

An abstract demonstration $\hat{d}$ associated to a demonstration d can be defined as a sequence of sets of propositions $\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_{n-1}$, where $\mathbf{p}_t$ is the abstraction of state s_t into *AP* by L, for each $t \in \{0, \ldots, T-1\}$. The set of abstract demonstrations $\hat{\mathcal{D}}$, thus, defines the directed connections between RM states. Transitions between abstract states observed in the demonstration should be reflected into the state-transition function δ_u. The inference of the state transition function δ_u is formalized in Algorithm 1. Given the definition of the labeling function (Eq. 5), multiple propositions may hold true for a single state. However, this would imply that the RM occupies multiple states simultaneously, which is not feasible. To avoid this, the hyperparameter κ should be tuned so that, for each state s_t, at most one proposition is true.

Next, we need to define the reward transition function δ_r to guide the agent toward reaching the goal states. A naive approach would be to assign a reward of 1 when the agent reaches a goal state in the RM and 0 otherwise. However, this creates a highly sparse reward signal, making it difficult for the agent to learn efficiently. To address this, we use potential-based reward shaping Ng et al. (1999), which helps to construct a denser reward function while maintaining the same optimal behavior as the original sparse reward. The idea is to provide intermediate rewards that encourage progress toward the goal, making training more efficient. Inspired by Camacho et al. (2021), we define the reward function as follows:

$$\Psi(u) = \gamma^{d_{\text{goal}}(u)} \tag{6}$$

$$\delta_r(u, u') = \delta_r'(u, u') + \gamma\Psi(u') - \Psi(u), \tag{7}$$

where Ψ is the potential function, d_{goal} represents the shortest distance from the current RM state u to the goal state in terms of edges and δ_r' is the original sparse reward function. This shaped reward has the property that the reward is negative when the agent moves away from the goal in the RM graph, is slightly less negative when the agent stays in the same state, is zero when the agent moves closer to the goal and evaluates to 1 when the agent reaches the goal state.

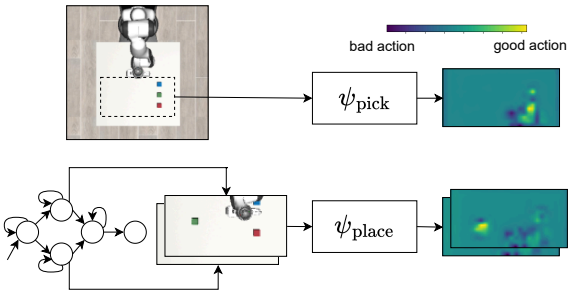


Figure 2: Overview of our DQN formulation. Each action consists of a picking operation at pixel coordinates $(u_{\text{pick}}, v_{\text{pick}})$ followed by a placing operation at $(u_{\text{place}}, v_{\text{place}})$. The Q-function $Q_\theta(s, a)$ is modeled using two FCN's: ψ_{pick} and ψ_{place} which generate pixel-wise Q-value maps for their respective actions.

5 DQN on the Inferred Reward Machine

The action space is defined by a pick-and-place primitive, parameterized by four pixel coordinates: two for the pick location $(u_{\text{pick}}, v_{\text{pick}})$ and two for the place location $(u_{\text{place}}, v_{\text{place}})$. Through camera calibration and depth measurements, each pixel coordinate is mapped accurately to the robot's coordinate system, ensuring precise execution of these actions. For RM inference, images are captured from a front-view camera, while control actions use a top-down view of the workspace. The top-down perspective aligns directly with the workspace plane, reducing distortion and occlusions and facilitating accurate pixel-to-coordinate mappings. In contrast, front-view images are used for inferring sub-goals, as they avoid occlusions caused by the robot and provide a fuller view of the workspace.

Inspired by prior work in robotic manipulation Camacho et al. (2021); Zeng et al. (2022), we define our deep Q-function $Q_\theta(s, a)$ using two fully convolutional networks (FCNs) Long et al. (2015), ψ_{pick} and ψ_{place}. Each FCN receives a top-down image and outputs a dense, pixel-wise map of Q-values for all pick-and-place actions. To stabilize training, we only consider the Q-values inside a rectangular region corresponding to the table area, as picking or placing outside this region is irrelevant. Both FCNs share the same architecture based on ResNet50 He et al. (2015) pre-trained on ImageNet Deng et al. (2009). We use an intermediate feature map from the output of the second residual block, then pass it through two convolutional layers that reduce the channels to one. Finally, the output is bilinearly upsampled to match the input resolution. During training, we compute gradients only for the Q-value of the executed action's pixel location, backpropagating zero loss for all other pixels. All parameters, including those in the pre-trained ResNet layers, are updated.

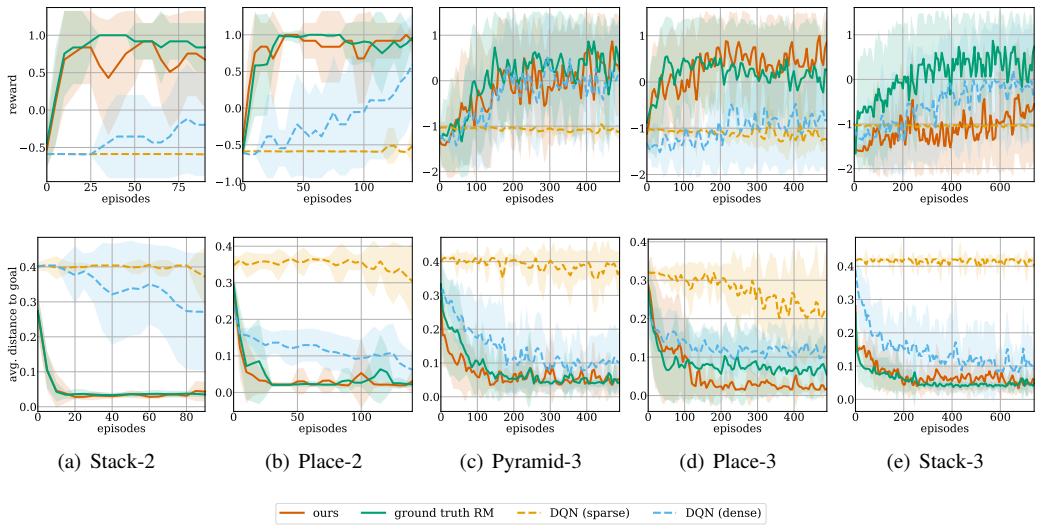
Following the DQRM approach Icarte et al. (2022), we train a discrete Q-function per RM state, each with a separate experience buffer for storing state-specific interactions and demonstrations. At the end of each episode, we update all Q-functions by sampling batches from their corresponding buffers. The input to the pick-action network ψ_{pick} is the current observation, while for ψ_{place}, we pass a top-down view prototype for each valid transition from the current RM state. This allows the network to focus on areas where blocks are expected in the next state, aiding in goal-oriented placement. Fig. 2 depicts an overview of our DQN formulation applied to the pyramid task introduced earlier. Fig. 2 is captured in the beginning of an episode with each block in its initial position and the RM in its initial state. There are two valid transitions from the initial RM state, resulting in the place network ψ_{place} receiving two corresponding top-down prototypes. The Q-value maps in Fig. 2 reveal that the pick network assigns high Q-values to the locations of the red and green blocks, indicating these are favorable pick locations. Similarly, the place network predicts high Q-values at the designated goal locations for each block.

6 Results

We evaluated our method on object manipulation tasks using unstructured human demonstrations within a simulated environment, employing a Franka Emika Panda robot in the robosuite simulation framework Zhu et al. (2020). Five block-based manipulation scenarios were designed: Stack-2, where two blocks are stacked in a predefined order; Place-2, placing two blocks at specific locations; Pyramid-3, building a predefined three-block pyramid; Stack-3 and Place-3, similar to Stack-2 and Place-2, but involving three blocks. The number of demonstrations collected varied with task complexity. For example, Stack-3 required only a single demonstration due to the unique path from the initial to terminal RM state. In contrast, Place-3 required six demonstrations to cover all possible variations in sub-goal sequences. Demonstrations were generated by controlling the robot through an algorithmically defined expert policy, ensuring high-quality demonstrations that fully represent task variations. For RM inference, the feature extractor ϕ is parameterized by a ResNet-50 model He et al. (2015) pre-trained on ImageNet Deng et al. (2009). DQN on the inferred RM was conducted with a fixed batch size of 16, using the Adam optimizer Kingma (2014) and a learning rate of 0.0001. An ϵ-greedy exploration strategy is used, with ϵ initialized at 0.7 and decaying exponentially to 0.1 over training. DBSCAN requires two parameters: $\epsilon_{\text{cluster}}$, which defines the maximum distance between two points for one to be considered in the neighborhood of the other, and *min points*, the minimum number of points needed to establish a dense region. For each scenario, we tuned these parameters along with the threshold κ for matching states with prototypes.

Our method successfully inferred the correct RMs across all tasks, producing meaningful prototype states representing the demonstrated task. The inferred RM accommodates variations in sub-goal sequences observed in the demonstrations by representing each possible ordering as an alternative route through the RM states. To quantitatively assess our approach, we report both total reward per episode and placement error, measured as the average distance between each object and its goal position (Fig. 3). The total reward indicates if an agent effectively learns an optimal policy for the inferred RM. The placement error assesses the accuracy of the inferred RM, particularly in terms of its defined sub-goals. We compared our method to an agent using a ground truth RM with predefined propositions. For this ground truth RM, the set of propositions is again defined such that for each state, one proposition becomes true. Each proposition is defined by the ground truth object positions corresponding to that sub-goal. A proposition becomes true if the placement error between the proposition's object positions and the current object positions is below 0.05m. It is important to note that the ground truth reward machine can only be utilized if the location of each object can be determined at every time step. Fortunately, this condition is met in our simulation environment. To evaluate the impact of the RM, we compare its performance against an agent trained using vanilla DQN. Specifically, we differentiate between an agent that leverages the shaped reward provided by the RM (i.e. dense reward) and an agent that only receives a reward upon successful completion of the entire task (i.e. sparse reward). To track performance during training, we sample a trajectory from the current greedy policy ($\epsilon = 0$) every five episodes, measuring both the total accumulated reward and the placement error.

In achieving the maximum total reward of 1 per episode, agents must consistently transition towards and reach the final goal state in the RM graph (see Eq. 7). For tasks with two blocks higher total reward levels were reached compared to three-block tasks. Both DQRM agents exhibit similar learning curves across most environments. Placement error comparisons also show parity between our inferred RM and the ground truth RM agent, suggesting that any remaining placement error is primarily due to the control algorithm handling the pick-and-place actions. All methods show notable fluctuations in reward and placement error, attributed to noise in executing robot primitives. In our approach, additional variance is introduced by prototype matching, where variations in feature embeddings occasionally prevent RM transitions despite correct block placements. This explains the lower rewards in the Stack-3 task, despite low placement errors, certain RM transitions fail to trigger, preventing the final reward from being obtained even though the task is completed correctly. In the Place-3 task, the inferred RM agent outperformed the ground truth RM agent in both placement error and reward. This is due to the latter failing to recover the optimal policy in two runs. The DQN (dense) agent typically converges to a suboptimal policy, resulting in lower rewards and higher



(a) Stack-2 (b) Place-2 (c) Pyramid-3 (d) Place-3 (e) Stack-3

Figure 3: Total reward obtained during one episode (top) and average distance between each object and its ground goal location, i.e. placement error (bottom) during training. Results are averaged over 10 runs, the shaded region represents the standard deviation.

placement errors. Unlike DQRM, which learns a policy for each RM state, vanilla DQN must learn a single policy for the entire task, making the problem significantly more complex. This difficulty is reflected in both the obtained reward and distance error. Since the DQN (dense) agent adopts the same shaped reward as the DQRM agent, this experiment can be seen as an ablation study on the role of abstract state information in DQRM. Finally, in all cases, the DQN (sparse) agent fails to learn anything due to the large policy search space and the lack of guiding rewards. This supports the claim made by Camacho et al. (2021) that RMs can significantly improve sample efficiency even in Markovian tasks.

7 Conclusion

We presented a novel method for inferring RMs from unstructured demonstrations, using video recordings as input. Unlike existing approaches, our method does not rely on predefined propositions. Instead, it uses clustering to derive meaningful sub-goals represented by prototypical states. By measuring the distance in feature space between the current observation and each prototype, our method detects sub-goal completion and enables state transitions within the RM. Experimental results show that our approach accurately infers the ground-truth RM with interpretable prototypes. Furthermore, the inferred RM enables an RL agent to learn an optimal policy, achieving similar placement accuracy to agents with access to ground-truth object positions and RMs.

Currently, our method converges on a single policy path between the initial and goal states, even when multiple valid paths exist in the RM. A promising direction for future work is to develop agents that can adaptively select alternative paths, which would improve robustness in environments where certain paths become infeasible due to perturbations (e.g., a block becomes unavailable). An interesting body of work in this direction is maximum entropy RL Haarnoja et al. (2018). Additionally, enhancing prototype quality and detection accuracy could be achieved by exploring alternative embedding types, such as object-centric representations Locatello et al. (2020) that are likely more suited to robotic manipulation tasks. This could also facilitate the use of more complex types of objects. Integrating multi-camera views, or projecting views (e.g., from front to top), could further enrich feature representations, especially for real-world scenarios where camera placement may be limited. Finally, exploring the application of this technique in other domains, such as navigation, could extend its impact beyond manipulation tasks.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

Brandon Araki, Kiran Vodrahalli, Thomas Leech, Cristian-Ioan Vasile, Mark D Donahue, and Daniela L Rus. Learning to plan with logical automata. 2019.

Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan DeCastro, Micah Fry, and Daniela Rus. The logical options framework. In *International Conference on Machine Learning*, pp. 307–317. PMLR, 2021.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Mattijs Baert, Pietro Mazzaglia, Sam Leroux, and Pieter Simoens. Maximum causal entropy inverse constrained reinforcement learning. *arXiv preprint arXiv:2305.02857*, 2023.

Mattijs Baert, Sam Leroux, and Pieter Simoens. Learning temporal task specifications from demonstrations. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems*, pp. 81–98. Springer, 2024a.

Mattijs Baert, Sam Leroux, and Pieter Simoens. Learning task specifications from demonstrations as probabilistic automata. *arXiv preprint arXiv:2409.07091*, 2024b.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Giuseppe Bombara and Calin Belta. Offline and online learning of signal temporal logic formulae using decision trees. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–23, 2021.

Alberto Camacho, Jacob Varley, Andy Zeng, Deepali Jain, Atil Iscen, and Dmitry Kalashnikov. Reward machines for vision-based robotic manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14284–14290. IEEE, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Taylor Dohmen, Noah Topper, George Atia, Andre Beckus, Ashutosh Trivedi, and Alvaro Velasquez. Inferring probabilistic reward machines from non-markovian reward signals for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 574–582, 2022.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pp. 226–231, 1996.

Behzad Ghazanfari, Fatemeh Afghah, and Matthew E Taylor. Sequential association rule mining for autonomously extracting hierarchical task structures in reinforcement learning. *IEEE Access*, 8: 11782–11799, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Zhaodan Kong, Austin Jones, and Calin Belta. Temporal logics for learning and detection of anomalous behavior. *IEEE Transactions on Automatic Control*, 62(3):1210–1222, 2016.

Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Encoding formulas as deep networks: Reinforcement learning for zero-shot execution of ltl formulas. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5604–5610. IEEE, 2020.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. GTI: learning to generalize across long-horizon tasks from human demonstrations. In Marc Toussaint, Antonio Bicchi, and Tucker Hermans (eds.), *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*, 2020. DOI: 10.15607/RSS.2020.XVI.061. URL https://doi.org/10.15607/RSS.2020.XVI.061.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287, 1999.

Rajarshi Roy, Jean-Raphaël Gaglione, Nasim Baharisangari, Daniel Neider, Zhe Xu, and Ufuk Topcu. Learning interpretable temporal properties from positive examples only. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6507–6515, 2023.

Ankit Shah, Pritish Kamath, Julie A Shah, and Shen Li. Bayesian inference of temporal task specifications from demonstrations. *Advances in Neural Information Processing Systems*, 31, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Rodrigo Toro Icarte, Ethan Waldie, Toryn Klassen, Rick Valenzano, Margarita Castro, and Sheila McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Christos K Verginis, Cevahir Koprulu, Sandeep Chinchali, and Ufuk Topcu. Joint learning of reward machines and policies in environments with partially known semantics. *Artificial Intelligence*, pp. 104146, 2024.

Cameron Voloshin, Hoang Le, Swarat Chaudhuri, and Yisong Yue. Policy optimization with linear temporal logic constraints. *Advances in Neural Information Processing Systems*, 35:17690–17702, 2022.

Zikang Xiong, Joe Eappen, Ahmed H Qureshi, and Suresh Jagannathan. Constrained hierarchical deep reinforcement learning with differentiable formal specifications. 2022.

Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 590–598, 2020.

Zhe Xu, Bo Wu, Aditya Ojha, Daniel Neider, and Ufuk Topcu. Active finite reward automaton inference and reinforcement learning using queries and counterexamples. In *Machine Learning and Knowledge Extraction: 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17–20, 2021, Proceedings 5*, pp. 115–135. Springer, 2021.

Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research*, 41(7):690–705, 2022.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 1–8. IEEE, 2018. DOI: 10.1109/ICRA.2018.8461249. URL https://doi.org/10.1109/ICRA.2018.8461249.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

8 Additional Hyperparameter Values

DBSCAN requires two parameters: $\epsilon_{\text{cluster}}$, which defines the maximum distance between two points for one to be considered in the neighborhood of the other, and *min points*, the minimum number of points needed to establish a dense region. For each scenario, we tuned these parameters along with the threshold κ for matching states with prototypes. Table 1 provides an overview of the tuned parameters.

	κ	$\epsilon_{\text{cluster}}$	*min points*	# demonstrations
Stack-2	3.0	0.8	15	1
Place-2	1.9	1.0	20	2
Pyramid-3	2.3	1.2	30	2
Stack-3	3.0	1.8	40	1
Place-3	1.9	1.7	110	6

Table 1: Tuned parameters and the number of demonstrations used for the different scenarios.

9 Additional Qualitative Results

As presented in the main text, each RM state corresponds to a unique high-level proposition that triggers a transition. The (0)-prototype represents the initial RM state, with each subsequent prototype corresponding to a specific task sub-goal. Fig. 4 and Fig. 5 depict the inferred RM and prototypes for the Place-3 and the Stack-3 scenarios respectively.

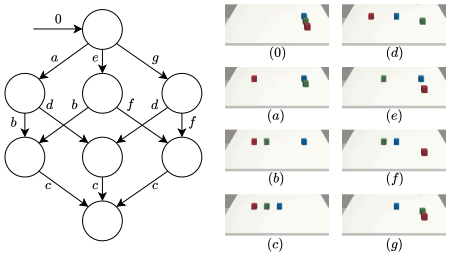


Figure 4: Inferred RM for the Place-3 task (left) and the corresponding state prototypes (right). We omitted the self-loop transitions for clarity.

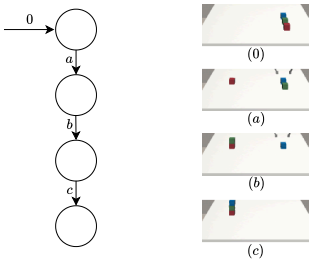


Figure 5: Inferred RM for the Stack-3 task (left) and the corresponding state prototypes (right). We omitted the self-loop transitions for clarity.