Subtask-Aware Visual Reward Learning from Segmented Demonstrations

Anonymous Author(s)
Affiliation
Address
email

Abstract: Reinforcement Learning (RL) agents have demonstrated their potential across various robotic tasks. However, they still heavily rely on human-engineered reward functions, requiring extensive trial-and-error and access to target behavior information, often unavailable in real-world settings. This paper introduces REDS: *REward learning from Demonstration with Segmentations*, a novel reward learning framework that leverages action-free videos with minimal supervision. Specifically, REDS employs video demonstrations segmented into subtasks from diverse sources and treats these segments as ground-truth rewards. We train a dense reward function conditioned on video segments and their corresponding subtasks to ensure alignment with ground-truth reward signals by minimizing the Equivalent-Policy Invariant Comparison distance. Additionally, we employ contrastive learning objectives to align video representations with subtasks, ensuring precise subtask inference during online interactions. Our experiments show that REDS significantly outperforms baseline methods on complex robotic manipulation tasks in Meta-World and more challenging real-world tasks, such as furniture assembly in FurnitureBench, with minimal human intervention.

1 Introduction

Reinforcement Learning (RL) has demonstrated significant potential for training autonomous agents in various real-world robotic tasks, provided that appropriate reward functions are available [1, 2, 3, 4, 5]. However, reward engineering typically requires substantial trial-and-error [6, 7] and extensive task knowledge, often necessitating specialized instrumentation (*e.g.*, motion trackers [8] or tactile sensors [9]) or detailed information about target objects [10, 11, 12, 13, 14, 15], which are difficult to obtain in real-world settings. Learning reward functions from action-free videos has emerged as a promising alternative, as it avoids the need for detailed action annotations or precise target behavior information, and video data can be easily collected from online sources [16, 17, 18]. Approaches in this domain include learning discriminators between video demonstrations and policy rollouts [19, 20], training temporally aligned visual representations from large-scale video datasets [21, 22, 23, 24, 25] to estimate reward based on distance to a goal image, and using video prediction models to generate reward signals [26, 27].

Despite this progress, existing methods often struggle with long-horizon, complex robotic tasks that involve multiple subtasks. These approaches typically fail to provide context-aware reward signals, relying only on a few consecutive frames or the final goal image without considering subsequent subtasks. For example, in One Leg task (see Figure 2d) from FurnitureBench [28], prior methods often overemphasize the reward for picking up the leg while neglecting crucial steps such as inserting the leg into a hole and tightening it. Recent work [29] proposes a discriminator-based approach that treats complex tasks as a sequence of subtasks. However, it assumes that the environment provides explicit subtask identification, which often demands significant human intervention in real-world scenarios. Moreover, discriminator-based methods are known to be prone to mode collapse [30, 31]. Consequently, designing an effective visual reward function for real-world, long-horizon tasks remains an open problem.

Submitted to the 8th Conference on Robot Learning (CoRL 2024). Do not distribute.

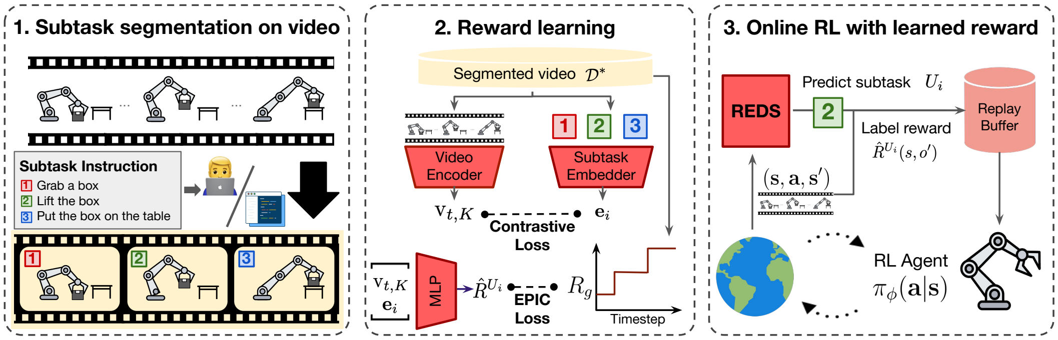


Figure 1: Illustration of REDS. Our main idea is to leverage expert demonstrations annotated with the ongoing subtask as the source of implicit reward signals (left). We train a reward model conditioned on video segments and corresponding subtasks with 1) contrastive loss to attract the video segments and corresponding subtask embeddings and 2) EPIC [32] loss to generate reward equivalent to subtask segmentations (middle). In online RL, REDS infers the ongoing subtask using only video segments at each timestep and computes the reward with that (right).

Our approach To address the aforementioned limitations, we propose a novel reward learning framework, REDS: *REward learning from Demonstration with Segmentations*, which infers subtask information from video segments and generates corresponding reward signals for each subtask. The key idea is to employ minimal supervision to produce appropriate reward signals for intermediate subtask completion. Specifically, REDS utilizes expert demonstrations, where subtasks are annotated at each timestep by various sources (e.g., human annotators, code snippets, vision-language models; see the left figure of Figure 1). These annotations serve as ground-truth rewards. For training, we introduce a new objective function minimizing the Equivalent-Policy Invariant Comparison (EPIC) [32] between the learned reward function and the ground-truth rewards, guaranteeing a theoretical upper bound on regret relative to the ground-truth reward function. Additionally, to correctly infer the ongoing subtask in online interactions, we adopt a contrastive learning objective to align video representations with task embeddings. In terms of architecture, our reward model is designed to capture temporal dependencies in video segments using transformers [33], leading to enhanced reward signal quality.

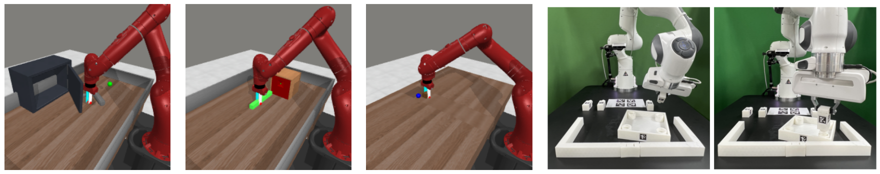
Contributions We present a novel visual reward learning framework REDS: *REward learning from Demonstration with Segmentations*, which can produce suitable reward signals aware of subtasks in long-horizon complex robotic manipulation tasks. We show that REDS significantly outperforms baselines in training RL agents for robotic manipulation tasks in Meta-world, and even surpasses dense reward functions in some tasks. Furthermore, we demonstrate that REDS can train real-world RL agents to perform long-horizon complex furniture assembly tasks from FurnitureBench.

2 REward learning from Demonstration with Segmentations

2.1 Intuition

The sparse reward function R provides feedback only on the overall success or failure of a task, which is insufficient for guiding the agent through intermediate states. To address this, drawing inspiration from previous work on long-range robotic manipulation tasks [28, 29], we decompose the task into m subtasks, denoted as $\mathcal{U} = \{U_1, ..., U_m\}$, using domain knowledge. Each subtask U_i represents a distinct step in the task sequence, where i indicates its order. For task success, the agent must complete these subtasks in sequence, meaning subtask U_i must be finished before moving to U_{i+1}.[1] To further guide the agent, we provide text instructions $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^m$, describing how to solve each subtask. To obtain subtask segmentations, we map each observation o_t in the trajectory

[1]For instance, Door Open can be divided into (i) reaching the door handle and (ii) pulling the door to the target position.



(a) Door Open (b) Peg Insert Side (c) Sweep Into (d) One Leg

Figure 2: Examples of visual observations used in our experiments. We consider a variety of robotic manipulation tasks from Meta-world [12] and FurnitureBench [28].

$\tau = (o_0, ..., o_T)$ to its corresponding subtask using a segmentation function $\psi : \mathcal{O} \rightarrow \mathcal{U}$ from various sources.

2.2 Reward Modeling

Architecture As mentioned in Section 1, previous reward learning methods generate rewards only by a single frame or consequent frames, not taking into account the order of subtasks. To resolve the issue, we propose a new reward predictor $\hat{R}^U = \hat{R}(s; U)$ conditioned on each subtask.

Reward equivariance with subtask segmentation Our key insight is that the subtask segmentation function ψ can be thought of as the ground-truth reward function, providing implicit signals for solving intermediate tasks. To ensure our reward function induces the same set of optimal policies as ψ, we train to minimize EPIC [32] distance between our reward model $\hat{R}^U_\theta$ parameterized by θ and ψ for all subtasks.

Progressive reward signal However, minimizing EPIC with ψ alone can lead to overfitting and the inability to provide progressive signals within each subtask. To mitigate this issue, we propose an additional regularization term to enforce progressive reward signals. Inspired by previous work [34, 35, 36], we view the reward function as a progress indicator for each subtask, and we regularize the reward function output to be higher in later states of expert demonstration.

Aligning video representation with subtask embeddings As the reward model lacks information about the ongoing subtasks in online interactions, it must infer the agent's current subtask. To achieve this, we train the video representation to be closely aligned with the corresponding subtask embedding by adopting a contrastive learning objective. The model can select the appropriate subtask embedding only by the video segment.

3 Experiments

3.1 Meta-world Experiments

Setup We first evaluate our method on 8 different visual robotic manipulation tasks from Meta-world [12]. As a backbone algorithm, we use DreamerV3 [37], a state-of-the-art visual model-based RL algorithm that learns from latent imaginary rollouts. For collecting subtask segmentations, we utilize a scripted teacher in simulation environments for scalability. Specifically, we use the predefined indicator for subtasks provided in the benchmark for all subtask segmentations (see Appendix E for the list of subtasks and corresponding text instructions for each task). We do not use these indicators when training/evaluating RL agents. For training REDS, we first collect subtask segmentations from 50 expert demonstrations for initial training and train DreamerV3 agents for 100K environment steps with the initial reward model to collect suboptimal trajectories, which is used for fine-tuning. In evaluation, we measure the success rate averaged over 10 episodes in every 20K steps. Please refer to Appendix B for more details.

Results Figure 3 shows that REDS consistently improves the sample-efficiency of DreamerV3 agents by outperforming all baselines. While baselines exhibit non-zero success rates in simple tasks like Faucet Close, their performance significantly deteriorates in more complex tasks, such as Peg Insert Side. On the other hand, our method maintains non-zero success rates across all tasks and even surpasses human-engineered reward functions in some tasks (e.g., Drawer Open, Push, Cof-

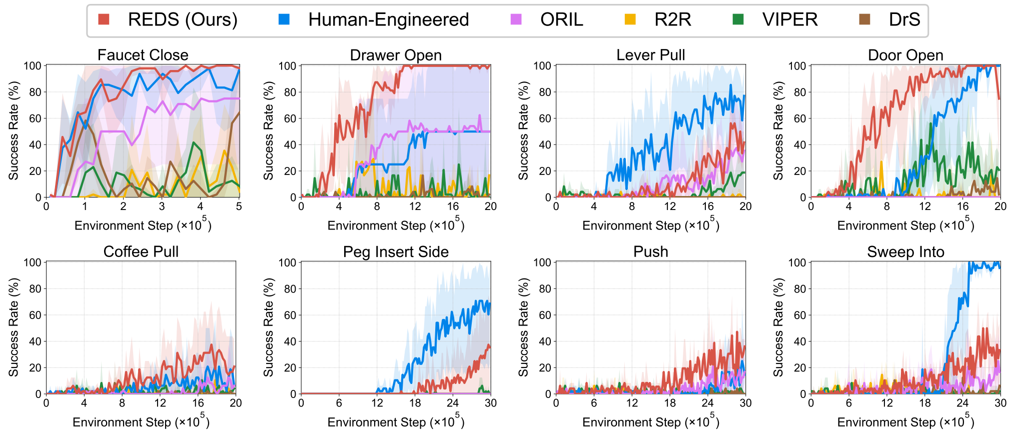


Figure 3: Learning curves of DreamerV3 [37] agents trained with different reward functions for solving eight robotic manipulation tasks from Meta-world [12], measured by success rate (%). The solid line and shaded regions represent the mean and stratified bootstrap interval across 4 runs.

fee Pull) without requiring task-specific reward engineering. These results show that REDS effectively generates appropriate rewards for solving intermediate tasks by leveraging subtask-segmented demonstrations.

3.2 FurnitureBench Experiments

Setup We further evaluate our method on real-world furniture assembly tasks from FurnitureBench [28], specifically focusing on One Leg assembly. This task involves a sequence of complex subtasks such as picking up, inserting, and screwing (see Figure 2d). For training REDS, we use 300 expert demonstrations with subtask segmentations provided by FurnitureBench, along with an additional 200 rollouts from IQL [38] policy trained with expert demonstrations in a single training iteration. To prevent misleading reward signals stemming from visual occlusions, we utilize visual observations from the front camera and wrist cameras in training REDS. For downstream RL, we first train offline RL agents using 300 expert demonstrations labeled with each reward model, followed by online fine-tuning to assess improvements. We provide more details in Appendix B.

Results As shown in Table 1, REDS achieves significant performance improvements through online fine-tuning, whereas the improvements from baselines are marginal. These results indicate that our method produces informative signals for solving a sequence of subtasks, while baselines either fail to provide context-aware signals or dense rewards for better exploration (see Appendix H for qualitative examples). Moreover, we note that our method outperforms the IQL trained with 500 expert demonstrations, achieving a score of 2.45 compared to 1.8 reported by FurnitureBench, despite using only 300 expert demonstrations. Considering that REDS does not require additional human interventions beyond resetting the environment, these results highlight the potential to extend our approach to a wider range of real-world robotics tasks.

Table 1: Online fine-tuning results of IQL agents in One Leg from FurnitureBench. We report the initial performance after offline RL (left) and the performance after 150 episodes of online RL (right).

Method	# Expert Demos	Completed Subtasks (Offline → Online)
Sparse (Offline) [28]	500	1.8
VIPER	300	1.10 → 1.25
DrS	300	1.05 → 1.10
REDS (Ours)	**300**	**1.10 → 2.45**

4 Conclusion

We proposed REDS, a visual reward learning framework considering subtasks by utilizing subtask segmentation. Our main contribution is based on proposing a new reward model leveraging minimal domain knowledge as a ground-truth reward function. Our approach is generally applicable and does not require any additional instrumentations in online interactions. We believe REDS will significantly alleviate the burden of reward engineering and facilitate the application of RL to a broader range of real-world robotic tasks.

References

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation*, 2017.

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *International Journal of Robotics Research*, 2020.

[4] L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In *Robotics: Science and Systems*, 2023.

[5] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. Van Wyk, A. Zhurkevich, B. Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *IEEE International Conference on Robotics and Automation*, 2023.

[6] S. Booth, W. B. Knox, J. Shah, S. Niekum, P. Stone, and A. Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *AAAI Conference on Artificial Intelligence*, 2023.

[7] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone. Reward (mis) design for autonomous driving. *Artificial Intelligence*, 2023.

[8] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[9] Y. Yuan, H. Che, Y. Qin, B. Huang, Z.-H. Yin, K.-W. Lee, Y. Wu, S.-C. Lim, and X. Wang. Robot synesthesia: In-hand manipulation with visuotactile sensing. *arXiv preprint arXiv:2312.01853*, 2023.

[10] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 2020.

[11] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

[12] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2020.

[13] T. Mu, Z. Ling, F. Xiang, D. C. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[14] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao, X. Yuan, P. Xie, Z. Huang, R. Chen, and H. Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

[15] C. Sferrazza, D.-M. Huang, X. Lin, Y. Lee, and P. Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

[16] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[18] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision*, 2018.

[19] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. In *Robotics: Science and Systems*, 2021.

[20] D. Yang, D. Tjia, J. Berg, D. Damen, P. Agrawal, and A. Gupta. Rank2reward: Learning shaped reward functions from passive video. In *IEEE International Conference on Robotics and Automation*, 2024.

[21] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *IEEE International Conference on Robotics and Automation*, 2018.

[22] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, 2021.

[23] S. Kumar, J. Zamora, N. Hansen, R. Jangir, and X. Wang. Graph inverse reinforcement learning from diverse videos. In *Conference on Robot Learning*. PMLR, 2022.

[24] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *International Conference on Learning Representations*, 2023.

[25] Y. J. Ma, W. Liang, V. Som, V. Kumar, A. Zhang, O. Bastani, and D. Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, 2023.

[26] A. Escontrela, A. Adeniji, W. Yan, A. Jain, X. B. Peng, K. Goldberg, Y. Lee, D. Hafner, and P. Abbeel. Video prediction models as rewards for reinforcement learning. In *Conference on Neural Information Processing Systems*, 2023.

[27] T. Huang, G. Jiang, Y. Ze, and H. Xu. Diffusion reward: Learning rewards via conditional video diffusion. In *European Conference on Computer Vision*, 2024.

[28] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[29] T. Mu, M. Liu, and H. Su. Drs: Learning reusable dense rewards for multi-stage tasks. In *International Conference on Learning Representations*, 2024.

[30] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. In *Conference on Neural Information Processing Systems*, 2017.

[31] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarejo, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*. PMLR, 2021.

[32] A. Gleave, M. D. Dennis, S. Legg, S. Russell, and J. Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations*, 2021.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Conference on Neural Information Processing Systems*, 2017.

[34] Y. Lee, A. Szot, S.-H. Sun, and J. J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In *Conference on Neural Information Processing Systems*, 2021.

[35] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

[36] Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, and S. Schaal. Learning dense rewards for contact-rich manipulation tasks. In *IEEE International Conference on Robotics and Automation*, 2021.

[37] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[38] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[39] A. Pan, K. Bhatia, and J. Steinhardt. The effects of reward misspecification: Mapping and mitigating misaligned models. In *International Conference on Learning Representations*, 2022.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 2021.

[41] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[42] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[43] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[44] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Machine Learning*, 2022.

[45] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. In *Conference on Robot Learning*. PMLR, 2022.

[46] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, 2022.

[47] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, 2023.

[48] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[49] K. Zolna, A. Novikov, K. Konyushkova, C. Gulcehre, Z. Wang, Y. Aytar, M. Denil, N. de Freitas, and S. Reed. Offline learning from demonstrations and unlabeled experience. In *Conference on Neural Information Processing Systems*, 2020.

[50] D. Xu and M. Denil. Positive-unlabeled reward learning. In *Conference on Robot Learning*, 2021.

[51] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[52] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. In *Conference on Neural Information Processing Systems*, 2022.

[53] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[54] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[55] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

[56] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Conference on Neural Information Processing Systems*, 2016.

[57] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[58] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarejo, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*, 2021.

[59] B. Wulfe, L. M. Ellis, J. Mercat, R. T. McAllister, and A. Gaidon. Dynamics-aware comparison of learned reward functions. In *International Conference on Learning Representations*, 2022.

[60] J. M. V. Skalse, L. Farnik, S. R. Motwani, E. Jenner, A. Gleave, and A. Abate. STARC: A general framework for quantifying differences between reward functions. In *International Conference on Learning Representations*, 2024.

[61] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *International Conference on Learning Representations*, 2024.

[62] A. Adeniji, A. Xie, and P. Abbeel. Skill-based reinforcement learning with intrinsic reward matching. *arXiv preprint arXiv:2210.07426*, 2022.

[63] X. Liang, K. Shu, K. Lee, and P. Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning. In *International Conference on Learning Representations*, 2022.

[64] S. James and A. J. Davison. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robotics and Automation Letters*, 2022.

[65] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[66] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2022.

[67] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn. Waypoint-based imitation learning for robotic manipulation. In *Conference on Robot Learning*, 2023.

[68] Z. Zhang, Y. Li, O. Bastani, A. Gupta, D. Jayaraman, Y. J. Ma, and L. Weihs. Universal visual decomposer: Long-horizon manipulation made easy. In *IEEE International Conference on Robotics and Automation*, 2024.

[69] L. Kou, F. Ni, Y. Zheng, J. Liu, Y. Yuan, Z. Dong, and H. Jianye. Kisa: A unified keyframe identifier and skill annotator for long-horizon robotics demonstrations. In *International Conference on Machine Learning*, 2024.

[70] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *IEEE International Conference on Robotics and Automation*, 2024.

[71] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[72] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[73] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[74] C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep object-centric representations for generalizable robot learning. In *IEEE International Conference on Robotics and Automation*, 2018.

A REDS Training and Inference

Inference For each transition (s_t, a, s_{t+1}) at timestep t, we compute the reward using s_{t+1}. To infer ongoing subtasks in REDS, we first encode the visual observations from executed actions and the history of previous observations using a pre-trained visual encoder and a causal transformer. The transformer's final output, $\mathbf{v}_t$, is used to predict the subtask. REDS selects the subtask index $\bar{i}$ by choosing the subtask embedding $\boldsymbol{e}_{\bar{i}}$ resulting in the highest cosine similarity with $\boldsymbol{v}_t$. The final reward is computed as $\hat{R}_\theta(s_{t-1}, o_t; U_{\bar{i}})$. Please refer to Appendix B for more details.

Training We outline the training procedure for REDS. First, we collect subtask segmentations from expert demonstrations, creating a dataset $\mathcal{D}^0$, and use it to train the initial reward model, M^0. However, reward models trained solely on expert data are susceptible to reward misspecification [39]. To address this, we iteratively collect suboptimal demonstrations and fine-tune the reward model using expert and suboptimal data. Unlike expert demonstrations, suboptimal demonstrations cover a broader range of states and more diverse observations, making manual segmentation labor-intensive and error-prone. To reduce the burden on human annotators, we develop an automatic subtask inference procedure, avoiding the need for manual segmentation.

Before the iterative process, we compute similarity scores for all states in the expert demonstrations using the initial reward model M^0. For each subtask U_i, we calculate a threshold T_{U_i} based on the similarity scores between the expert states and the corresponding instructions, ensuring T_{U_i} represents the minimum similarity required for successful subtask completion. In each iteration $i \in \{1, ..., n\}$, we proceed as follows:

- Step 1 (Suboptimal data collection): We train an RL agent using the reward model M^i and collect suboptimal demonstrations $\mathcal{D}^i_{\text{replay}}$ from the agent's replay buffer.
- Step 2 (Subtask inference for suboptimal data): For each timestep in the suboptimal trajectory, we infer the subtask index $\hat{i}$ using the same procedure as in inference and compute $\text{sim}(\boldsymbol{v}_t, \boldsymbol{e}_{\hat{i}})$. If the similarity falls below the threshold T_{U_i} at any timestep, we mark the subtask as failed and assign the remaining timesteps to that subtask.
- Step 3 (Fine-tuning): We fine-tune the reward model M^{i-1} using the combined dataset $\mathcal{D}^i = \mathcal{D}^i \cup \mathcal{D}^i_{\text{replay}}$ to obtain M^i.

We use the final reward model M^n for downstream RL training.

B Experiment Details

Training and inference details We used the open-source pre-trained CLIP [40] with ViT-B/16 architecture to encode images and subtask instructions for all experiments. We adopt a GPT [41] architecture with 3 layers and 8 heads for the causal transformer. To canonicalize our reward functions, we use the same $\mathcal{D}$ for both coverage distribution $\mathcal{D}_\mathcal{C}$ and potential shaping distribution $\mathcal{D}_\mathcal{S}$, and we estimate the expectation over state distributions using a sample-based average over 8 additional samples from $\mathcal{D}$ per sample. All models are trained with AdamW [42] optimizer with a learning rate of 1×10^{-4} and a mini-batch size of 32. To ensure robustness against visual changes, we apply data augmentations, including random shifting [43, 44] and color jittering. For optimization, we train REDS with AdamW [42] optimizer with a learning rate of 1×10^{-4}, weight decay of 2×10^{-2}, and a cosine decay schedule for adjusting the training learning rate. We apply a warm-up scheduling for the initial 500 gradient steps starting from a learning rate of 0. Note that the parameters for CLIP visual/text encoders have not been updated. For training downstream RL agents, we normalize the reward by dividing it by the maximum value observed in the expert demonstrations. We report the hyperparameters used in our experiments in Table 2. For both coverage distribution $\mathcal{D}_\mathcal{C}$ and potential shaping distribution $\mathcal{D}_\mathcal{S}$, we use the same dataset with subtask segmentations $\mathcal{D}^i$, unlike prior work dealing with arbitrarily random distributions because of the absence of subtask segmentations. To canonicalize our reward functions, we estimate the expectation over state distributions using a sample-based average over 8 additional samples from $\mathcal{D}$ per sample. To prevent false positive cases in predicting subatsks in online interactions, we add margins to similarity scores inversely proportional to the subtasks in online interactions. Specifically, we infer the subtask $\hat{i}$ as follows:

$$\hat{i} = \mathrm{argmax}_{i \in \{1,\dots,k\}} (\mathrm{sim}(\boldsymbol{v}_t, \boldsymbol{e}_i) + \eta * (k - i)), \tag{1}$$

where η is a hyperparameter for the margin between subtasks.

Table 2: Hyperparameters of REDS used in our experiments.

Hyperparameter	**Value**
Batch size	32 (Meta-world, RLBench), 8 (FurnitureBench)
Training steps	5000
Learning rate	0.0001
Optimizer	AdamW [42]
Optimizer momentum	$\beta_1 = 0.9$, $\beta_2 = 0.999$
Weight decay	0.02
Learning rate decay	Linear warmup and cosine decay
Warmup steps	500
Context length	4
Causal transformer size	3 layers, 8 heads, 512 units
EPIC canonical samples	8
ϵ for progressive reward signal	0.05
η for inferring subtasks	0.01 (Meta-world, RLBench), 0.05 (FurnitureBench)
number of training iterations n	2 (Meta-world, RLBench), 1 (FurnitureBench)

Meta-world experiments We use visual observations of $64 \times 64 \times 3$. To consistently use a single camera viewpoint over all tasks, we use the modified version of the `corner2` viewpoint as suggested by Seo et al. [45]. Expert demonstrations for each task are collected using scripted policies publicly released in the benchmark. We use an action repeat of 2 to accelerate training and set the maximal episode length as 250 for all Meta-world tasks. For downstream RL, we use the implementation of DreamerV3 from VIPER[2]. We report the hyperparameters of DreamerV3 agents used in our experiments in Table 3. Unless otherwise specified, we use the same set of hyperparameters as VIPER.

RLBench experiments For training both reward models and downstream RL agents, we utilize $64 \times 64 \times 3$ RGB observations from the front camera and wrist camera. For downstream RL, we don't use any expert demonstrations and we use the same set of hyperparameters as VIPER.

FurnitureBench experiments We use the implementation of IQL from FurnitureBench [3] for our experiments. We utilize $224 \times 224 \times 3$ RGB observations from the front camera and wrist cameras, along with proprioceptive states, to represent the current state. We encode each image with pre-trained R3M [46] for visual observations. Following [38], we first run offline RL for 1M gradient steps, then continue training while collecting environment interaction data, adding it to the replay buffer, and repeating this process for 150 episodes. Before online fine-tuning, we pre-fill the replay buffer with 10 rollouts from the pre-trained IQL policy. We adopt techniques from RLPD [47] for efficient offline-to-online RL training. Specifically, we sample 50% of the data from the replay buffer and the remaining 50% from the offline data buffer containing 300 expert demonstrations. We also apply LayerNorm [48] in the critic/value network of the IQL agent to prevent catastrophic overestimation. We list the hyperparameters used in our experiments in Table 4. For training REDS, we collect subtask segmentations for suboptimal demonstrations using the automatic subtask inference procedure described in Appendix A, and we manually modified some subtask segmentations with false negatives to guarantee stable performance. For baselines, we compare against VIPER and DrS. We emphasize that our method enables fully autonomous training in online RL sessions, in contrast

[2] https://github.com/Alescontrela/viper_rl

[3] https://github.com/clvrai/furniture-bench/tree/main/implicit_q_learning

Table 3: Hyperparameters of DreamerV3 [26] used in Meta-world experiments.

Hyperparameter	Value
General	
Replay Capacity (FIFO)	5×10^5
Start learning (prefill)	5000
MLP size	2×512
World Model	
RSSM size	512
Base CNN channels	32
Codes per latent	32

Table 4: Hyperparameters of IQL [38] used in FurnitureBench experiments.

Hyperparameter	Value
Learning rate	3×10^{-4}
Batch size	256
Policy # hidden units	(512, 256, 256)
Critic/value # hidden units	(512, 256, 256)
Image encoder	R3M [46]
Discount factor (γ)	0.996
Expectile (τ)	0.8
Inverse Temperature (β)	10.0

to DrS, which relies on a subtask indicator provided by humans. In our DrS experiments, subtasks were manually identified by a human. We measure the average number of completed subtasks over 20 rollouts for evaluation.

Computation We use 24 Intel Xeon CPU @ 2.2GHz CPU cores and 4 NVIDIA RTX 3090 GPUs for training our reward model, which takes about 1.5 hours in Meta-world and 3 hours in FurnitureBench due to high-resolution visual observations from multiple views. For training DreamerV3 agents in Meta-world, we use 24 Intel Xeon CPU @ 2.2GHz CPU cores and a single NVIDIA RTX 3090 GPU, which takes approximately 4 hours over 500K environment steps. For training IQL agents in FurnitureBench, we use 24 Intel Xeon CPU @ 2.2GHz CPU cores and a single NVIDIA RTX 3090 GPU, taking approximately 2 hours for 1M gradient steps in offline RL and 4.5 hours over 150 episodes of environment interactions in online RL.

C REDS Architecture Details

We encode visual observations with a pre-trained CLIP [40] ViT-B/16 visual encoder, utilizing all representations from the sequence of patches. We adopt 1D learnable parameters with the same size for positional embedding, and we add these parameters to 2D fixed sin-cos embeddings and add them to features. To encode temporal dependencies in visual observations, we use a GPT [41] architecture with 3 layers and 8 heads. In FurnitureBench, we use a sequence of images from both the front camera and wrist camera as input. Given $s_t^{\text{front}}/s_t^{\text{wrist}}$ from the front/wrist camera, we concatenate visual observations to $[o_{t-K-1}^{\text{front}}, o_{t-K-1}^{\text{wrist}}, ..., o_t^{\text{front}}, o_t^{\text{wrist}}]$, add positional embeddings, 2D fixed sin-cos embeddings, and additional 1D learnable parameters for each viewpoint for effectively utilizing images from multiple cameras. We then pass the features to the transformer layer, the same as the model with a single image. The subtask embedder and final reward predictor are implemented as 2-layer MLPs.

D Baseline Details

We consider the following baselines: (1) human-engineered reward functions provided in the benchmark, (2) ORIL [49], an adversarial imitation learning (AIL) method trained only with offline demonstrations, (3) Rank2Reward (R2R) [20], an AIL method which trains a discriminator weighted with temporal ranking of video frames to reflect task progress, (4) VIPER [26], a reward model utilizing likelihood from a pre-trained video prediction model as a reward signal, and (5) DrS [29], an AIL method that assumes subtask information from the environment and trains a separate discriminator for each subtask.

ORIL [49] For implementing ORIL with visual observations, we use the CNN architecture from Yarats et al. [43] to encode image observations. For training data, we use the same set of demonstrations as for training REDS. Since our training data are divided into success and failure demonstrations, we do not use positive-unlabeled learning [50] in our experiments. For robustness against visual changes, we apply the same augmentation techniques used for training REDS.

Rank2Reward (R2R) [20] To ensure compatibility with backbone RL algorithms [37, 38] implemented in JAX, we reimplement the reward model with JAX following the official implementation of Rank2Reward [4] and use the same hyperparameters. We first pre-train the ranking network using the same expert demonstrations as REDS, and we then train a discriminator for the expert demonstration and policy rollouts, weighted by the output from the pre-trained ranking network. For training efficiency, we use the CNN architecture from Yarats et al. [43] for encoding visual observations instead of R3M [46], finding no significant difference when we use the pre-trained visual representations like R3M, but with much slower training in online RL. We observe that our R2R implementation with DreamerV3 in JAX outperforms the original version implemented with DrQ-V2 [44] agents.

DrS [29] Similar to R2R, we reimplement DrS with JAX following the official implementation of DrS [5], and use the same set of hyperparameters for reward learning. As the original DrS implementation is based on a state-based environment, we switch the backbone RL algorithm from SAC to DrQ-V2 [44] and apply the augmentation technique in the reward learning phase for processing visual observations efficiently. To report the RL performance, we use the learned dense reward model to train new RL agents. In FurnitureBench experiment, we train the reward model with the same expert/failure demonstrations as in Section 3.2, without online interaction, to avoid unsafe behaviors and a significant increase in training time from online interactions.

VIPER [26] We use the official implementation of VIPER [6] for our experiments. Given the similarities among robotic manipulation tasks, we use the same set of hyperparameters as in RLBench [10] experiments to train VQ-GAN and VideoGPT. We train 100K steps, choosing the checkpoint with the minimum validation loss. In FurnitureBench experiment, we use images from the front camera, resized to $64 \times 64 \times 3$, and set the exploration objective β as 0.

E Task Descriptions

In this section, we list the subtasks and corresponding text instructions for each task in Table 5. For Meta-world tasks, we provide the code snippet used to determine the success of each subtask (Please refer to the Meta-world [12] for more details). For the FurnitureBench One Leg task, we outline the criteria used by human experts to assess the success of each subtask based on the metric defined in FurnitureBench [28].

F Related Work

Reward learning from videos Learning from observations without expert actions has been a promising research area because it does not require extensive instrumentation and allows for the easy collection of vast amounts of video from online sources. Notably, several studies have proposed methods for learning rewards directly from videos and using the signal to train RL agents. Previous work has been focused on learning a reward function by aligning video representations in temporal order [21, 22, 23] while others train a reward function for expressing the progress of the agent towards the goal [35, 34, 20]. Most recent work [26] inspired by the success of video generative models [51, 52] utilizes the likelihood of pre-trained video prediction models as a reward. To effectively utilize video for long-horizon tasks, we propose a new reward model conditioned both on video segments and corresponding subtasks trained with subtask segmentations.

[4] https://github.com/dxyang/rank2reward
[5] https://github.com/tongzhoumu/DrS
[6] https://github.com/Alescontrela/viper_rl

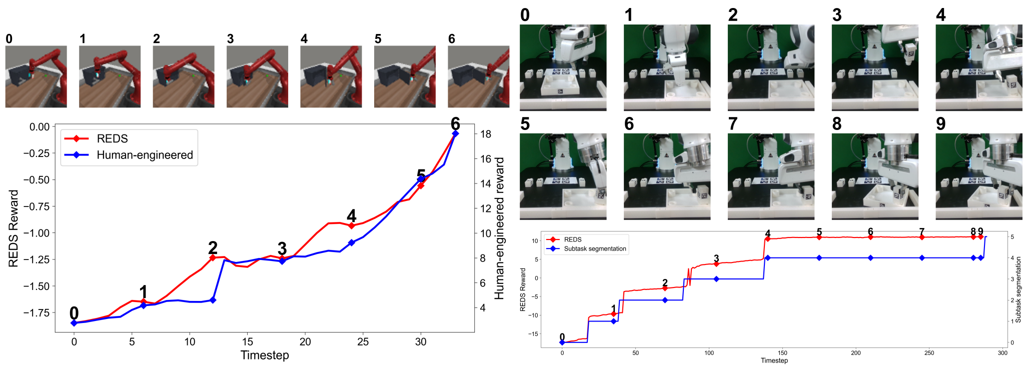
Table 5: A list of subtasks and language description for each subtask used in our experiments.

Task	Subtask	Success condition	Language description
Meta-world Faucet Close	1	$\mathrm{object\_grasped} \leq 0.9$	a robot arm reaching the faucet handle.
	2	$\mathrm{target\_to\_obj} \leq 0.07$	a robot arm rotating the faucet handle to the right.
Meta-world Drawer Open	1	$\mathrm{gripper\_error} \leq 0.03$	a robot arm grabbing the drawer handle.
	2	$\mathrm{handle\_error} \leq 0.03$	a robot arm opening a drawer to the green target point.
	3	$\mathrm{handle\_error} \leq 0.03$	a robot arm holding the drawer handle near the green target point after opening.
Meta-world Lever Pull	1	$\mathrm{ready\_to\_lift} > 0.9$	a robot arm touching the lever.
	2	$\mathrm{lever\_error} \leq \mathrm{np.pi}/24$	a robot arm pulling up the lever to the red target point.
Meta-world Door Open	1	$\mathrm{reward\_ready} \geq 1.0$	a robot arm grabbing the door handle.
	2	$\mathrm{abs}(\mathrm{obs}[4] - \mathrm{self.\_target\_pos}[0]) \leq 0.08$	a robot arm opening a door to the green target point.
	3	$\mathrm{abs}(\mathrm{obs}[4] - \mathrm{self.\_target\_pos}[0]) \leq 0.08$	a robot arm holding the door handle near the green target point after opening.
Meta-world Coffee Pull	1	$\mathrm{tcp\_to\_obj} < 0.04 \wedge \mathrm{tcp\_open} > 0$	a robot arm grabbing the coffee cup.
	2	$\mathrm{obj\_to\_target} \leq 0.07$	a robot arm moving the coffee cup to the green target point.
	3	$\mathrm{obj\_to\_target} \leq 0.07$	a robot arm holding the cup near the green target point.
Meta-world Peg Insert Side	1	$\mathrm{tcp\_to\_obj} < 0.02 \wedge \mathrm{tcp\_open} > 0$	a robot arm grabbing the green peg.
	2	$\mathrm{obj}[2] - 0.1 > \mathrm{self.obj\_init\_pos}[2]$	a robot arm lifting the green peg from the floor.
	3	$\mathrm{obj\_to\_target} \leq 0.07$	a robot arm inserting the green peg to the hole of the red box.
	4	$\mathrm{obj\_to\_target} \leq 0.07$	a robot arm holding the green peg after inserting.
Meta-world Push	1	$\mathrm{tcp\_to\_obj} \leq 0.03$	a robot arm grabbing the red cube.
	2	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm pushing the grabbed red cube to the green target point.
	3	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm holding the grabbed red cube near the green target point.
Meta-world Sweep Into	1	$\mathrm{self.touching\_main\_object} > 0 \wedge \mathrm{tcp\_opened} > 0$	a robot arm grabbing the red cube.
	2	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm sweeping the grabbed red cube to the blue target point.
	3	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm holding the grabbed red cube near the blue target point.
Meta-world Door Close	1	$\mathrm{in\_place} == 1.0$	a robot arm grabbing the door handle.
	2	$\mathrm{obj\_to\_target} \leq 0.08$	a robot arm closing a door to the green target point.
	3	$\mathrm{obj\_to\_target} \leq 0.08$	a robot arm holding the door handle near the green target point after closing.
Meta-world Window Close	1	$\mathrm{tcp\_to\_obj} \leq 0.05$	a robot arm grabbing the window handle.
	2	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm closing a window from left to right.
	3	$\mathrm{target\_to\_obj} \leq 0.05$	a robot arm holding the window handle after closing.
FurnitureBench One Leg	1	robot gripper tips make contact with one surface of the tabletop.	a robot arm picking up the white tabletop.
	2	nearest corner of the tabletop is placed close to the right edge of the obstacle.	a robot arm pushing the white tabletop to the front right corner.
	3	robot gripper securely grasps a leg of the table and lifts it.	a robot arm picking up the white leg.
	4	leg is inserted into one of the screw holes of the tabletop, and the robot releases the gripper.	a robot arm inserting the white leg into screw hole.
	5	leg is fully assembled to the tabletop.	a robot arm screwing the white leg until tightly lifted.
	6	leg is fully assembled to the tabletop.	a robot arm holding the white leg in place.
RLBench Take Umbrella Out of Umbrella Stand	1	$\mathrm{GraspedCondition}(\mathrm{self.robot.gripper}, \mathrm{self.umbrella}).\mathrm{condition_met}()[0]$	a robot arm grasping the umbrella.
	2	$\mathrm{DetectedCondition}(\mathrm{self.umbrella}, \mathrm{self.success_sensor}, \mathrm{negated} = \mathrm{True}).\mathrm{condition_met}()[0]$	a robot arm taking the grasped umbrella ouf of the umbrella stand.
	3	$\mathrm{DetectedCondition}(\mathrm{self.umbrella}, \mathrm{self.success_sensor}, \mathrm{negated} = \mathrm{True}).\mathrm{condition_met}()[0]$	a robot arm holding the umbrella on the umbrella stand.

Inverse reinforcement learning Designing an informative reward function remains a long-standing challenge for training RL agents. To achieve this, Inverse Reinforcement Learning (IRL) [53, 54, 55] aims to estimate the underlying reward function from expert demonstrations. Adversarial imitation learning (AIL) approaches [56, 57, 49, 58, 29] address this by training a discriminator network to discriminate transitions from expert data or policy rollouts and using the output from the discriminator as a reward for training agents with RL. The most similar work to ours is DrS [29], which also utilizes subtask information of the multi-stage task. While DrS assumes that the information on ongoing subtasks can be obtained from the environment during online interaction, our method has no such assumption, so it can be applied in more general cases when the segmenting of the subtask is hard in automatic ways (e.g., [28]).

Quantifying differences between reward functions Previous work has explored methods for measuring the difference between reward functions without relying on policy optimization procedures [32, 59, 60]. In particular, Gleave et al. [32] introduced the EPIC distance, a pseudometric invariant to equivalent classes of reward functions. Subsequent work [61, 62, 63] has employed EPIC to assess the quality of reward functions. In this paper, we take a different approach by using EPIC distance as an optimization objective. While Adeniji et al. [62] also utilizes EPIC distance for optimizing intrinsic reward functions in skill discovery, our method applies EPIC distance to train dense reward functions for long-horizon tasks, serving as a direct reward signal for RL training.

Segmenting demonstrations for long-horizon manipulation tasks Several approaches have been proposed to decompose long-horizon demonstrations into multiple subgoals to prevent error accumulation and provide intermediate signals for agent training. These include extracting key points from proprioceptive states [64, 65, 66, 67], employing greedy heuristics on off-the-shelf visual representations pre-trained with robotic data [68], and learning additional modules on top of pre-trained visual-language models to align with keyframes [69]. Our work builds on these efforts by leveraging subtask segmentations but focuses on developing a reward learning framework that explicitly incorporates subtask decomposition to generate suitable reward signals for intermediate tasks. Additionally, we further demonstrate that our model generalizes effectively to unseen tasks and robot embodiments.



(a) Door Open

(b) One Leg

Figure 4: Qualitative results of REDS in Door Open in Meta-world [12] and One Leg from FurnitureBench [28]. We observe that REDS produces suitable reward signals aligned with ground-truth reward functions by predicting ongoing subtasks effectively and providing progressive reward signals.

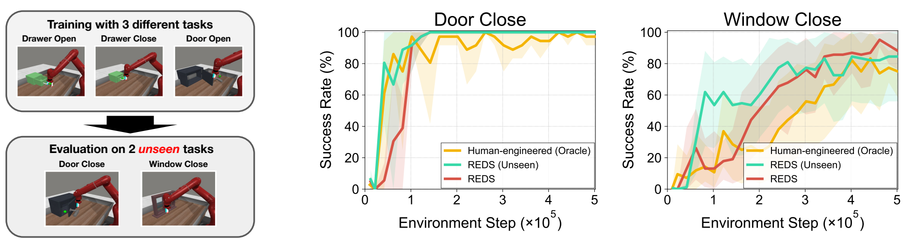


Figure 5: We train REDS with 3 different tasks from Meta-world [12] and use this model to train RL agents in 2 unseen tasks (left). We present learning curves on Door Close (center) and Window Close (right), as measured by success rate (%). The solid line and shaded regions represent the mean and stratified bootstrap interval across 4 runs.

G Additional Experimental Results

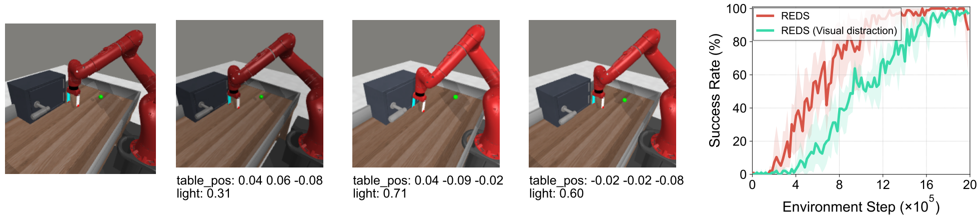
G.1 Alignment with Ground-truth Rewards

EPIC measurement To quantitatively validate the alignment of our method with ground-truth reward functions, we measure the EPIC distance with a set of unseen demonstrations during training. Specifically, we use rollouts from the reference policy trained with expert demonstrations for state distribution. In Table 6, we observe that REDS exhibits significantly lower EPIC distance than baselines across all tasks. Particularly, the difference between REDS and baselines is more pronounced in complex tasks like One Leg. This result consistently supports the empirical findings from previous sections.

Table 6: EPIC [32] distance (lower is better) between learned reward functions and hand-engineered reward functions (Meta-world) / subtask segmentations (FurnitureBench) in unseen data.

Task	VIPER	R2R	ORIL	REDS (Ours)
Meta-world Door Open	0.5934	0.5649	0.7071	**0.4913**
Meta-world Push	0.6144	0.6838	0.7073	**0.5381**
Meta-world Peg Insert Side	0.5974	0.5806	0.6989	**0.4674**
Meta-world Sweep Into	0.6248	0.6413	0.7001	**0.4673**
FurnitureBench One Leg	0.7035	0.6001	0.7014	**0.0713**

Qualitative analysis We provide the graph of computed rewards from REDS in Figure 4. We observe that REDS can induce suitable reward signals aligned with ground-truth reward functions. For example, REDS provides subtask-aware signals in transition states (e.g., between 2 and 3, and between 4 and 5) and generates progressive reward signals throughout each subtask. Please refer to Appendix H for the extensive comparison between REDS and baselines.



(a) Original (b) Examples of visual distractions (c) Learing curve

Figure 6: We provide visual observations from (a) the original environment and (b) unseen environments with visual distractions used in our experiments in Section G.2 .

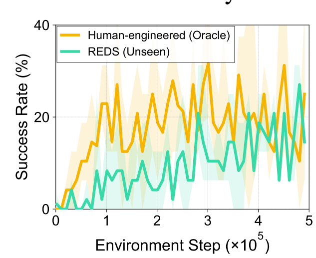
G.2 Generalization Capabilities

Transfer to unseen tasks REDS can be applied as a reward function in unseen tasks. To validate this, we conduct additional experiments by training REDS with segmentation data from 3 tasks (Door Open, Drawer Open/Close) and using the reward model to train RL agents in two unseen tasks. In Door Close, we aim to validate that REDS can provide informative signals for a new task involving a previously seen object and behaviors. In Window Close, we aim to determine whether REDS can provide suitable reward signals for familiar behaviors (closing) with an unseen object (window). In evaluation, we change the text instruction following the target object (as shown in Table E), and we do not fine-tune the reward model. Figure 5 shows that REDS provides effective reward signals on unseen tasks and achieves comparable or even better RL performance than REDS trained on the target task. This result demonstrates that REDS can be applied to RL training in unseen tasks that share properties with training tasks.

Robustness to visual distractions To prove the robust performance of REDS against visual distractions, we train RL agents with our reward model in new Meta-world environments incorporating visual distractions, such as varying light and table positions following [70] (see Figure 6b). Note that the reward model was trained using demonstrations only from the original environment. As Figure 6c shows, REDS can generate robust reward signals despite visual distractions and train RL agents to solve the task effectively.

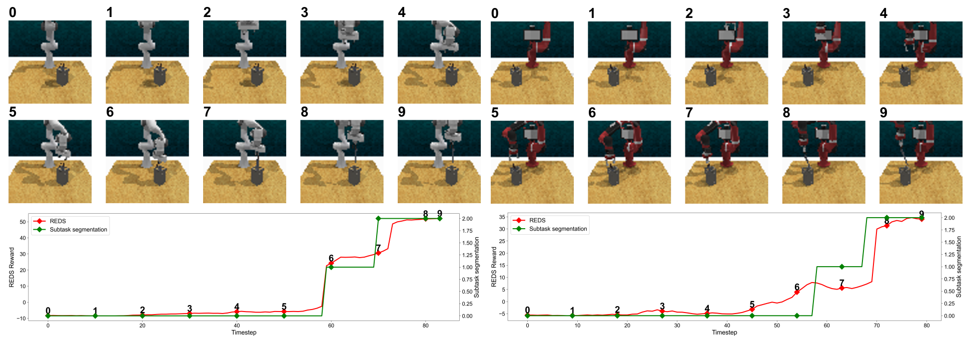
Transfer to unseen embodiments Since our framework leverages only action-free video data, we hypothesize that transferring to other robot embodiments with similar DoFs is feasible. To support this claim, we train REDS with demonstrations of the Franka Panda Arm and then compute the reward of an unseen demonstration of the Sawyer Arm in Take Umbrella Out of Stand from RLBench [10]. Figure 8 shows that REDS generates informative reward signals even with the unseen embodiment. For instance, REDS can capture the behavior of taking the umbrella out of the stand, as indicated by the increased reward signals between 6 and 7. Additionally, Figure 7 shows that REDS trained only with the Panda Arm can be used to train downstream RL agents in the environment with the Sawyer Arm.

Figure 7: Learning curve for DreamerV3 agents in environments of the Sawyer Arm.



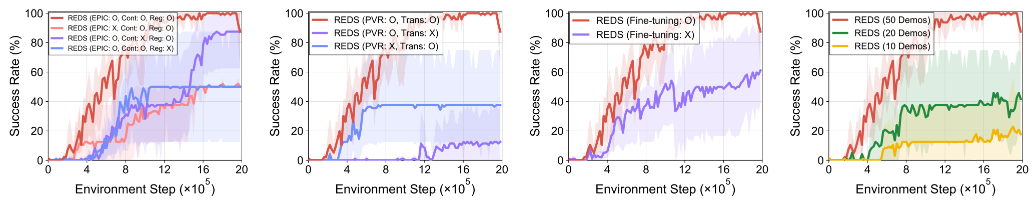
G.3 Ablation Studies

Effect of training objectives We investigate the effect of each training objective in Figure 9a. Specifically, we compare REDS with 1) a baseline trained with regression to subtask segmentation instead of EPIC loss $\mathcal{L}_{\text{EPIC}}$, 2) a baseline that utilizes only video representations without subtask embeddings, and 3) a baseline trained without the regularization loss $\mathcal{L}_{\text{reg}}$. We observe that RL performance significantly degrades without each component, implying that our losses synergistically improve reward quality.



(a) (Train) Panda (b) (Unseen) Sawyer

Figure 8: Qualitative results of REDS with different robot embodiments. REDS was trained using demonstrations from the Panda Arm and evaluated on an unseen demonstration from the Sawyer Arm in Take Umbrella Out of Stand from RLBench [10]. We visualize several frames above the graph and mark them with a diamond symbol.



(a) Training objectives (b) Architecture (c) Fine-tuning (d) Expert demonstrations

Figure 9: Learning curves for two Meta-world [12] robotic manipulation tasks, measured by success rate (%), to examine the effects of (a) training objectives, (b) architecture, (c) fine-tuning, and (d) the number of expert demonstrations. The solid line and shaded regions show the mean and stratified bootstrap interval across 8 runs.

Effect of architecture To verify the design choice, we compare REDS with 1) a baseline using a CNN for encoding images instead of pre-trained visual representations (PVR) and 2) a baseline simply concatenating pre-trained visual representations without a causal transformer. Figure 9b shows that both baselines show worse performance compared to ours. Notably, detaching a causal transformer significantly degrades RL performance, implying that temporal information is essential for providing suitable reward signals in robotic manipulation.

Effect of fine-tuning In Figure 9c, we compare REDS trained only with the expert demonstrations in the initial phase to REDS fine-tuned with additional suboptimal demonstrations as described in Appendix A. REDS shows improved RL performance when trained with additional suboptimal demonstrations, indicating that the coverage of state distribution impacts the reward quality. Further investigation on how to efficiently collect suboptimal demonstrations to enhance the performance of learned reward function is a promising future direction.

Effect of the number of expert demonstrations We investigate the effect of the number of expert demonstrations by measuring the RL performance of DreamerV3 agents with REDS trained with different numbers of expert demonstrations in 2 tasks (Door Open, Drawer Open) from Meta-world. Figure 9d shows that the agents' RL performance positively correlates with the number of expert demonstrations trained for reward learning.

H Extended Qualitative Analysis

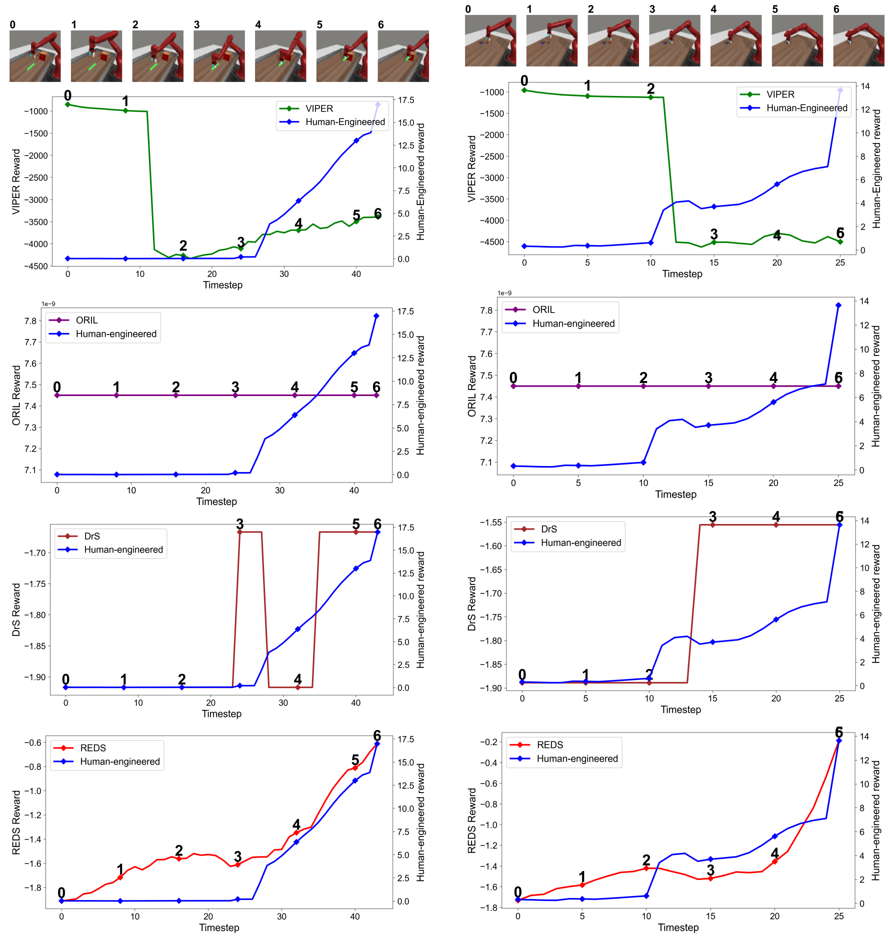


Figure 10: Qualitative results of VIPER [26], ORIL [49], DrS [29], and REDS (Ours) in Peg Insert Side (left), and Sweep Into (right) from Meta-world [12]. We visualize several frames above the graph and mark them with a diamond symbol.

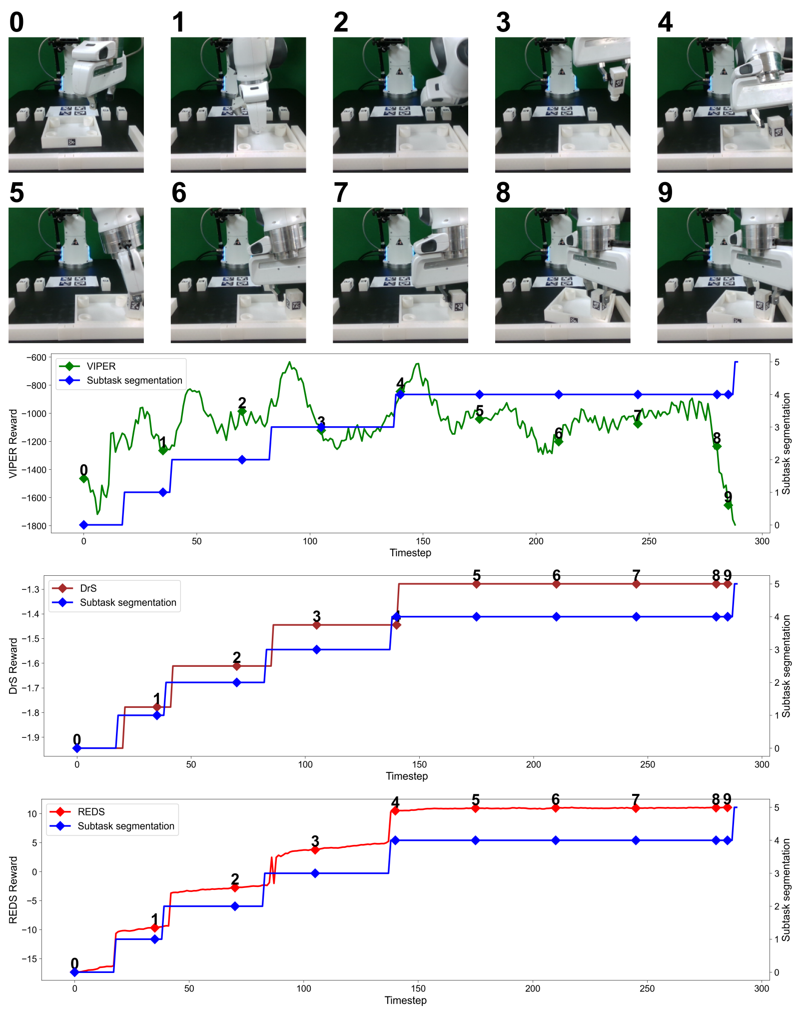


Figure 11: Qualitative results of VIPER [26], DrS [29], and REDS (Ours) in One Leg from FurnitureBench [28]. We visualize several frames above the graph and mark them with a diamond symbol. VIPER, which does not utilize subtask information, failed to produce suitable rewards for transitioning between subtasks. While DrS uses ground-truth subtask information from the environment, it produces sparse reward signals within each subtask. In contrast, REDS provides subtask-aware signals in transition states (e.g., between 2 and 3, and 4 and 5) and generates dense reward signals throughout each subtask.

I Limitation and Future Directions

One limitation of our work is the reliance on pre-trained representations trained with natural image/text data for encoding videos and subtasks. Although REDS proves its effectiveness in various robotic manipulation tasks, we observe that REDS struggles to distinguish subtle changes. We believe that the quality of rewards can be further improved by utilizing 1) larger models pre-trained with large-scale data with diverse robotic tasks [71, 72] and 2) representations trained with objectives considering affordances [73] or object-centric methods [74]. Moreover, the number of expert demonstrations and the number of iteration for fine-tuning REDS are determined by empirical trials. Investigating how to efficiently collect failure demonstrations to mitigate reward misspecification is an interesting future direction.

J Ethics Statement

Video demonstrations and subtask segmentations used in the experiments were sourced from publicly available benchmarks (Meta-world, RLBench, FurnitureBench), ensuring no personal or sensitive information is involved. Potential risks could arise when training and deploying RL agents directly in real-world scenarios, particularly in human-robot interactions. Ensuring the safety and reliability of these agents before deployment is essential to prevent harm.

K Reproducibility Statement

For the reproducibility of REDS, we have provided a detailed explanation of implementation details and experimental setups in Appendix A, Section 3, and Appendix B. In addition, to further facilitate the reproduction, we attach the source code used in our experiments in the supplementary materials.