*robotics*

MDPI

*Article*

# MURM: Utilization of Multi-Views for Goal-Conditioned Reinforcement Learning in Robotic Manipulation

**Seongwon Jang, Hyemi Jeong and Hyunseok Yang ***

Mechanical Engineering Department, Yonsei University, Seoul 03722, Republic of Korea; jj22jjj4@yonsei.ac.kr (S.J.); stree104@yonsei.ac.kr (H.J.)

* Correspondence: hsyang@yonsei.ac.kr

**Abstract:** We present a novel framework, multi-view unified reinforcement learning for robotic manipulation (MURM), which efficiently utilizes multiple camera views to train a goal-conditioned policy for a robot to perform complex tasks. The MURM framework consists of three main phases: (i) demo collection from an expert, (ii) representation learning, and (iii) offline reinforcement learning. In the demo collection phase, we design a scripted expert policy that uses privileged information, such as Cartesian coordinates of a target and goal, to solve the tasks. We add noise to the expert policy to provide sufficient interactive information about the environment, as well as suboptimal behavioral trajectories. We designed three tasks in a Pybullet simulation environment, including placing an object in a desired goal position and picking up various objects that are randomly positioned in the environment. In the representation learning phase, we use a vector-quantized variational autoencoder (VQVAE) to learn a more structured latent representation that makes it feasible to train for RL compared to high-dimensional raw images. We train VQVAE models for each distinct camera view and define the best viewpoint settings for training. In the offline reinforcement learning phase, we use the Implicit Q-learning (IQL) algorithm as our baseline and introduce a separated Q-functions method and dropout method that can be implemented in multi-view settings to train the goal-conditioned policy with supervised goal images. We conduct experiments in simulation and show that the single-view baseline fails to solve complex tasks, whereas MURM is successful.

**Keywords:** goal-conditioned reinforcement learning (GCRL); multiple camera views; robot manipulation; vector-quantized variational autoencoders (VQVAE)



**Citation:** Jang, S.; Jeong, H.; Yang, H. MURM: Utilization of Multi-Views for Goal-Conditioned Reinforcement Learning in Robotic Manipulation. *Robotics* **2023**, *12*, 119. https://doi.org/10.3390/robotics12040119

Academic Editor: Guanghui Wen

Received: 18 July 2023
Revised: 14 August 2023
Accepted: 17 August 2023
Published: 19 August 2023



## 1. Introduction

One of the most remarkable characteristics of human intelligence is that humans can achieve one goal after another. For example, if a person is asked to clean up a messy table, they finish the task by achieving minor goals such as putting trash in a bin and closing a drawer. Although standard reinforcement learning (RL) algorithms [1] have proposed remarkable works like beating a human in Go [2] and learning robotic manipulations [3,4], they are limited to learning a single policy at a time, specific to individual tasks. To tackle multiple goals simultaneously, goal-conditioned reinforcement learning (GCRL) was recently proposed [5] where the agents are expected to make decisions according to different goals given. GCRL aims to train the agent to efficiently achieve specifically assigned goal states which can be classified into three typical representations: desired property and feature vectors, images, and languages [6]. Desired property and feature vectors are defined as goals with specific feature information such as particular positions and rotations of an object or desired moving speeds and language goals are defined as goals expressed in the form of instruction sentences or words. In this work, we are especially focused on dealing with image goals which are a straightforward representation compared to vectors describing simple properties and features like the Cartesian coordinates of an object.

To this end, previous works [7–12] dealing with image goals for GCRL are dedicated to learning a meaningful latent representation for training from an image taken from a single camera that outlooks the entire scene environment. However, these studies did not consider adding other camera viewpoints to provide more comprehensive information about the environment, which could potentially enhance task performance. If multiple camera views that provide important state information of the environment can be efficiently utilized, tasks where a single camera view is not enough to solve the task can be tackled. Furthermore, even in cases where a single camera can provide sufficient state information to solve the task, the addition of another viewpoint can improve the agent's task performance.

In order to train the agent to solve GCRL by effectively utilizing multiple camera viewpoints, we propose the multi-view unified reinforcement learning for robotic manipulation(MURM) algorithm and our main contributions are as follows:

- We propose a novel framework, MURM, for solving goal-conditioned RL by leveraging images from multiple viewpoints with two effective implementation methods called the dropout method and the separated Q-functions method.
- We empirically show the effectiveness of our framework MURM in complicated tasks compared to single-view baselines.

Our paper is organized as follows. We discuss recent research related to our work in the Related Works section and discuss the main theories in the Preliminaries section. Our proposed methodology is discussed in the Methods section and the results are compared with the existing SOTA method and shown in the Experimental Evaluation section. Lastly, we discuss the results and future works in the Conclusion section.

## 2. Related Works

Goal-Conditioned Reinforcement Learning (GCRL) with image goals: Although standard RL [1] can only train the agent to complete one specific task defined by the reward function, GCRL focuses on more general and complex tasks. In GCRL, the agent can learn to tackle multiple tasks simultaneously or decompose complicated, long-horizon tasks into intermediate subgoals. Goals in GCRL are mainly classified into desired property and feature vectors, language goals, and image goals. Our work focuses on image goals, which require a good representation to solve tasks effectively. It is possible to directly feed state and goal images into RL algorithms and train them in an end-to-end manner [13] where Deep RL can automatically find a good representation. However, this approach has been proven to be challenging in previous works, and the high dimensionality of images makes it more practical to decouple representation learning objectives from RL and use compact vector representation methods [7–12].

For example, Hansen-Estruch et al. [7] proposed goal-conditioned bisimulation (GCB), a new representation learning approach for goal-conditioned problems that uses functional equivariance over a family of goal-conditioned tasks. Furthermore, Qian et al. [9] proposed the DR-GRL framework that combines goal-conditioned visual RL with disentangled representation learning, where a spatial transform autoencoder was introduced to learn a controllable representation. Additionally, Nair et al. [10] developed a method to learn a latent representation using generative model variational autoencoders (VAE), which offers a structured representation and enables state sampling that can be utilized for sampling goals in GCRL. Similarly, Cong et al. [8] used VAE to learn a latent representation, which is combined with the robot end-effector position to form the state for RL, and trained the robot to solve a planar object-pushing task. Instead of a standard VAE model, Nair et al. [11] advanced prior work [10] by proposing a context-conditioned generative model (CC-VAE) that conditions a latent goal on the latent representation encoded from the initial state image. This approach enabled the model to propose currently feasible goals in diverse settings. Moreover, Khazatsky et al. used a vector-quantized variational VAE (VQVAE) [12] to solve GCRL, which is capable of representing diverse datasets expressively and generating more consistent and reasonable goal images. Most recently, Fang et al. [14]

proposed a conditional subgoal generator based on CCVAE and VQVAE which can train the goal-conditioned policy to solve long-horizon tasks. The aforementioned methods have mainly focused on either learning more compact representations from images or more effective implementations of learned representation in GCRL. However, when using representations from images, all prior methods utilize only images from a single-camera viewpoint. Our work, on the other hand, focuses on using images from multiple cameras, which allows for more efficient training of the goal-conditioned policy and outperforms single-view baselines.

Robotic Manipulation utilizing Multi-views: In the field of robotics, the fusion of multi-modal sensor information is a well-studied problem and numerous algorithms have been developed to account for all available information about the world [15–19]. However, we are primarily interested in fusing multiple sensor information of the same modality in robot manipulation settings, which remains somewhat of an uninvestigated topic [20–23]. Plappert et al. [22] proposed asymmetric self-play, where one agent learns to propose goals and the other learns to solve the proposed goals. They trained a single, goal-conditioned policy for the ShapeNet training environment using state observations and images from three cameras. However, vision data were utilized only as additional information to state information. S James et al. [21] presented a sample efficient robot learning algorithm that utilizes Q-attention [24] and allows discretization of the translation space, which can support multiple camera inputs. Moreover, Jangir et al. [20] used transformers with a cross-view attention mechanism to fuse visual information from multiple cameras and successfully trained an RL policy with the learned features. Furthermore, Akinola et al. [23] presented several multi-view approaches to robot learning of precise tasks with reinforcement learning. In the work of Younggyo Seo et al. [25], a reinforcement learning framework for learning multi-view representations was proposed, which utilizes multi-view masked autoencoders for a variety of visual robotic manipulation scenarios. Although the aforementioned works have utilized multiple cameras in robotic manipulation systems, to the best of the authors' knowledge, this paper is the first research to tackle GCRL with the utilization of multiple camera views in robot manipulation tasks.

## 3. Preliminaries

In this section, we cover preliminaries on variational autoencoders (VAE) with a focus on vector-quantized varational autoencoders (VQ-VAE). We also discuss goal-conditioned reinforcement learning and offline reinforcement learning.

### *3.1. Variational Autoencoders (VAE)*

To handle high-dimensional images, a latent representation of the state can be learned by using variational autoencoders (VAEs). VAEs are probabilistic generative models that successfully learn to encode high-dimensional data, such as images, into a lower-dimensional structured representation and decode it back to the original input. A standard structure of VAE is shown in Figure 1. The generative process of VAE is composed of an encoder that generates a set of latent variable $z$ from the prior distribution $q_\theta(z|x)$ and a decoder that generates data $x$ by the generative distribution $p_\psi(x|z)$ while keeping the latent $z$ similar (in Kullback–Leibler divergence [26]) to its prior $p(z)$, which is a standard normal distribution. The parameters $(\theta, \psi)$ of the encoder and decoder are jointly trained to minimize the negative evidence lower bound:

$$\mathcal{L}_{VAE}(\theta, \psi) = -\mathbb{E}_{q_\theta(z|x)}\left[\log p_\psi(x|z)\right] + \beta D_{KL}(q_\theta(z|x)||p(z)) \quad (1)$$

where $\mathbb{E}$ is the expected value, $D_{KL}$ is the KL divergence, and $\beta$ is a parameter that balances the two terms and when the use of $\beta$ values are other than one, it is referred to as a $\beta$-VAE [27].

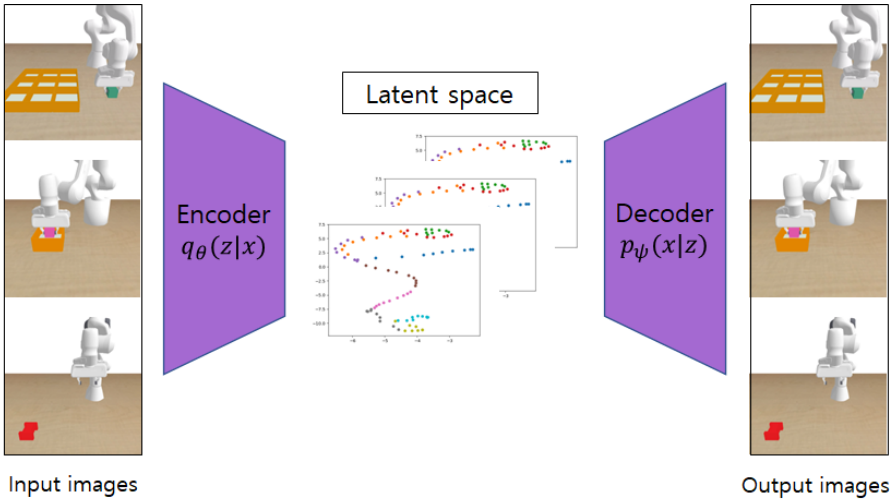



**Figure 1.** Standard structure of VAE.

Although standard variational autoencoders (VAEs) are a powerful tool for unsupervised learning, they suffer from several limitations, such as the continuous nature of the latent space and the difficulty of capturing local structures in high-dimensional data. To address these issues, vector-quantized variational autoencoders (VQ-VAEs) [28] were introduced as an extension of the standard VAE architecture, incorporating vector quantization into the encoding process. This modification allows VQ-VAEs to learn a discrete and interpretable representation of the data while addressing the challenges of continuous latent spaces and local structure capture. VQ-VAEs have been shown to be highly effective at learning a discrete and compact representation of high-dimensional data, making them ideal for tasks such as compression, noise reduction, and data generation. In this study, we utilize VQ-VAEs to capture the local structure of the data better and learn a more interpretable latent representation.

### *3.2. Goal-Conditioned Reinforcement Learning*

Goal-conditioned reinforcement learning (GCRL) is a machine learning method that trains an agent to achieve specified goals in particular scenarios [6]; whereas standard RL can be applied to solve various problems, GCRL is especially well-suited to tasks that require achieving specific outcomes, such as navigating a maze or performing a particular sequence of actions. As the agent focuses on the specific actions needed to achieve the goal, rather than exploring a broader set of possible actions, GCRL allows more efficient and targeted learning.

The formulation of the Markov Decision Process (MDP) remains the same with standard RL, except that an extra tuple $(p, r_g)$ is augmented. The goal-augmented MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, p, \gamma, r_g)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p(s_{t+1}|s_t, a_t)$ is the environment dynamics function, $\gamma$ is the discount factor, and $r_g$ is the goal-conditioned reward function. The aim of the agent is to optimize a policy $\pi(a_t|s_t, g)$ to maximize the expected discounted goal-conditioned return:

$$J_\pi = E_{g\in\mathcal{G}}\left[\sum_{i=0}^{H} \gamma^t r_g(s_t)\right] \tag{2}$$

where $\mathcal{G}$ is the space of goals describing the tasks and $H$ is the horizon, which may be infinite. Rewards in standard RL are typically scalar values, with higher values indicating better performance. However, rewards in GCRL are typically given according to the agent's progress towards the given goal, positive rewards to agents who make progress towards the goal, and negative rewards to agents who move away from the goal. Since more targeted feedback on its performance is provided with GCRL, many researchers claim that GCRL allows the agent to learn more quickly and effectively. A variety of algorithms can be

implemented for GCRL such as Trust Region Policy Optimization (TRPO) [29] and Deep Deterministic Policy Gradient (DDPG) [30]. However, we also prioritize an algorithm suitable for offline RL training as well as goal-conditioned settings. Thus, in our work, we employ the Implicit Q-Learning (IQL) [31] algorithm as the underlying RL algorithm, which is the SOTA algorithm for our settings.

### *3.3. Offline Reinforcement Learning*

In offline reinforcement learning (RL), the agents learn from a fixed dataset of collected experiences prior, rather than interacting with the environment in real-time. In the dataset, a behavior policy that may be optimal or near-optimal is typically collected and the goal of offline RL is to use this dataset to learn an optimized policy that maximizes the expected cumulative reward. Many recent offline Rl algorithms are built on approximate dynamic programming methods that minimize temporal difference error, according to the following loss:

$$L(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma \max_{a'} Q_{\hat{\theta}}(s',a') - Q_\theta(s,a))^2] \quad (3)$$

with $\mathcal{D}$ as the dataset, $Q_\theta(s,a)$ as a parameterized Q-function, $Q_{\hat{\theta}}(s,a)$ as a target network, and the policy defined as $\pi_\beta = arg\max_a Q_\theta(s,a)$. In recent offline reinforcement learning methods, modifications have been made to either regularize the value function loss in a manner that maintains the resulting policy's proximity to the data or to directly restrict the policy which often results in overestimation.

Offline RL has several advantages over online RL, such as the ability to reuse existing datasets, learn policies without the need for online experiments, and potentially learn from human expert demonstrations. Offline RL has garnered significant attention in the field of robotics due to its potential to reduce the time-consuming process of collecting data and enable solving tasks without an exploration process that may damage the hardware or cause harm to humans. However, applying offline RL to real-world environments poses several challenges, such as distributional shifting and overfitting due to differences between simulation and real-world environments. To address these challenges, methods such as offline RL with online fine-tuning have been proposed. However, in the scope of our experiments, we focus on the offline setting to decouple from exploration difficulties.

## 4. Methods

In this section, we propose our novel Multi-View Unified Reinforcement Learning for Manipulation (MURM) module, which leverages multiple camera views to perform goal-conditioned reinforcement learning in robotic manipulation tasks. Our method consists of a three-phase pipeline, and the full scheme is shown in Figure 2. In phase 1 (Section 4.1), we collect prior data consisting of varied demonstrations for the given tasks which are later used in offline RL. The demonstrations are designed by using scripted policies with privileged information in our simulation environment (e.g., Cartesian coordinates of the target and goal) to solve the given tasks. In Phase 2 (Section 4.2), we train VQVAE models to learn a latent representation of the state, which are high-dimensional images provided by different camera viewpoints. In Phase 3 (Section 4.3), we train a goal-conditioned policy with offline RL, and Implicit Q-Learning (IQL) is used as our underlying RL algorithm. We introduce two distinct methods that can be used for multi-view settings: the dropout method and the separated Q-functions method; whereas simply adding camera views to a conventional SOTA method downgrades the performance, our methods can train the agent in multi-view settings to outperform conventional methods.

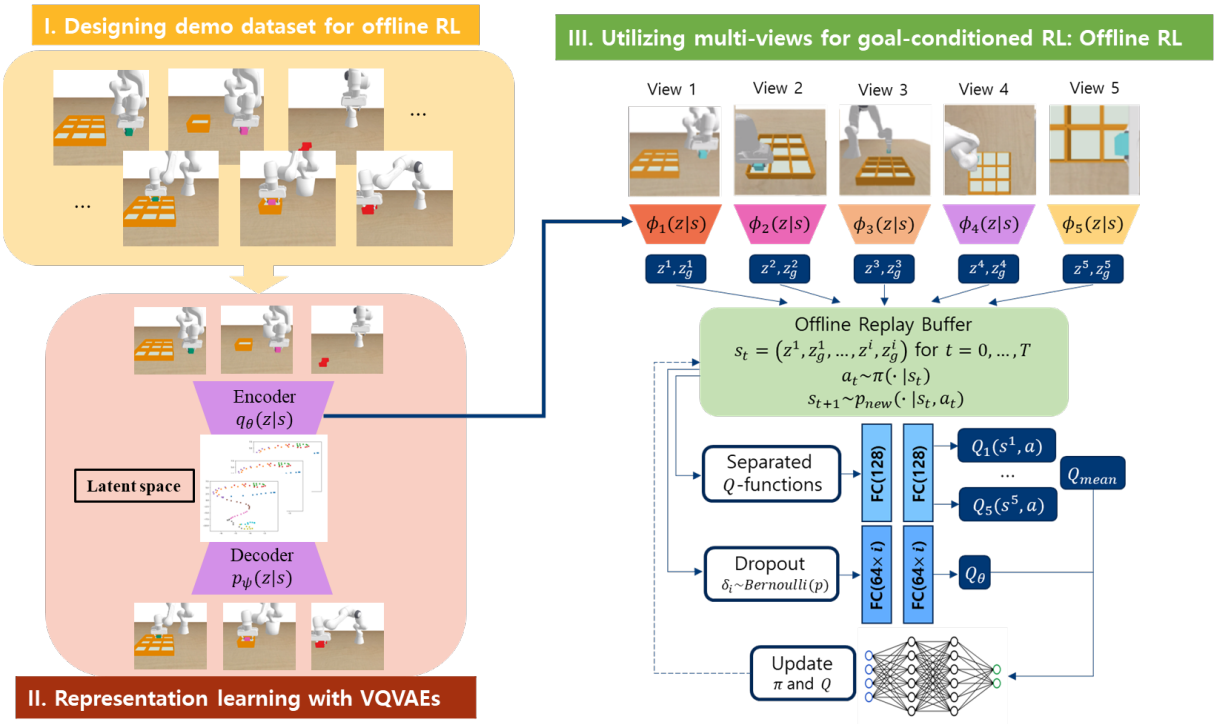



**Figure 2.** Full scheme of our MURM algorithm, comprising three distinct phases.

### *4.1. Designing Demo Dataset for Offline RL*

In phase 1, we collect a demo dataset for offline replay buffer, creating a dataset $\mathcal{D} = \{\tau_1, \tau_2, \tau_3, \ldots\}$ consisting of trajectories that achieve final outcomes $s_T$ for the given task. To provide our agent with more variety of information when interacting with the environment, we use a noisy expert policy which allows the robot to execute random actions during demonstrations. A noisy expert policy $\pi_{demos}(a_i|s,g) = \pi^*_{demos}(a_i|s,g) + k \cdot \mathcal{N}(0, 0.1)$ where $-1 \leq a_i \leq 1 \quad \forall i \quad$ and $0 \leq k \leq 1$ is used. Our demonstrated policy achieves the goal in roughly 75% of attempted episodes and we collect 1000 episodes for each task, with each episode lasting 100 timesteps. All of our experiments are conducted within a Pybullet simulated environment [32] where we develop and test three distinct tasks as illustrated in Figure 3.

Three tasks implemented for our study involve robotic manipulation within simulated environments, specifically focusing on: (1) placing an object into one of several boxes on a table, (2) placing an object into a goal box located at a random position, and (3) picking up objects with various shapes that are randomly scattered throughout the environment. The success criteria for task 1 require the robot to place the object within a 3 cm radius of the center of the goal box (a square box with 10 cm width) that is chosen randomly among nine boxes at the initial timestep of episodes. Task 2 is similar to task 1, except a single goal box is randomly placed in a 3-dimensional space and the robot should place the object within a 5 cm radius of the center of the goal box. Lastly, for task 3, success is defined when the robot picks up the object 5 cm above the initial ground position.

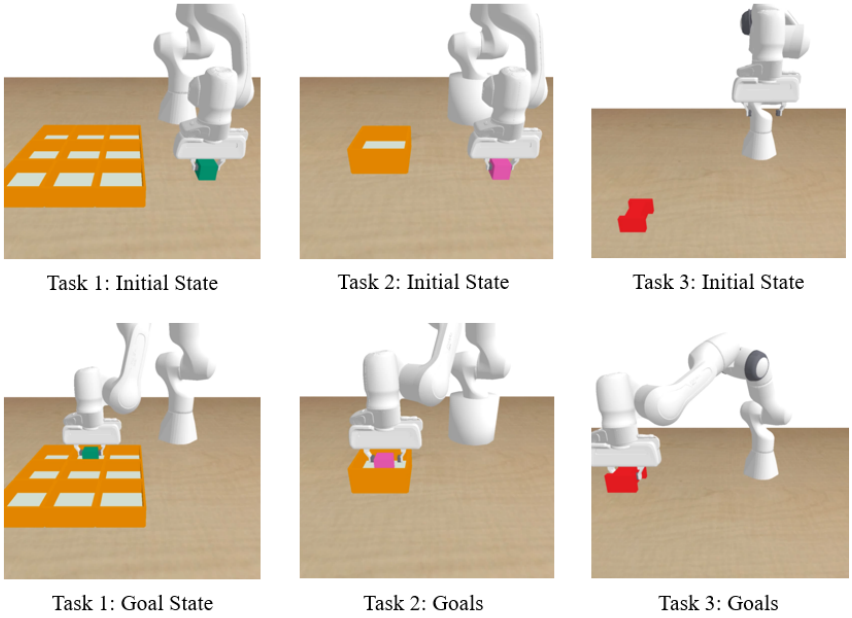
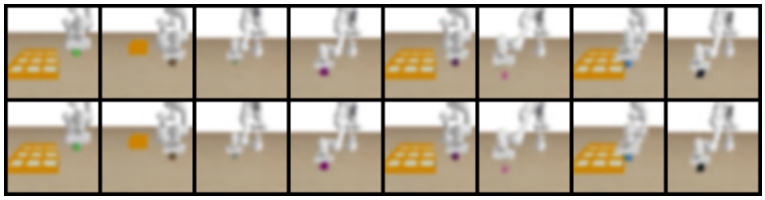



**Figure 3.** Simulation environments for three tasks designed for experiments. First row shows the initial state of the environment, whereas the second row shows the final state where the goal is achieved.

### *4.2. Representation Learning with VQVAE*

The high dimensionality of image observations causes tremendous difficulty in using them directly as input for training Rl agents. As a solution, we learn a lower-dimensional latent space $p(z_t|s_t)$ by training a generative model VQVAE. There are several models that can be used, such as VAEs [10] and CC-VAEs [11], but we use VQVAE [28] for its compact representation compared to other models. To train the model, we collect $48 \times 48 \times 3$ images by running demos, as in Section 4.1, and train separate VQVAE models for each camera perspective. In our work, we use five distinct viewpoints for experiments: global-view, adjacent-view, side-view, top-view, and active-view; whereas the global-, adjacent-, side-, and top-views remain fixed relative to the ground, the active-view is captured by a camera attached to the robot's end-effector, which moves along with the robot. Throughout our experimental setup, we experiment on these viewpoints to determine their effectiveness and efficiently utilize them to achieve better performance in solving the tasks. Some examples of reconstructed images for each viewpoint are shown in Figure 4 where the first row of each image pair shows the original dataset images, and the second row displays the images reconstructed by our trained model after training.

We train separate VQVAE models for each perspective, with each model trained for 1000 epochs with reconstruction errors below 0.00003 and a total loss of 0.0001, with training times of approximately 8 h for each model. Using the VQVAE model, we can encode images from the environments into 720-dimensional latent embeddings, which are later used as states for reinforcement learning.



(**a**)

(**b**)

(**c**)

(**d**)

(**e**)

**Figure 4.** Examples of reconstructed images obtained for each viewpoints after training VQVAE model for 1000 epochs. (**a**) Global-view, (**b**) adjacent-view, (**c**) top-view, (**d**) side-view, (**e**) active-view. The first row of each pair shows the original images, whereas the second row displays the corresponding reconstructed images produced by the model. We trained five models for each distinct viewpoint: global-view, adjacent-view, top-view, side-view, and active-view.

### *4.3. Utilizing Multi-Views in Goal-Conditioned RL in Offline RL Settings*

In this phase, our goal is to learn a good goal-conditioned policy by utilizing images from multiple viewpoints with offline RL. As our baseline RL algorithm, we implement the IQL algorithm, which has shown strong performance in offline RL training compared to other algorithms. The algorithm is modified to effectively utilize images from multiple viewpoints, which are described in detail in this section.

To represent the system's state, we utilize latent vectors obtained from $i$ number of viewpoints by concatenating the current latent states $z^1, \ldots, z^i$ with the corresponding latent goals $z_g^1, \ldots, z_g^i$ to form a single-state vector $s = (z^1, z_g^1, \ldots, z^i, z_g^i)$; whereas an alternative state vector $s = (z^1, z^2, \ldots, z_g^1, z_g^i)$ is also viable, we use the former expression since it results

in slightly superior performance. Latent states and goals are encoded with VQVAE models trained in Section 4.2. To produce latent goals, we place the object and the end-effector of the robot in desired goal positions by solving inverse kinematics within the simulation environment. We compute the reward by denoting the full reward function as follows:

$$r(z^1, z_g^1, \ldots, z^i, z_g^i) = \sum_{k=1}^{i} \alpha_i - i \tag{4}$$

where $\alpha_i$ are defined with the latent distance as in [10].

$$\alpha_i \begin{cases} 0 & \text{otherwise} \\ 1 & \text{if } ||z^i - z_g^i||_A < \epsilon_{\alpha_i} \end{cases} \tag{5}$$

The parameters $\epsilon_{\alpha_i}$ represent a fixed threshold of squared distance in the latent space, which is used to reward states that are within the threshold distance from the latent goal in latent space, whereas it is possible to use the concatenated state vector directly for reinforcement learning; in order to effectively utilize information from each of the image views provided by different cameras, we propose two effective methods that can be implemented: the **dropout** method and the **separated Q-functions** method.

***Dropout method***: The dropout method aims to improve robustness in the absence of camera views by randomly masking out some of the states in the state vector during training. For research dealing with multi-sensory fusions, removing a part of sensor information has been shown to be effective and in standard RL, the dropout method [23] showed its effectiveness in multi-view settings. In our GCRL settings, from the given state of $i$ number of viewpoints $s = (z_t^1, z_g^1, \ldots, z_t^i, z_g^i)$, some of the components $(z_t^k, z_g^k)$ are removed and replaced by zero components with $\delta_i \sim$ Bernoulli probability$(p)$. We perform this process after sampling batches of $s_t$ and $s_{t+1}$. Since we do not want to consider the case where all cameras are obscured, we define this case as a situation where none of the components are replaced. Using the basis from the IQL algorithm, the state vector after the dropout process is used to fit the value function and Q-function by performing a number of gradient updates alternating between the following losses:

$$L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(\hat{Q}_{\hat{\theta}}(s,a) - V_\psi(s))] \tag{6}$$

$$L_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2] \tag{7}$$

Then stochastic gradient descent is performed for the following loss:

$$L_\pi(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[exp(\beta(Q_{\hat{\theta}}(s,a) - V_\psi(s))) \log \pi_\phi(a|s)] \tag{8}$$

For all steps, the clipped double Q-learning [33] method is used, where a minimum of two Q-functions for the value function and policy updates are taken. We use a network architecture with two 128-dimensional fully connected (FC) layers for the Q-function network and value-function network and four 128-dimensional FC layers for the policy network. This same network architecture is used for the single-view baseline as well.

***Separated Q-functions method***: In the separated Q-functions method, we use separate q-functions for each of viewpoints $(z_t^k, z_g^k)$ in the concatenated state vector $s = (z_t^1, z_g^1, \ldots, z_t^i, z_g^i)$. Similarly with the base from the IQL algorithm, the value function and Q-function are fit with a number of gradient updates from following losses, respectively:

$$L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(\hat{Q}_{mean}(Q_1, \ldots, Q_i) - V_\psi(s))] \tag{9}$$

$$L_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}[(r(s,a) + \gamma V_\psi(s') - Q_{mean}(Q_1, \ldots, Q_i))^2] \tag{10}$$

Both Q-function $Q_{mean}$ and target Q-function $\hat{Q}_{mean}$ are defined as the mean of separate Q-functions for each state viewpoint and are formalized as follows:

$$Q_{mean}(Q_1, \ldots, Q_i) = \frac{1}{i} \sum_{k=1}^{i} (Q_k(s_k, a)) \tag{11}$$

where $i$ is the number of camera views. With loss from (9), we update all Q-function inputs for $Q_{mean}$, $Q_1$ to $Q_i$ with the MSE loss. We maintain the clipped double Q-learning method by taking a minimum of two mean Q-functions for the value function and policy updates. For each separated Q-functions, we use network architecture with two 128-dimensional FC layers, and for the value-function and policy network, the same network architecture as the dropout method is used.

Our overall methods for both methods are summarized in Algorithm 1. In addition to the implementation of two methods, we also use future hindsight experience replay (HER) [34] to relabel our goals with 15% probability.

**Algorithm 1** MURM

**Require:** Dataset $\mathcal{D}$, policy $\pi(a|s)$, Q-function $Q(s)$, RL algorithm $\mathcal{A}$, replay buffer $\mathcal{R}$, state $s = (z^1, z_g^1, \ldots, z^i, z_g^i)$, state $s_i = (z^i, z_g^i)$

Collect demos of $\{\tau_1, \tau_2, \ldots, \tau_n\}$ from noisy expert
Learn state-encoders $\phi_i(z|s)$ for each $i$ viewpoint
Change states with raw images to latent states
**if** *method = Separated Q-functions* **then**
    $Q \leftarrow Q = \frac{1}{n} \sum_{i=1}^{n} (Q_i(s_i, a))$
**end if**
Initialize $\pi$ and $Q$ by $\mathcal{A}$
**for** 1, ... , $N_{episodes}$ **do**
    Sample goals for used $i$ views: $z_g, \cdots, z_g^i$
    **for** $t = 0, ..., T$ **do**
        sample $a_t \sim \pi(\cdot|s_t)$
        sample $s_{t+1} \sim p_{new}(\cdot|s_t, a_t)$
    **end for**
    Store trajectory $(z_0, a_0, \ldots, z_T)$ in replay buffer $\mathcal{R}$
    **if** *method = Dropout* **then**
        with $\delta_i \sim$ Bernoulli($p$), $(z_t^k, z_g^k) = (0, \ldots, 0)$
    **end if**
    Update $\pi$ and $Q$ with sampled batches using $\mathcal{A}$
**end for**

## 5. Experimental Evaluation

Our experiments aim to investigate the effectiveness of MURM on robotic manipulation tasks that involve visual information. For evaluation, we conducted experiments on three tasks described in Section 4.1. All tasks involve controlling a Panda Franka robot with 4 degrees of control and due to the complexity of action space when including rotational motion control, we used three dimensions of end-effector position control in Euclidean space, as well as an additional degree of control to open and close the gripper. All of our experiments are conducted within the simulation environment Pybullet with hardware of RTX 3080 ti GPU and AMD 5900X CPU.

### *5.1. Experimental Setups*

Based on the IQL algorithm, we first determine a suitable network architecture to use for GCRL training, which is closely related to the MLP (multi-layer perceptron), the number of neurons in the FC layers. We conducted experiments for all three tasks within the experimental setups, and as the optimized parameter results were consistent across all tasks, we present representative results specifically for task 1. To observe the effect of the

number of neurons, we experimented with a single-view baseline using global-view images and an epsilon value of 2.0. The goal-conditioned policy is trained on the offline dataset for 500 epochs, with each epoch consisting of 1000 training iterations where the graph results converged. To verify the results, evaluation episodes were collected 10 times every 5 epochs. The success rates are calculated using a moving average for every 100 epochs. Before reaching 100 epochs, the calculation is based on the results obtained from the 0 epoch up to the current epoch. The results for task 1 are shown in Table 1. Considering both the success rate and time consumption(relative value with 128 neurons as the standard), we form our FC layers with 128 neurons per viewpoint for our subsequent experiments.

**Table 1.** The success rates with its standard deviation for varying the number of neurons are presented in the second column, whereas the times required to train with different neuron counts are displayed in the last column, with relative values to the standard of 128 neurons. All results are averaged across 3 seeds.

| Neurons (MLP) | Success Rate (%) | Time Consumption |
|---|---|---|
| 32 | $29.85 \pm 4.43$ | 0.52 |
| 64 | $32.19 \pm 3.0$ | 0.66 |
| 128 | $\mathbf{33.14} \pm 2.73$ | **1.0** |
| 256 | $30.73 \pm 3.13$ | 2.64 |

As our next process, we search for the best epsilon values for every single-view baseline in the three tasks. For the network architecture, we use two FC layers for the Q-function and value-function networks and four FC layers for the policy network. As with the previous step, we conduct experiments on task 1 using global-view images for the single-view baseline. The success rates during training are shown in Figure 5. In Table 2, the overall mean of standard deviation during training is shown in the second row, and the final success rate results with their standard deviations in the third row.

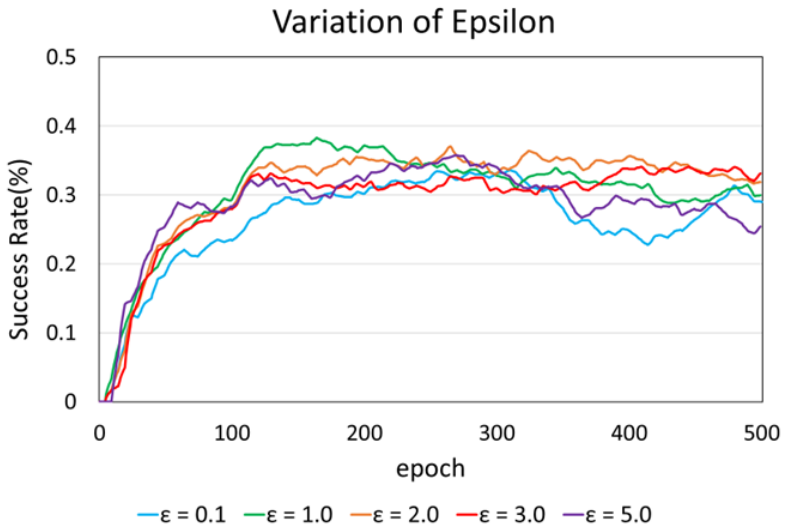



**Figure 5.** Learning curves of global-view images with variation in epsilon from 0.1 to 5.0. The training is performed only with offline reinforcement learning for 500 epoches. All experiments are averaged across 3 seeds.

**Table 2.** Mean values for standard deviations and final success rate results with variation in epsilons. All results averaged across 3 seeds.

| Epsilon Values | 0.1 | 1.0 | 2.0 | 3.0 | 5.0 |
|---|---|---|---|---|---|
| **Std ($\sigma$) mean** | $\pm$5.64 | $\pm$2.70 | $\pm$2.97 | $\pm$**1.94** | $\pm$2.86 |
| **Success Rate (%)** | 29.04 $\pm$ 0.89 | 29.97 $\pm$ 2.08 | **34.49** $\pm$ 3.08 | 33.14 $\pm$ 2.73 | 27.92 $\pm$ 3.43 |

The results show that for epsilon values that are too short ($\epsilon = 0.1$) or long ($\epsilon = 5.0$), latent distances lead to lower success rates and increased difficulty in convergence as indicated by high mean values of standard deviations. We postulate that this may be due to these values failing to provide appropriate rewards during RL training. It can be observed that the epsilon value of $\epsilon = 3.0$ demonstrates the greatest strength in convergence, whereas the highest success rate is obtained with an $\epsilon = 2.0$. Therefore, for our subsequent experiments, we experiment with the epsilon values within 2.0 and 3.0 and select the best-performing value for evaluation.

We next investigate the success rates for different viewpoints in single-view baselines. We use five distinct viewpoints and for all views, optimized network settings with epsilon values are used. The overall results are shown in Table 3. It can be observed that for all tasks, success rates exhibit a significant variation across different views but adjacent-view and global-view provide the best results overall. We conjecture that the views with the highest success rates provide the most effective visual information for solving the tasks, and using these views together will provide the best input data for our multi-view utilization method.

**Table 3.** Single-view baseline results for all three tasks with five different camera viewpoints. All experiment results are averaged across 3 seeds.

| Viewpoints | Success Rate (%) | | |
|---|---|---|---|
| | Task 1 | Task 2 | Task 3 |
| Global-view | **34.49** | **46.63** | **15.74** |
| Adjacent-view | 33.76 | 39.50 | 14.65 |
| Top-view | 18.84 | 38.65 | 0.17 |
| Side-view | 17.99 | 33.86 | 12.74 |
| Active-view | 12.51 | 8.68 | 3.63 |

### *5.2. Results and Analysis*

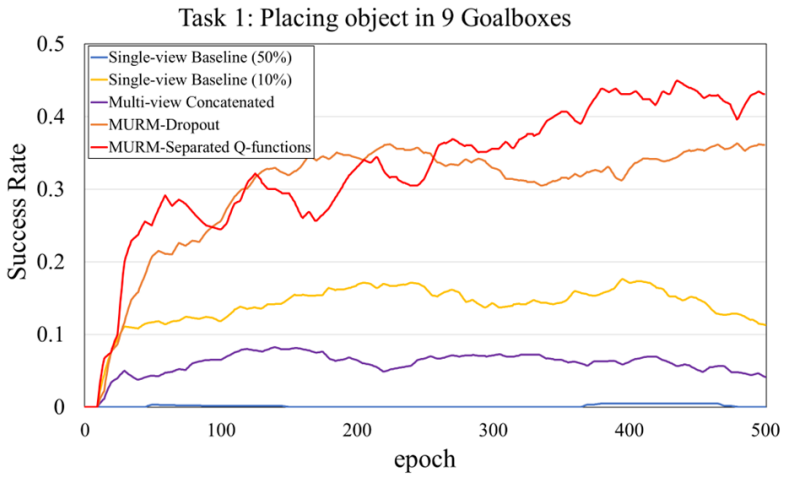
We implement the MURM method with optimized epsilon values and viewpoints of best success rates from single-view baselines. For example, in task 1, we use global-view and adjacent-view images since they have better success rates compared to other viewpoints. Offline RL training is conducted as described in Section 5.1. However, during the evaluation process, we randomly mask out image data to assess the robustness of the trained RL model to the absence of camera views while performing tasks. For single-view baselines, the image data are masked out with 10% and 50% probabilities before being used by our trained RL model. For cases using multiple viewpoints, the image data are masked out as same as in the dropout method but with a 50% probability. The experiments with single-view baselines are conducted with the parameters and viewpoints that give the best success rates in Section 5.1 process. Multi-view concatenation method, where the state is simply formed by concatenating the latent vectors of multiple viewpoints, is also experimented and evaluated in the same way as the MURM methods. Figure 6 depicts the training performances of the three tasks achieved through offline reinforcement learning. The corresponding success rates for each task and method are shown in Table 4. The success rate defined for the three tasks is the same as in the demonstrations. Specifically, placing an

object into the goalbox for tasks 1 and 2 and picking up the object above the ground for task 3.

**Table 4.** Success rates after offline reinforcement learning. All experiments are averaged across 3 seeds and our method significantly outperforms in all tasks.

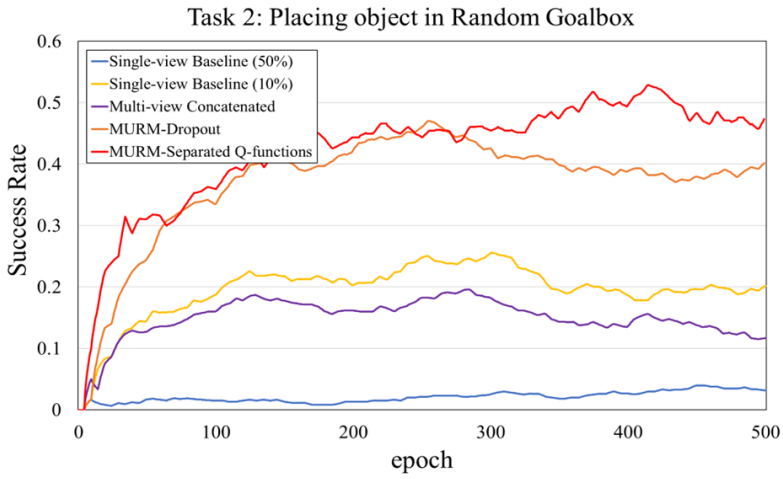
| Methods | Success Rate (%) | | |
|---|---|---|---|
| | Task 1 | Task 2 | Task 3 |
| Single-view Baseline (50%) | 0.0 | 3.17 | 0.0 |
| Single-view Baseline (10%) | 11.35 | 20.1 | 7.85 |
| Multi-view Concatenated | 4.22 | 11.65 | 3.93 |
| MURM-Dropout | 36.07 | 40.2 | 9.67 |
| MURM-Separated Q-functions | **38.35** | **43.96** | **14.88** |

As can be seen in Figure 6, MURM methods outperform baselines with a large performance gap. In all tasks, the success rates for the single-view baseline with 50% masking out data and multi-view concatenation methods show poor performance, indicating that the policy cannot be trained using these methods. The single-view baseline with 10% of the masking out data shows a slight improvement but its success rates are significantly decreased compared to the case without noise, due to its poor robustness. However, when the policy is trained with the dropout method, the policy achieves success rates of 36%, 40%, and 10% in the three tasks, respectively. Moreover, the policy achieves success rates of 38%, 44%, and 15% with the separated Q-functions method. Notably, with the best-performing separated Q-functions method, the success rates significantly improve by +27%, +23.86%, and 7.03% even when evaluated in more noisy conditions.

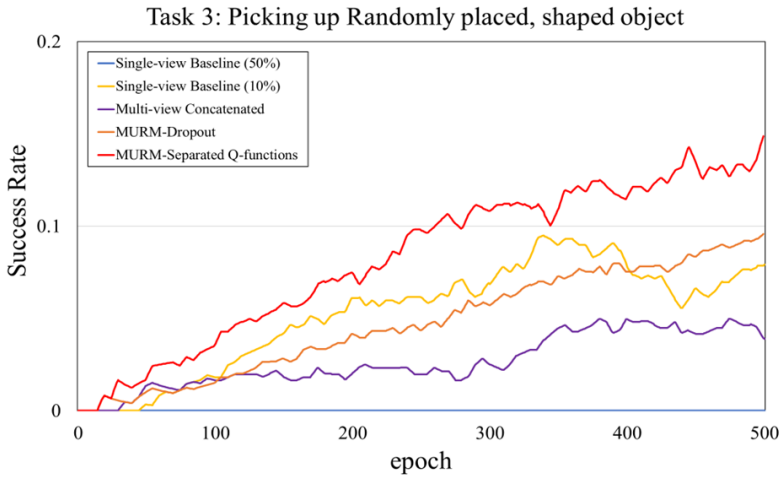



(**a**)

**Figure 6.** *Cont.*




(**b**)




(**c**)

**Figure 6.** Final training results. (**a**) Training results for task 1: placing an object into one of several boxes on a table. (**b**) Training results for task 2: placing an object into a goal box located at a random position. (**c**) Training results for task 3: picking up objects with various shapes that are randomly scattered throughout the environment.

We also compare the time consumption for each method and the results are shown in Table 5. Time consumption is calculated as a relative value with the single-view baseline as a standard. It can be seen that the time required for training the MURM-dropout method is almost the same as the single-view baseline, and there is a 1.5 times increase for the MURM-separated Q-functions method, which shows the efficiency of our method. However, as more viewpoints are used, the separated Q-functions method requires more networks to be trained, whereas the dropout method requires heavier networks to be trained, albeit with only a minor increase in success rate. If a single viewpoint is added, the time consumption for both methods increases by approximately half the amount of time required for the single-view baseline. Therefore, it is important to determine an appropriate number of viewpoints that can enhance the success rate without compromising time consumption.

Furthermore, we also experiment with whether combining the most successful single-view baselines with other poorly performing baselines can achieve better results than most successful single-view baselines. The experiments are conducted in task 1 and we use a separated Q-functions method, which outperforms other methods. The final success rate results are shown in Table 6.

**Table 5.** Time consumption for single-view baseline, MURM-dropout method, and MURM-separated Q-functions method. The time needed for training with MURM-dropout method is almost the same as single-view baseline and a 1.5 times slight increase for MURM-separated Q-functions method. However, if more viewpoints are used, time consumption also increases by approximately 0.5 times for both methods.

| Methods | Time Consumption (If Add a Viewpoint) |
|---|---|
| Single-view Baselines | 1.0 |
| MURM-Dropout | 1.05 (+0.46) |
| MURM-Separated Q-functions | 1.55 (+0.5) |

**Table 6.** Success rates of MURM when other poor-performing different viewpoints (side, top, active) are combined with the most successful viewpoint. We experiment on task 1 where global-view showed the most successful result. All results averaged across 3 seeds.

| Viewpoint Added | Success Rate (%) |
|---|---|
| Top-view | 30.23 (−8.12) |
| Side-view | 32.87 (−5.48) |
| Active-view | 23.07 (−15.28) |

The experimental results show that simply adding a viewpoint leads to improved performance compared to the single-view baseline. However, it can be seen that the increase in success rate is directly related to the success rate of the specific viewpoint used solely, indicating that the most effective implementation of MURM methods is to combine the best-performing single-view baselines.

We hypothesize that MURM methods can solve the tasks more successfully because of two reasons. First, the additional viewpoint can provide state information that is not present in the single view, which may be essential in reaching the goal. For instance, in task 2, the robot mostly failed to solve the task with a single view because the position of the goal box appears to be similar in its 2-dimensional view despite being quite different in 3-dimensional space. However, by adding a new viewpoint where the difference can be distinguished clearly, the robot was able to achieve the goal more successfully. Secondly, when noise masks out an image at time $t$, it crucially affects single-view baselines since the trained model suddenly loses input data to use.However, MURM can still use image data provided by other cameras which allows the trained model to follow up the task.

### *5.3. Ablation Experiments*

*Demo Dataset Modification*: In this section, we investigate whether increasing the number of demo episodes or changing the proportion of noisy demo episodes can improve the agent's performance on the tasks. Specifically, we conduct experiments on task 1, using various numbers of demos ranging from 250 to 3000 and different proportions of noisy demonstrations, with proportion $k$ ranging from 0% to 100%, and with 1000 episodes as a baseline. All cases are trained for the same number of iterations and go through the same evaluation process as in Section 5.1. Furthermore, we perform experiments on MURM using the separated Q-functions method, which yields the best performance. Table 7 shows the overall results.

**Table 7.** Success rates for MURM with separated Q-functions method. First table shows the results when number of demo episodes are increased and second table shows the results when the proportion of noisy demos $k$% is changed.

| **Demo Episodes** | 250 | 500 | 1000 | 2000 | 3000 |
|---|---|---|---|---|---|
| **Success Rates (%)** | 29.11 | 31.22 | **38.35** | 35.84 | 34.75 |
| **Noisy Experts ($k$%)** | 0% | 25% | 50% | 100% | |
| **Success rates (%)** | 26.57 | 31.85 | 33.73 | **38.35** | |

The results demonstrate that adding more demonstrations can improve the agent's ability to solve the task overall. However, it is also observed that there is a limit to the benefits of adding demonstrations, beyond which the performance gain diminishes. Furthermore, it can be observed that decreasing the proportion of noisy demonstrations in the offline RL buffer leads to inferior performance. We hypothesize that when demos performed by noisy experts are included in offline training, the agent has more information about interactions with the environment which leads to better exploitation. Thus, our results suggest that our method will show its best performance when the number of expert trajectories is carefully selected for noisy experts.

## 6. Conclusions

We have presented the Multi-View Unified Reinforcement Learning for Manipulation (MURM) algorithm, which effectively fuses information from multiple cameras and can train the robot to solve complex tasks. Even in the cases where conventional single-view baselines (SOTA) and the multi-view concatenated method, which simply adds more camera views to the SOTA method, fail to solve the task, our separated Q-functions and dropout method show successful performances. Our method is validated through three designed tasks and experimental results show that the effective utilization of multi-views allows for a richer observation of the state and improves robustness to noise and occlusion in camera views. Specifically, whereas the existing SOTA method shows poor performance with 0%, 3%, and 0% of success rates in three tasks, respectively, our suggested dropout method and separated Q-functions method achieve 36%, 40%, and 10% and 38%, 44%, and 15% respectively, show an insight into reinforcement learning-based robotic manipulation with visual information. Lastly, to leverage our method effectively, it is crucial to select viewpoints that provide the necessary information to solve the task and complement other viewpoints being used.

In future work, we may combine MURM with human language-conditioned robotic manipulation where the underlying state can be formed as multi-modal inputs with images and human languages. By incorporating multi-modal inputs with human language, agents will be able to relate human languages to their perceptions and actions and generalize abstract concepts to unseen entities just like humans. Furthermore, integrating our work with hierarchical reinforcement learning which decomposes the long-horizon decision-making process into simpler subtasks will enable us to tackle more complex and longer-horizon tasks. Increasing the degree of freedom in the task space, such as incorporating rotational degrees of freedom, can enable the robot to handle more complex tasks as well, an intriguing avenue for further exploration. Additionally, to enhance the latent feature representation from VQVAEs and effectively utilize mutual information from multi-views, exploring the addition of a transformer network with an attention mechanism that incorporates multi-views into a new representation would be an intriguing research direction. Lastly, to perform real-world training in a reasonable amount of time, we believe the sample efficiency needs to be improved even further and differences between simulation

and real space must be solved. In the realm of Sim2Real transfer, the challenges encountered in training image representation networks and reinforcement learning arise due to the divergence between simulated images, which lack real-world complexities like lighting and blur, and actual image samples. As a future work, we aim to enhance sample efficiency and solve differences between simulation and real space, finally implementing our approach on a real robot for Sim2Real transfer.

**Author Contributions:** Conceptualization, S.J.; methodology, S.J; software, S.J.; formal analysis, S.J. and H.J.; investigation, S.J.; writing—original draft preparation, S.J. and H.J.; writing—review and editing, H.J; supervision, H.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are available on request to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| GCRL | Goal-Conditioned Reinforcement Learning |
| RL | Reinforcement Learning |
| VQVAE | Variational Autoencoders |
| MURM | Multi-view Unified Reinforcement Learning for Manipulation |
| MDP | Markov Decision Process |
| IQL | Implicit Q-Learning |
| HER | Hindsight Experience Replay |
| SOTA | State-of-the-art |
| MLP | Multi-Layer Perceptron |

## References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
2. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
3. Chen, P.; Lu, W. Deep reinforcement learning based moving object grasping. *Inf. Sci.* **2021**, *565*, 62–76. [CrossRef]
4. Su, H.; Hu, Y.; Li, Z.; Knoll, A.; Ferrigno, G.; De Momi, E. Reinforcement learning based manipulation skill transferring for robot-assisted minimally invasive surgery. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 2203–2208.
5. Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv* **2018**, arXiv:1802.09464.
6. Liu, M.; Zhu, M.; Zhang, W. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv* **2022**, arXiv:2201.08299.
7. Hansen-Estruch, P.; Zhang, A.; Nair, A.; Yin, P.; Levine, S. Bisimulation Makes Analogies in Goal-Conditioned Reinforcement Learning. *arXiv* **2022**, arXiv:2204.13060.
8. Cong, L.; Liang, H.; Ruppel, P.; Shi, Y.; Görner, M.; Hendrich, N.; Zhang, J. Reinforcement Learning with Vision-Proprioception Model for Robot Planar Pushing. *Front. Neurorobotics* **2022**, *16*, 829437 [CrossRef]
9. Qian, Z.; You, M.; Zhou, H.; He, B. Weakly Supervised Disentangled Representation for Goal-conditioned Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2202–2209. [CrossRef]
10. Nair, A.V.; Pong, V.; Dalal, M.; Bahl, S.; Lin, S.; Levine, S. Visual reinforcement learning with imagined goals. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
11. Nair, A.; Bahl, S.; Khazatsky, A.; Pong, V.; Berseth, G.; Levine, S. Contextual imagined goals for self-supervised robotic learning. In Proceedings of the Conference on Robot Learning, Cambridge, MA, USA, 16–18 November 2020; pp. 530–539.
12. Khazatsky, A.; Nair, A.; Jing, D.; Levine, S. What can i do here? learning new skills by imagining visual affordances. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 14291–14297.
13. Laskin, M.; Srinivas, A.; Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 5639–5650.

14. Fang, K.; Yin, P.; Nair, A.; Levine, S. Planning to practice: Efficient online fine-tuning by composing goals in latent space. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 4076–4083.
15. Yang, R.; Zhang, M.; Hansen, N.; Xu, H.; Wang, X. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv* **2021**, arXiv:2107.03996.
16. Chen, B.; Abbeel, P.; Pathak, D. Unsupervised learning of visual 3d keypoints for control. In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 1539–1549.
17. Hu, F. Mutual information-enhanced digital twin promotes vision-guided robotic grasping. *Adv. Eng. Inform.* **2022**, *52*, 101562. [CrossRef]
18. Gupta, D.S.; Bahmer, A. Increase in mutual information during interaction with the environment contributes to perception. *Entropy* **2019**, *21*, 365. [CrossRef] [PubMed]
19. Kroemer, O.; Niekum, S.; Konidaris, G. A review of robot learning for manipulation: Challenges, representations, and algorithms. *J. Mach. Learn. Res.* **2021**, *22*, 1395–1476.
20. Jangir, R.; Hansen, N.; Ghosal, S.; Jain, M.; Wang, X. Look Closer: Bridging Egocentric and Third-Person Views With Transformers for Robotic Manipulation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3046–3053. [CrossRef]
21. James, S.; Wada, K.; Laidlow, T.; Davison, A.J. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 13739–13748.
22. OpenAI, O.; Plappert, M.; Sampedro, R.; Xu, T.; Akkaya, I.; Kosaraju, V.; Welinder, P.; D'Sa, R.; Petron, A.; Pinto, H.P.d.O.; et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv* **2021**, arXiv:2101.04882.
23. Akinola, I.; Varley, J.; Kalashnikov, D. Learning precise 3d manipulation from multiple uncalibrated cameras. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 4616–4622.
24. James, S.; Davison, A.J. Q-attention: Enabling efficient learning for vision-based robotic manipulation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 1612–1619. [CrossRef]
25. Seo, Y.; Kim, J.; James, S.; Lee, K.; Shin, J.; Abbeel, P. Multi-View Masked World Models for Visual Robotic Manipulation. *arXiv* **2023**, arXiv:2302.02408.
26. Kullback, S.; Leibler, R.A. On information and sufficiency. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]
27. Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; Lerchner, A. Beta-vae: Learning basic visual concepts with a constrained variational framework. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
28. Van Den Oord, A.; Vinyals, O. Neural discrete representation learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
29. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1889–1897.
30. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
31. Kostrikov, I.; Nair, A.; Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv* **2021**, arXiv:2110.06169.
32. Coumans, E.; Bai, Y. Pybullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016. Available online: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=E_82W3EAAAAJ&citation_for_view=E_82W3EAAAAJ:hqOjcs7Dif8C (accessed on 16 August 2023).
33. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
34. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.