Farm Technology Group

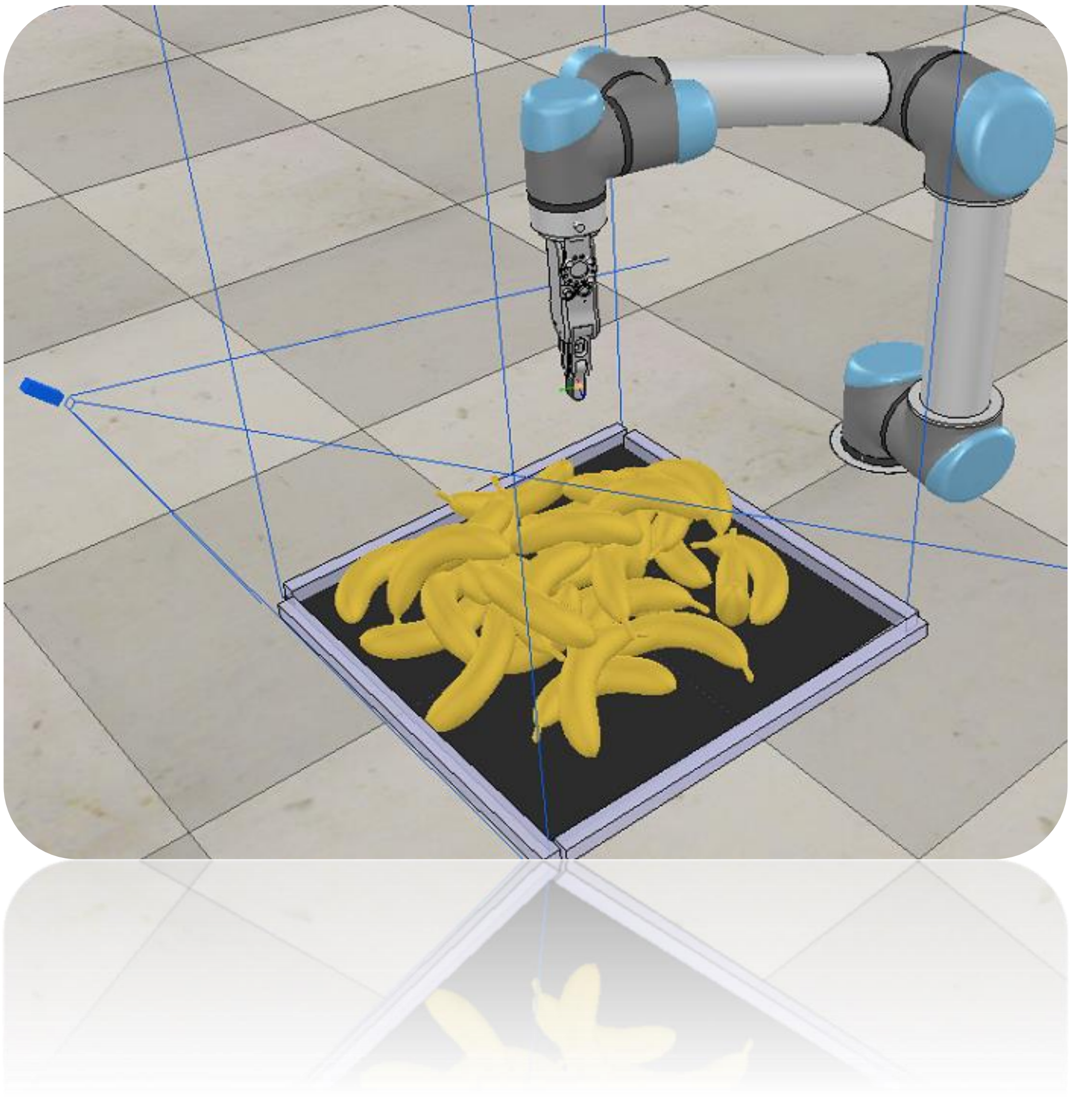
MSc Thesis Farm Technology

Learning Pushing and Grasping for Agro-Food Clutter Clearance

An Exploration of Deep Reinforcement Learning for Food Manipulation

Gregory Bandy

22-10-2021



WAGENINGEN UNIVERSITY & RESEARCH

marel

Learning Pushing and Grasping Actions for Agro-Food Clutter Clearance

An Exploration of Deep Reinforcement Learning for Food Manipulation

Name course	:	MSc Thesis Farm Technology
Number	:	FTE-82000
Study load	:	36 credits
Date	:	22-10-2021
Student	:	Gregory Bandy
Registration number	:	1043870
Study programme	:	MSc Biosystems Engineering
Supervisor(s)	:	dr. R (Rekha) Raja PhD
	:	dr. A (Ali) Leylavi Shoushatari
Examiner	:	dr. GW (Gert) Kootstra
Group	:	Farm Technology Group Droevendaalsesteeg 1 (Building 107) 6708 PB Wageningen T: +31 (317) 48 29 80 E: office.fte@wur.nl

DISCLAIMER

Abstract

The food processing industry is becoming increasingly automated and mechanized to increase food production for larger, future populaces. Despite this, there are multiple applications where current levels of automation do not suffice. At the moment, only human labour can meet the demands imposed by product variation and complexity of agro-food products. This dependence on human labour is becoming an increasing issue because of challenges related to repetitive motion disorders, points of contamination, and labour shortages, which are all drivers to tackle the shortcomings faced by automation in food processing. One food processing step where current levels of automation do not suffice is the task of bin picking and placing agro-food products in cluttered scene for packaging or further processing tasks. This task is deceptively hard not only because of product variations but overlapping objects make it difficult for a robot to identify the best method for picking up. To handle these complexities, skilled robotic manipulation is necessary. Skilled manipulation uses both non-prehensile (pushing) and prehensile (grasping) actions: pushing can rearrange clutter to create space for fingers while grasping can displace objects making pushing more precise. In this thesis, deep reinforcement learning was used to learn partially skiled manipulation for the purpose of picking up agro-food products in cluttered environments. The deep learning method performance was found to improve when the reward was removed from the secondary motion primitive for pushing and when a new set of training objects were used to train the agent.

Table of Contents

List of Abbreviated Terms

AE	Action Efficiency
AFV	Agro-Food Variant Agent
BA	Block Adversarial Test
BCA	Block Crop Adversarial Test
CNN	Convolutional Neural Network
DQN	Deep Q Network
FCN	Fully Convolutional Network
GPU	Graphics Processing Unit
GS	Grasp Success Rate
GtP	Grasp to Push Ratio
HCT	Higher Change Threshold Agent
MDP	Markov Decision Process
PNR	Pushing No Reward Agent
PLR	Pushing Low Reward Agent
ReLU	Rectified Linear Unit Activation Function
SCT	Small Colour Training Object Agent

SGD	Stochastic Gradient Descent
TD	Temporal Difference
VPG	Visual Pushing for Grasping Method

List of Symbols

$\boldsymbol{\alpha}$	Learning rate of the update rule
$\boldsymbol{\beta}$	Rotation of end-effector in xy-plane
$\boldsymbol{\gamma}$	Discount factor
$\boldsymbol{\delta_t^{a_t}}$	Temporal difference for motion primitive action at time step t
$\boldsymbol{\epsilon}$	Exploration probability in ϵ-greedy policy
$\boldsymbol{\theta}$	Parameter set of neural network function approximation
$\boldsymbol{\theta^-}$	Parameter set of target network function approximation
$\boldsymbol{\pi}$	Policy that directs agent behaviour
$\boldsymbol{\pi'}$	All other possible policies
$\boldsymbol{\tau}$	Manually set threshold for image change detection
$\boldsymbol{\psi}$	Primitive motion parameterization; push or grasp
$\boldsymbol{\nabla L(\theta)}$	Global gradient of loss computation
$\boldsymbol{A(s)}$	Set of all possible actions
$\boldsymbol{a}$	Action in reinforcement learning formulation
$\boldsymbol{b}$	Bias added to weights of neuron inputs
$\boldsymbol{G_t}$	Discounted expected reward at time t
$\boldsymbol{l}$	Layer in a DenseNet block
$\boldsymbol{L}$	Total number of layers in a DenseNet block
$\boldsymbol{L(\theta)}$	Loss computation
$\boldsymbol{m}$	Mini batches for batch learning
$\boldsymbol{n}$	Neuron input connections
$\boldsymbol{P_{s,s_t}^a}$	Transition probabilities for action a given s all possible next states s'
$\boldsymbol{p}$	Pixel of the heightmap image representation
$\boldsymbol{Q(s,a)}$	Q value function
$\boldsymbol{Q(s_t,a_t)}$	State-action pair and Q value at time t
$\boldsymbol{q}$	3D location of pixel p
$\boldsymbol{q_*(s)}$	Optimal action value function
$\boldsymbol{q_\pi(s,a)}$	Action-value function for policy π
$\boldsymbol{R_{s,s'}^a}$	Reward function for action a given s for all possible next states s'
$\boldsymbol{R_g}$	Reward function for grasping
$\boldsymbol{R_p}$	Reward function for pushing
$\boldsymbol{S}$	All possible states
$\boldsymbol{s}$	State in reinforcement learning formulation
$\boldsymbol{s'}$	All possible transition states
$\boldsymbol{t}$	Time step
$\boldsymbol{v_*(s)}$	Optimal state value function
$\boldsymbol{v_\pi(s)}$	Value function for policy π
$\boldsymbol{x}$	Input of supervised learning process
$\boldsymbol{x_x}$	x location in xy-plane for motion primitive action
$\boldsymbol{x_y}$	Y location in xy-plane for motion primitive action
$\boldsymbol{y}$	Output of supervised learning process
$\boldsymbol{y_i}$	Ground truth labels for Huber Loss Function

1 Introduction

Human handling of agro-food products is required in many stages of modern food processing. This interaction with agro-food products places various challenges on the food supply chain. One of these challenges is the physical demand placed on workers performing highly repetitive tasks. Poultry processing systems in particular have a history of overprocessing which has resulted in workers developing repetitive motion disorders (Striffler, 2007). An additional challenge comes from surfaces, in particular workers' hands, that come in contact with whole or cut food products creating points of contamination throughout the entire food system (Beuchat & Ryu, 1997). Recently, the COVID-19 pandemic has revealed additional vulnerabilities by blocking seasonal labourers from food processing sites resulting in disruptions of the food supply chain (NFU, 2020). These challenges are exacerbated by growing populations. In the past, food processing systems have overprocessed by increasing the number of workers to satisfy rising demands, but this approach increases the dependence on manual labour while increasing the negative effects previously described (Striffler, 2007). Now overprocessing can be handled and transferred to machines, but fully replacing manual tasks remains a challenge. An effective way to circumvent these challenges is through the application of automated manipulation tasks in food processing. Automating manipulation tasks may also result in lower production costs, higher productivity, and lower costs of agro-food products.

1.1 Current Situation

Robotics is just one method of automation used for primary and secondary processing of food variants (Granta, 2017). Primary processing is the way raw food is processed through cleaning, sorting, blending, and transport, while robotic automation levels can currently handle butchery, sorting, and grading fruits and vegetables. After primary processing, secondary processing is how the primarily processed food is converted into a new food product by cooking, baking, freezing, etc. Current levels of secondary food processing by robotic automation include sorting, defect removal, and mixing operations (Sain et al., 2020).

Raw meat is a primary reason for viral contamination, which spreads mostly by direct contact (Purnell et al., 2013). Therefore, current industries are shifting their business models toward the automated model replacing manual labourers with robots to improve production rates and avoid serious injuries. Unpleasant conditions in the meat and fruit production industries are not fit for human labourers to work comfortably and consistently because of smell, microbial infection, and heat (Ranger et al., 2004). By using automated robots, these issues can be circumvented with more reliable and consistent results. Automation used for food processing industries is done by using robots that can cut, load, material handle, etc. (Prasad, 2017). Some examples of current robot use include deboning and smart cutting of chickens (Anon, 2000). For high-quality cutting and maximizing chicken breast meat yield, robots can cut close to the bone ensuring food safety by preventing bone chips or contamination (Calderone, 2013).

Compared to the cutting of raw meats, the packaging of processed food as performed by various automated robots is lagging (Buckenhüskes & Oppenhäuser, 2014). The basic functions here focus on the picking up or grasping of products from one place and relocating them to another place. Picking up objects for processing is a significant challenge for automated manipulation in food processing and packaging.

Robotic bin picking is a challenging yet important technique that could potentially automate many tasks in industrial processing including the previously mentioned food processing tasks. Current pick and place are so difficult because of the requirement of many sub-tasks to work robustly with one another (Joffe et al., 2019). Recently, deep learning methods have become more popular because of increases in the computational capabilities of devices like GPUs. These methods have shown state-of-the-art performance in a wide range of computer vision, audio, and language processing scenarios. They have also been successfully applied to detect grasping points in more complex learning environments (Najafabadi et al., 2015). Current deep learning approaches do not rely on object identity but can use deep learning to predict object affordances. Affordance-based grasping point detection has gained attention in its ability to learn from the shape, colour, and texture of the manipulation objects. This makes these algorithms capable of generalizing to never before seen objects, creating flexible grasping solutions (Zeng et al., 2018B).

1.2 Desired Situation

Inspired by the way humans effortlessly grasp objects, the food processing industry aims to develop methods for robotic grasping of deformable agro-food objects. To accomplish this, the methods must enable a robotic system to learn how to grasp agro-food objects with variations in size, shape, and colour within a cluttered setting. Through automation, object singulation of individual agro-food objects, such as chicken pieces, should be possible by detecting object orientation and position. The agro-food objects should then be autonomously picked up from the cluttered arrangement and placed on a conveyor for further processing. To be successful, the automation methods must be capable of performing these tasks without damaging the products and be able to learn how to deal with new object variations. Additionally, the robotic systems used must be capable of safely functioning around human workers to ensure maximum safety of employees. If the robots cannot be safely operated around workers, then the advantages of the robotic system in terms of reduction of work-induced injuries as stated in Section 1.1 becomes a moot point.

1.3 Problem Definition

This thesis aims to advance the development of efficient robotic manipulation through deep learning methods for use in food and agriculture processing scenarios. Current deep reinforcement learning methods for picking up objects in cluttered environments have focused almost entirely on rigid objects with well-defined geometries (Luijkx, 2020). These methods have rarely been extended to agro-food products in clutter, which is a frequent occurrence in food processing tasks. The method must therefore be able to handle non-rigid objects such as poultry or fruit objects that have product variations in shape, size, and colour.

1.4 Objective

The focus of the thesis is to experiment with a self-learning method to pick up agro-food products from a cluttered environment. The study will aim to demonstrate the potential of using deep reinforcement learning methods on pick and place operations of deformable objects in the food processing industry. The objective is to determine if the addition of a secondary motion primitive, such as pushing, can enable a robotic agent to perform pick and place operations of deformable agro-food products through self-learning. The addition of a secondary motion primitive to a grasping algorithm can allow for a robotic manipulator to partially mimic human skilled manipulation which is necessary for handling cluttered environments with product variations. To handle cluttered environments, skilled robotic manipulation that can partially mimic human behaviour is necessary (Luijkx, 2020). Skilled manipulation makes use of both non-prehensile (e.g., pushing) and prehensile (e.g., grasping) actions: pushing can rearrange clutter to create space for fingers while grasping can displace objects making pushing more precise (Lynch & Mason, 1999). Pushing and grasping planning in robotics have been primarily studied in isolation of one another. The limited work that has combined these two motion primitives has not been applied to specific use case scenarios like the pick and place of deformable agro-food objects in a cluttered scene.

In the following work, a robotic arm with top-down grasping and a two-finger gripper will be used to improve upon the Visual Pushing for Grasping, referred to throughout this report as VPG, method for a specific agro-food application (Zeng et al., 2018A). For reinforcement learning, the Q-learning algorithm will be made use of to learn grasping and pushing policies for workspace clearance of objects. To test performance, the base method of VPG without any changes to training objects or reward system was used as a baseline.

1.5 Research Questions

The main research question focuses on the specific improvements that can be found for better performance of VPG on complex, densely cluttered agro-food scenarios:

- What are effective strategies for improving VPG (visual pushing for grasping) performance when applying the method to a cluttered agro-food scenario?

The measures of performance will be discussed later in Section 3.13. The new method will take the same concepts as used in VPG but use a revised reward system and expand the set of training objects. To find effective strategies for improving the method, a series of experiments will be used to find what the optimal revised reward system is. New sets of training objects will be applied to the method as well. These experiments will focus on answering the following sub-questions which will determine the new strategies that could improve VPG and in turn provide the basis for answering the main research question:

- How does the reward system for pushing affect performance?
- How do the training objects affect performance?

1.6 Demarcation

The supervision and materials for this thesis project were provided by the Wageningen University & Research Farm Technology FlexCRAFT Group. As a group, the research and projects performed are meant to develop the building blocks for addressing the automatization of poultry processing (P6: Food-Processing Robotics, n.d.). The specific use case is meant to address the challenge of how to deal with variation in shape and size of objects, the deformability of objects, and variation in containers.

This thesis project focused specifically on providing a building block for robotic bin picking of agro-food objects through deep reinforcement learning. The eventual goal is to singulate chicken objects individually from a complex pile of cluttered objects. Since this project is an early attempt of the FlexCRAFT group to explore the use of deep reinforcement methods combined with multiple robotic actions (i.e., pushing and grasping) for bin-picking operations, the experiments and evaluation remained in simulation and were not performed on a robot in the real-world. This limitation required different manipulation objects to be used within the simulation as raw chicken objects have physical parameters which are difficult to model in a simulated environment. Instead of chicken objects, crop objects of bananas and carrots were used for evaluating if VPG is an acceptable candidate method for bin-picking operations in the poultry industry.

1.7 Outline

The structure of the thesis is organized as follows: Section 2 formally defines the State-of-the-Art robotic manipulation that provides a basis for how the robotic agent in this thesis will manipulate objects in the experiments. In Section 3, the formal description of the methods and materials used to create and carry out the experiments is provided. Section 4 presents the results of the experiments from revised reward for pushing, new sets of training objects, and a final comparison of the baseline VPG with an agro-food specific variant (AFV) of VPG. In Section 5, the results of this project will be compared with other methods for combining multiple motion primitives using deep reinforcement learning. Section 6 will then provide conclusions of the work and summaries of the different experiments that will answer the previously presented research questions. Finally, Section 7 will provide recommendations for future researchers in obtaining the poultry use case application desired.

2 Background

The work in this thesis lies at the intersection of reinforcement learning, computer vision, and robotic manipulation. In this section, the related works that apply the fundamentals of reinforcement learning and deep learning to robotic manipulation will be discussed further. The state-of-the-art of robotic grasping, pushing, and combined actions will be discussed to lay the foundation for the project and experiments carried out in this thesis.

2.1 Non-Prehensile Manipulation

Pushing is one of many non-prehensile manipulation modalities (Lynch & Mason, 1999). When objects are too large or too cluttered this motion primitive enables greater performance on such complex manipulation tasks. A majority of previous works on robotic pushing focused on generating accurate predictions for the push outcome. For many of these works, pure analytical methods were used that relied heavily on strong assumptions without considering uncertainty (Dogar & Srinivasa, 2012; Miyazawa et al., 2005). Other studies take a data-driven approach that uses observations for estimating the parameters of the analytical models (Walker & Salisbury, 2008; Bauza & Rodriguez, 2017). These data-driven approaches can generalize pushing dynamics, but they require explicit object modelling and are incredibly sensitive to parameter settings. These issues do not appear under approaches that learn from data like in deep reinforcement approaches.

For vision-based methods, a learning method was introduced that could push an unknown object with a single point of rotation for contact to an arbitrary goal location (Salganicoff et al., 1993). This approach was expanded upon to handle objects with more complex shapes (Lau et al., 2011). Apart from these methods, recent work has begun leveraging deep learning to model the pushing dynamics (Bryavan & Fox, 2017; Watters et al., 2017). More recently, deep reinforcement learning methods have been developed for modelling pushing dynamics (Ghadirzadeh et al., 2017; Finn & Levine, 2017). Despite these advances, these recent works have focused on accurate and stable pushing on isolated objects without considering additional types of manipulation. Additionally, they do not explore pushing actions and their effect on dense-cluttered workspaces. In contrast, the work of this thesis will combine both non-prehensile pushing with grasping to allow the system to learn clutter clearance autonomously.

2.2 Prehensile Manipulation

Grasping motion primitives have been widely explored, so this section will only discuss the branch of grasping works that lay the foundation of grasping for the context of this work.

Classical analytical approaches for grasping find stable force-closure for known objects using 3D models of the objects along with their physical properties (Bohg et al., 2013). Beyond these classic approaches, deep learning-based computer vision has improved considerably providing data-driven approaches for robotic grasping. These data-driven approaches can be divided into two groups: model-based or model-free approaches. For model-based methods, grasps are sampled using object detection combined with pose estimation and grasp estimation to pick or grasp objects (Li et al., 2018; Shi et al., 2019). These approaches all rely on leveraging object-specific knowledge such as shape or pose, which limits their ability to generalize to unknown object classes and novel shapes. Assuming knowledge of object features is rarely possible for novel objects in unstructured environments. There now exist real-world datasets with a large amount of labelled data, which have been developed to improve generalization abilities (Depierre et al., 2018; Fang et al., 2020). Even with these datasets the collection and labelling of real-world data is time-consuming and laborious.

Inversely, a few data-driven approaches explore model-agnostic grasping strategies that directly connect visual data to candidate grasps (Zeng, et al., 2018B; Kumra et al., 2020). These methods detect grasps by exploiting learned visual features and do not explicitly use object-specific knowledge. The approach explored in this thesis is based on model-free deep reinforcement learning where FCNs are used to efficiently model the policies with grasping affordances to improve run-times (Zeng et al., 2018B). Reinforcement learning allows for learning of more extensive grasp representation via exploitation and exploration in a self-supervised way.

2.3 Combined Pushing and Grasping

Combining both non-prehensile and prehensile manipulation policies has the potential to expand the environments and abilities of robotic manipulation, but as an area of research, it is still being explored and developed. A robust planning framework for pushing and grasping under uncertainty was first introduced by Dogar et al. (2012). This framework exploited pre-grasping actions for dense clutter manipulation, but the policies were largely hand-crafted. In contrast, the approach used in this thesis is data-driven and learned online via self-supervision.

Other methods explored the model-free planning of pushing objects to target locations that are more favourable for pre-designed grasping algorithms, which are handcrafted, fixed, and known in advance (Omrcen et al., 2009; Ignasi et al., 2017). This knowledge is required for defining concrete goals, such as the target locations, which aid in designing or training of pushing policies. Attempting to define similar goals for a data-driven model-agnostic grasping policy is much less clear because this type of policy is constantly learning and adapting behaviour over time with more data. The optimal behaviours emerge from experience for such data-driven model-agnostic policies.

The work of Boularias *et al*. explores reinforcement learning for training control policies to select amongst push and grasp proposals represented by hand-crafted features (Boularias et al., 2015). The pipeline of their framework first segments images into objects proposes pushing and grasping actions, extracts hand-tuned features for each action, then executes the action with the highest expected reward. The method is limited by reliance on model-based simulation and hand-tuning to mainly convex objects. However, their work did provide the inspiration for the Visual Pushing for Grasping (VPG) framework applied in this thesis. In the VPG framework, a Q-learning methodology was introduced to learn the complementary pushing and grasping strategies (Zeng et al., 2018A). With an FCN as a function approximator to estimate the Q function of the reinforcement learning framework. The method significantly reduced training data required, but only trained with a limited range of objects. Additionally, the direct reward of pushing primitives which encourages pushing may cause issues. Encouraging too much pushing may push objects out of the workspace or a plethora of unnecessary pushing actions when there are only a few objects scattered in the environment and potential successful grasps are available (Mataric, 1994). Despite the limited training settings, the generalization to novel objects is promising. In this thesis, the VPG methodology was expanded and experimented upon by the introduction of new training objects as well as revised reward systems for pushing.

3 Materials and Methods

Before starting a discussion on the usage of deep learning for cluttered environment clearance, this chapter provides background information on key concepts of Reinforcement Learning and Deep Fully Convolutional Neural Networks, or FCNs. In the first section, Neural Networks and their basic concepts are presented with a special focus set on Fully Convolutional Networks. The next section details the fundamentals of traditional Reinforcement Learning. The basic algorithm for Temporal Difference Methods is presented as this forms the basis of the learning process for the methodology used in this thesis. Finally, the knowledge of these two sections will be combined in the final section which details Deep Reinforcement Learning.

In this chapter, the experiments and methods that were used in the thesis will be discussed in detail. The simulated model of the robotic arm that was used for training, simulation platform and overall problem formulation of learning joint pushing and grasping policies will be described.

3.1 Neural Networks

Biological neurons that makeup animal nervous systems inspired the concept of artificial neurons. Artificial neurons model complex biological interactions in a simplified way. Every neuron has n input connections. These inputs are processed by the neuron by taking the weighted sum, adding a bias b and applying an activation function (Wang, 2003). A series of neurons and their connections make up a neural network, which is capable of approximating non-linear functions. The parameter set of θ contains all weights θ_l of all neurons needs to be adjusted so the network results in the best possible function approximation (Wang, 2003). Finding a good parameter set is the learning process. Feedforward neural networks organize the neurons in different layers. The layers of neurons are connected in a forwarding direction. No connections are fed back to previous neurons making it a one-way pipeline. Each network has a single input layer, that processes raw input data, and one output layer, that contains the approximated result. Between these layers, there can be one or more hidden layers where relevant computing is happening to achieve the final function approximation (Goodfellow et al., 2016).

3.1.1 Learning Process

The learning process of neural networks focuses on finding a parameter set θ that results in the best function approximation. The true output y of a certain input x is given and can be used to update the parameters θ through supervised learning. This is done through an iterative process consisting of a forward pass, loss computation, back-propagation, followed by an update (Wang, 2003).

1. **Forward Pass:** Here the input is forwarded through the network and results in the predicted output $y_{pred} = f(x, \theta)$.
2. **Loss Computation:** Next the predicted output y_{pred} is compared to the true output by computing the loss $L(\theta)$. The loss function choice depends on the learning task.
3. **Back-propagation:** Now the global gradient of loss $\nabla L(\theta)$ is computed and back-propagated through the network. The back-propagation algorithm, introduced by Rumelhart et al., (1986) provides a local gradient of loss to all hidden layers. The chain rule is the underlying concept of the algorithm (Goodfellow, et al. 2016).

The chain rule propagates the global gradient loss $\frac{\partial L(\theta)}{\partial \theta}$ back through the network (the opposite direction of the forward pass).

4. **Update:** When the backpropagation is completed all neuron weights are updated. Stochastic gradient descent (SGD) is a common optimizer. It combines batch learning with gradient descent. In gradient descent, the weights are changed in the negative direction of the gradient of loss so that the function approximation approaches closer to the minimum in each iteration. After several iterations, a local minimum is expected to be reached. Equation (1) shows the update rule of gradient descent. Alpha α is the learning rate that defines how quickly the minimum should be approached. Too high of a learning rate risks the minimum never being reached because of overshoot. Too low of a learning rate will result in inefficient training of the network (Karpathy, n.d.).

$$\theta_i \leftarrow \theta_i + \alpha \nabla_\theta L(\theta) \qquad [-] \qquad (1)$$

3.1.2 Activation Functions

There are three different commonly used activation functions but here the focus will be on Rectified Linear Unit, or ReLU, activation. The other two common activation functions (sigmoid and tanh) have a crucial disadvantage when it comes to saturation. If the neuron's output saturates at 0 or 1, the local gradient becomes almost zero. Then during backpropagation, the global gradient is multiplied by the local gradients resulting in a product of zero. Because of this the weights will no longer update and change, and the network will lose its capability to learn effectively. The ReLU function is the most popular and is presented below in Equation (2). It does not allow the outputted term to be below zero. ReLU activation has a non-saturating form and allows gradients to converge faster during training. An additional advantage is that it is a simple function with a small computational loss (Nair & Hinton, 2010).

$$ReLU(x) = \max(0, x) \qquad [-] \qquad (2)$$

3.1.3 Batch Learning and Normalization

Batch learning processes a set of m training samples (mini-batches) instead of just a single training iteration. A gradient is averaged over all m processed training examples. By doing this, a more accurate gradient can be found with less variance resulting in reduced training time (Joffe & Szegedy, 2015). When using a graphic processing unit (GPU), batch learning significantly speeds up the learning in training. The batch idea can process all samples independently or in parallel (Karpathy, n.d.).

Batch normalization normalizes the whole batch by zero-centring and rescaling the data. The mean of the normalized data is expected to be zero or close to it with a variance close to one. Applying batch normalization to input data as well as the outputs of any hidden layers leads to a regularized effect that can prevent overfitting. Doing so will also speed up training time. Batch normalization is only applicable if the exact positioning of features is not relevant. All that is necessary is that the feature exists in the input. This makes batch normalization handy for unstructured learning environments (Joffe & Szegedy, 2015).

3.1.4 Fully Convolutional Networks

Fully Convolutional Networks, or FCNs, are neural networks that use convolutional neural networks, or CNNs, to transform image pixels into pixel classes (Long et al., 2015). CNNs are typically used in the field of Computer Vision for object detection (Girshick, 2015; Redmon et al., 2016) or image segmentation (He et al., 2017). Unlike CNNs, FCNs transform the height and width of intermediate feature maps between network layers back to the original input image size. This is done through the transposed convolutional layer. The result is a classification output that has a one-to-one correspondence in pixel level with the input image. An example illustrating the input and output of an FCN can be found in Figure 1. With FCNs, both learning and inference can be performed whole-image-at-a-time by dense feedforward computation and backpropagation. This enables pixel-wise prediction and learning in the networks through upsampling of layers with subsampled pooling (Long et al., 2015).

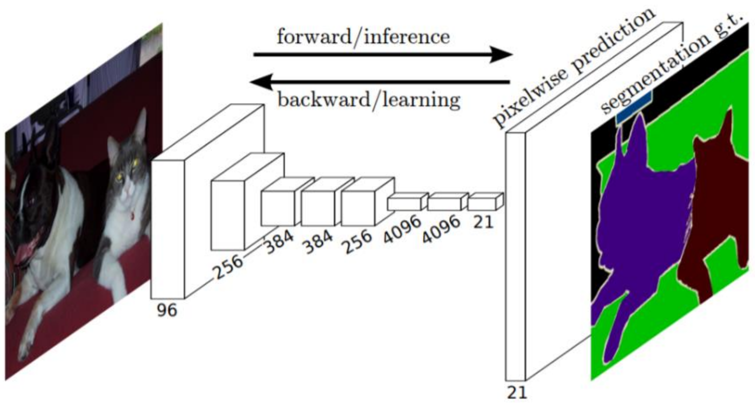


Figure 1: Fully convolutional networks can efficiently learn to make dense predictions for individual pixel based tasks like semantic segmentation (Long et al., 2015).

3.1.5 DenseNets

Recent work has ensured maximum information flow between layers in a deep network by connecting all layers directly with each other. The feed-forward action of neural networks was preserved by each layer receiving additional inputs from all previous layers while passing on its feature maps to all subsequent layers. Figure 2 illustrates how this layout works schematically. In this framework, features are never combined via summation before being passed to a layer. Features are instead combined by concatenating them. Therefore, the l^{th} layer has l inputs, consisting of the feature maps of all preceding convolutional blocks. An individual layer's feature maps are passed on to all $L-l$ subsequent layers. This creates $\frac{L(L+1)}{2}$ connections in an L-layer network, as opposed to traditional architectures with only L number of connections. This dense connectivity is the reasoning behind the name Dense Convolutional Network or DenseNet. DenseNets have several compelling advantages as they alleviate vanishing gradients between layers, strengthen feature map propagation, encourage feature reuse, and reduce the number of parameters required (Huang et al., 2017).

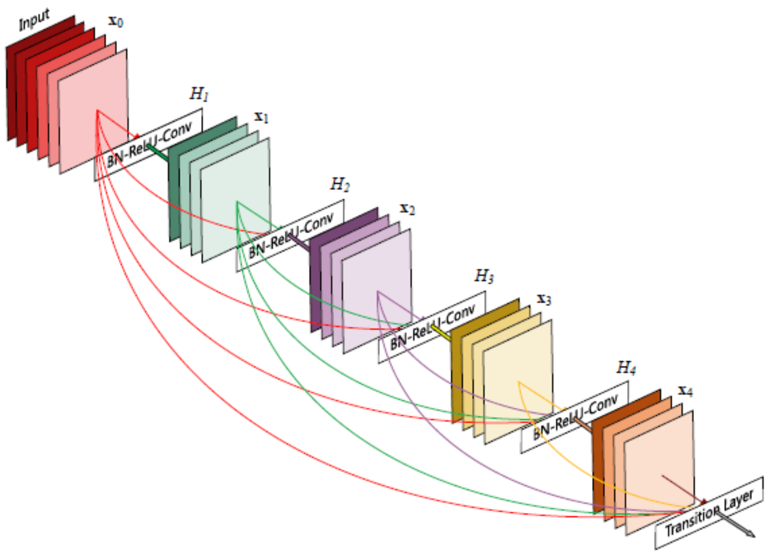


Figure 2: A 5-layer dense block. Each layer takes all preceding feature maps as input (Huang et al., 2017).

3.2 Reinforcement Learning

Within reinforcement learning, an agent is meant to learn a specified behaviour via trial and error. As the agent interacts with its environment, it collects experiences. Figure 3 illustrates the basic idea that is reinforcement learning. For each time step t, the agent is in a certain state $s_t \in S$ and takes one of the possible available actions $a_t \in A(s)$. Subsequently the environment will change due to the action taken and the agent will begin in a new state s_{t+1}. The action also produces a reward R_{t+1} from the environment which gives feedback to the how well the action a_t performed in state s_t (Sutton & Barto, 2014).

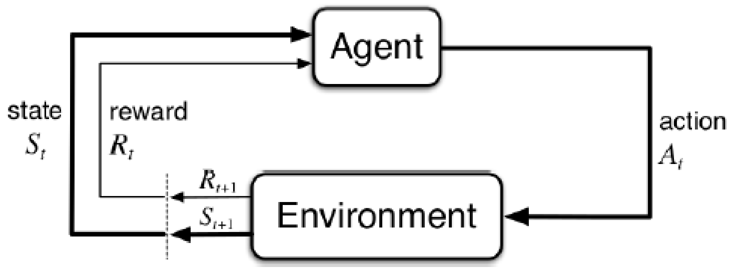


Figure 3: General formulation of reinforcement learning: an agent interacts with the environment to learn from experience. At each time step t, the agent is in a given state s_t (Sutton & Barto, 2014).

In the following subsections, an introduction to the basics of classical Reinforcement Learning will be discussed. It will provide a foundation for the more advanced Deep Reinforcement Learning approaches that will be discussed afterwards. All information in the following sections, specifically equations and derivations, are provided and accessible in Sutton and Barto (2014).

3.2.1 Markov Decision Process

The Markov Assumption assumes independence between past and future states, which means that the state and the behaviour of the environment at time t are not influenced by the past agent-environment interactions. When a Reinforcement Learning task satisfies this assumption, the task can be formulated as a five-tuple Markov Decision Process (MDP). These five components are as follows:

- Set of states: S
- Set of actions: A where $A(s)$ is the set of all available actions in state s.
- Transition probabilities $P^{a}_{s,s'}: (S \times A \times S) \rightarrow [0,1]$ which are the probabilities of the transition of any state s to the next state s' when taking action a at time step t.
- Reward functions $R^{a}_{s,s'}: (S \times A \times S) \rightarrow [0,1]$ which defines the immediate rewards the agent receives after transitioning from one state s to the next s'.
- Discount factor $\gamma \in [0,1]$ for computing the discounted expected return.

An MDP is finite if the set of states S and actions A is finite.

3.2.2 Discounted Expected Reward

To effectively train an agent, its goal should be to maximize the reward in the long term instead of prioritizing immediate returns. The discounted expected return is the cumulative sum of possible future rewards and can be seen below in Equation (3). The discount factor γ as previously mentioned is a value between 0 and 1 that rates the future rewards and defines how far in the future rewards are considered. For a discount factor of 0, only the immediate reward is taken into account. This can be considered a myopic or shortsighted reward structure. A discount factor of 1, considers all rewards of the future to have the same value making it a farsighted reward structure.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad [-] \qquad (3)$$

3.2.3 Policy and Value Functions

The policy π directs agent behaviour. The policy models a probability distribution $\pi(a|s)$ over the number of available actions $a \in A(s)$ for all states s. The agent's next action is sampled from this probability distribution $\pi(a|s)$. A simpler definition can represent the policy as the strategy the agent uses to pursue an objective. If a policy is greedy, it will look to exploit previously learned information. However, a greedy policy risks getting stuck in a local minimum. An alternative exists, called ϵ-greedy where $\epsilon \in [0,1]$, creates a policy that has ϵ probability of exploring, and a (1-ϵ) chance of following the optimal path.

$$\pi(a|s) \leftarrow \begin{cases} 1 - \epsilon + \dfrac{\epsilon}{|A(s)|} & \text{if } a = arg \max_{a \in A(s)} Q(s,a) \\ \dfrac{\epsilon}{|A(s)|} & \text{otherwise} \end{cases} \qquad [-] \qquad (4)$$

A value function $v_\pi(s)$ is an estimation of how the agent's current state s will affect future performance and reward. If the agent acts according to the policy π, then the value function $v_\pi(s)$ is the expected discounted return for that state s. The mathematical interpretation of this is presented below:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[\textstyle\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s], for\ all\ s \in S \qquad [-] \qquad (5)$$

If Equation (5) is formulated recursively, then it is called the Bellman Equation for v_π. The recurvsive form is presented below in Equation (6). For the Bellman Equation, the value of state s is dependent only on the next possible states s' where each is weighted by the transition probability $P^{a}_{s,s'}$. A majority of Reinforcement Learning applications approximate the optimal Bellman Equation via approximating the next states' values.

$$v_{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P^{a}_{s,s'}[R^{a}_{s,s'} + \gamma v_{\pi}(s')] \qquad [-] \qquad (6)$$

The action-value function $q_n(s,a)$ is the estimation of performance for an action a in state s. This function can be described as the expected return of taking action a in state s and then directing its behaviour according to the policy π.

$$q_{\pi}(s) = \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a] = \mathbb{E}_{\pi}[\textstyle\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s, A_t = a] \qquad [-] \qquad (7)$$
$$\text{for all } s \in S \text{ and } a \in A$$

The objective of Reinforcement Learning is to find an optimal policy π. If the value function of a new policy is better $v_{\pi}(s) \geq v_{\pi'}(s)$ for all $s \in S$, then the new policy is also better $\pi \geq \pi'$. For an optimal state-value function, an optimal policy will be used by the agent. There is always a possibility there exists more than a single optimal policy that leads to the same optimal state-value function. An optimal state-value function can be seen below:

$$v_{*}(s) = \max_{\pi} v_{\pi}(s) \text{ for all } s \in S \qquad [-] \qquad (8)$$

Subsequently, optimal policies also result in an optimal action-value function q_*

$$q_{*}(s) = \max_{\pi} q_{\pi}(s,a) \text{ for all } s \in S \text{ and } a \in A(s) \qquad [-] \qquad (9)$$
$$= \mathbb{E}[R_{t+1} + \gamma v_{*}(s')|S_t = s, A_t = a]$$

Lastly, the Bellman Optimality Equation can be derived from the previously introduced equations and is shown below:

$$v_{*}(s) = \max_{a} \sum_{s'} P^{a}_{s,s'}\,[R^{a}_{s,s'} + \gamma \max_{a'} q_{\pi_*}(s',a')] \qquad [-] \qquad (10)$$

3.2.4 Temporal Difference Methods

The advantage of Temporal Difference (TD) methods is that they apply to online learning and continuous tasks. In each time step t, the policy is updated so there is no need for completed episodes. Additionally, no model of the environment is required allowing for model-free approaches.

When using TD-methods, the true expected return G_t is not available so it is instead approximated by the TD-target. This target can be seen as an approximation of the Bellman Equation (10). The TD-target is found by taking the sum of the immediate return and the discounted expected value of the next state: $R_{t+1} + \gamma V(S_{t+1})$. Pseudo-code of this general concept can be found in algorithm 1. In each iteration of each episode, an action A_t is retrieved from policy π with R_{t+1} and transitioned to the next state S_{t+1}. Finally, the value-table V is updated with the TD-error, which is the difference between the TD-target and $V(S_t)$.

```
Data: policy $\pi, a \in [0,1]$
```

```
Initialize $V(s)$ for all $s \in S$ arbitrarily, except $V(terminal) = 0$;
for each episode do:
    Initialize $S_t$ do:
        $A_t$ ←action given by $\pi$ for $S_t$;
        Take action $A_t$, observe $R_{t+1}$ and $S_{t+1}$;
        $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+!} + \gamma V(S_{t+1}) - V(S_t)]$; where $R_{t+!} + \gamma V(S_{t+1})$ is TD-target
                                                   and $R_{t+!} + \gamma V(S_{t+1}) - V(S_t)$ is TD-error
        $S_t \leftarrow S_{t+1}$
    while $S$ is not terminal;
end
```

Algorithm 1: General format of TD method algorithms: where expected return G_t is approximated with the sum of immediate return and the discounted expected value of the next state (TD-target) (Sutton & Barto, 2014).

3.2.4.1 Q-Learning

Q-learning is an off-policy TD control method. This means that the policy π is not used for updating the Q-table. Instead of using the action-value of the next state-action pair $Q(S_{t+1}, A_{t+1})$, only the maximum Q-value of the next state is taken. Below the action-value update rule of Q-learning can be seen.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \qquad [-] \qquad (11)$$

Here $\alpha \in [0,1]$ is the learning rate. This parameter shows how much the agent must adjust the policy estimates based on the TD-error. As previously stated in Section 3.1.1 large learning rates adjust aggressively and may create high fluctuating training results that do not converge. Smaller learning rates lead to slow adjustments that take more time to converge.

3.3 Deep Reinforcement Learning

In classical reinforcement learning, all approaches make use of a tabular setting which can lead to essential disadvantages. The usage of a table limits the classical approaches to tasks with a low number of states and actions. In the real world, the state space of tasks can quickly get very large. This makes it not feasible to visit all possible states to retrieve their respective values for all action-state pairs. Also, the size of the table is limited due to hardware memory constraints. Inversely, the tabular setting does not share knowledge about similar states which could lead to better representation and lower training times (Sutton & Barto, 2014).

To overcome such restrictions, value tables can be replaced with a Deep Neural Network as a function approximator. Neural networks and their ability to approximate non-linear functions to extract relevant features such as affordances from raw inputs make it possible to generalize over unseen states (Minh et al., 2015).

3.3.1 Value-Based Methods

Value-Based methods extend upon Temporal Difference Methods from classical Reinforcement Learning. The goal is to replace the value table one-to-one and to approximate it with a Deep Neural Network. The output from the neural network provides probabilities for each possible action. On top of the network, a traditional policy supervises in a sense to choose the final action, e.g, ϵ-greedy policy (4).

3.3.1.1 Deep Q-Network

Combining Q-learning with non-linear function approximation has been investigated in the past, but did not see much success because of unstable learning. The DeepMind group presented an approach called deep Q-network (DQN) that had great success. Here the researchers combined the model-free, off-policy Q-Learning with Deep Neural Networks. For input data, high-dimensional raw sensory input with no hand-crafted features are used. This end-to-end architecture allows the network to extract relevant features by itself. The Q-network can then output a probability distribution over all possible discrete actions. This allows an agent to determine the best action for a given state with a single forward pass. The issue of unstable learning experienced in early attempts of DQNs was resolved with two mechanisms, called experience replay and frozen target network (Minh et al., 2015).

Experience replay allows the agent to store its experiences in a buffer (Shaul et al., 2016). At each training step, a batch of these stored experiences is uniformly sampled from the buffer and fed to the network. Experience replay removes the correlations in data sequences and feeds independent data to the network while ensuring old experiences are repeated occasionally. Overall, the experience replay has a smoothing effect over changes in data distribution.

For DQN, the network is updated according to the loss function from Equation (12). This loss function is found by taking the squared TD-error.

$$L_i(\theta_i) = \dot{\mathbb{E}}_t\,[((R_{t+1} + \gamma \max_a Q(S_{t+1}, a, \theta_i^-)) - Q(S_t\,, A_t, \theta_t)\,)^2] \qquad [\text{-}] \qquad (12)$$

In the above equation, the second mechanism acts as the frozen target network. Here two networks with the same structure, but different weights are used: θ for the Q-network and θ^- for the target network. Regular updates are made to the Q-network according to the loss function as seen in Equation (12), while the target network is updated by copying the Q-network parameters to the target network $\theta^- = \theta$ after a set number of time steps. This means that the target network θ^- weights are frozen and held constant for those time steps. This mechanism smooths oscillating policies and leads to more stabilized learning (Minh et al., 2015).

3.3.1.2 Prioritized Experience Replay

The goal of prioritized experience replay is to prioritize experiences that contain more important information than other past experiences. Each experience is stored with a priority value so that higher priority past experiences have a higher sampling probability. This allows these prioritized experiences a chance to remain in the replay buffer longer than others. The TD-error can be used as the measure for determining the priority value. It can be assumed that if the TD-error is high, the agent can learn more from the linked experience. A high TD-error means the agent behaved much better or worse than expected, so the information should have value on how to direct further actions and experiences. By applying prioritized experienced replay, the learning process can be sped up by a factor of two (Shaul, et al. 2016).

3.4 Reinforcement Learning Problem Formulation

The task of using pushing and grasping to handle cluttered environments was first formulated as an MDP. MDP framework is discussed previously in Section 3.2.1. The MDP for pushing-for-grasping can be viewed as a 5-tuple: at any given state s_t at time t, the agent chooses and executes an action a_t according to a policy $\pi(s_t)$, then transitions to a new state s_{t+1} and receives an immediate corresponding reward $R_{a_t}(s_t, s_{t+1})$. The goal is to find an optimal policy that maximizes the expected sum of future rewards. Formally, the method makes use of deep Q-learning with FCNs as the function approximators.

The actions consist of the motion primitive chosen, which is executed at the 3D location q projected from a pixel p of the heightmap image representation of the current state:

$$a = (\psi, q) \mid \psi \in \{push, grasp\}, q \rightarrow p \in s_t \qquad [-] \qquad (13)$$

The default reward scheme is a piecewise function for successfully executed motion primitives. A successfully executed grasp is determined by checking the antipodal distance between the parallel-jaw gripper fingers after a grasp attempt is lower than a pre-defined threshold determined by the antipodal distance at the beginning of the grasp. In other words, if the distance between gripper fingers at the end of the grasp has not changed from when the object was first grasped, then the grasp was successful or assumed to be.

$$R_g(s_t, s_{t+1}) = \begin{cases} 1, successful\ grasp, \\ 0, otherwise. \end{cases} \qquad [-] \qquad (14)$$

A push is determined to be successful if a detectable change to the environment is made. This detectable change is found if the sum of differences between heightmaps before and after the pushing action exceed a certain manually set threshold, τ:

$$R_p(s_t, s_{t+1}) = \begin{cases} 0.5, if \sum (s_{t+1} - s_t) > \tau\,, \\ 0, otherwise. \end{cases} \qquad [-] \qquad (15)$$

The intrinsic reward for pushing does not explicitly consider whether a pushing action enables future grasps. The direct reward for pushing is meant to encourage the agent to make pushes that cause change and is set to half the reward of the grasping network to limit agent overfitting to the secondary motion primitive. The way this reward function is formulated and how it will impact the agent's performance is of specific interest for this project and will be discussed later in this section.

3.5 Learning Process

The reinforcement learning of the method is done through an off-policy Q learning algorithm. The goal of the algorithm is to enable an agent to learn a deterministic policy for collaborative pushing and grasping. For any state s_t at time t, the agent takes an action a_t based on a policy $\pi(s_t)$ and transitions to a new state s_{t+1} with a reward $R_{a_t}(s_t, s_{t+1})$. At each time step t, the agent selects an action that has the highest Q-value for a given heightmap state. The policy picks the action by maximizing the Q-function which estimates the expected reward for choosing action a_t at state s_t. Temporal difference (TD) learning, as discussed in Section 3.2.4, was used to minimize the TD-error of the current Q-value function $Q_\pi(s_t, a_t)$ to the label y_t where a' is the set of all possible actions.

$$y_t = R_{a_t}(s_t, s_{t+1}) + \gamma Q(s_{t+1}, \arg\max_{a'}(Q(s_{t+1}, a')) \qquad [-] \qquad (16)$$

3.6 Loss Function

Using the Huber loss function for the loss computation as described in Section 3.1.1, the Q-learning FCNs for pushing and grasping skills were trained. The loss is only computed for the selected pixel, p that corresponds to the 3D location q where the action will take place. All other pixels backpropagate with a loss of zero. For pushing, y_t is generated by calculating the scene image difference after the push. If it is higher than threshold τ, the push was successful as previously mentioned. The manual selection of the threshold makes the pushing method sensitive, but the use of Huber loss minimizes the sensitivity to inaccurate labels (Girshick, 2015). This is not the case for the grasping method as ground truth labels for y_i are generated via a feedback signal from the gripper.

$$\delta_t^{a_t} = \left| Q^{\theta_i} - y_i^{\theta_i^-} \right| \qquad [-] \qquad (17)$$

$$L_i = \begin{cases} \frac{1}{2}\left(\delta_t^{a_t}\right)^2, \left|\delta_t^{a_t}\right| < 1, \\ \left|\delta_t^{a_t}\right| - \frac{1}{2}, otherwise. \end{cases} \qquad [-] \qquad (18)$$

$\delta_t^{a_t}$ denotes the temporal differences for the motion primitives. θ_i are the parameters of the neural network at iteration i, and the target network parameters θ_i^- remain fixed between individual updates.

3.7 Perception Module

The RGB-D images are captured by a fixed-mount camera. The data is then projected onto a 3D point cloud. Finally, the data is orthographically back-projected upwards to create a heightmap image representation with colour (RGB) and depth (D) channels. The workspace for the system covers an area of 0.448^{2}m. With a pixel resolution of 224×224 for the heightmaps, the heightmap pixels correspond spatially to a 2^{2}mm vertical column of 3D space in the workspace. The method is formulated as a pixel-wise labelling problem, which means that each image pixel corresponds to a specific robot motion primitive that will be executed on the 3D location of that pixel. This pixel-wise labelling is possible through the usage of FCNs as described in Section 3.1.4.

Once the perception module has generated the feature representations and point clouds, they are fed into the pushing and grasping modules. The FCNs of the perception module are trained to infer affordances for both motion primitive modules across a dense pixel-wise sampling of end-effector orientations and locations (Zeng et al., 2018B). The approach relies on the assumption that graspable regions can be deduced from local geometry and visual appearance. To do so, the RGB-D heightmaps are used as input for a Q-function which is modelled as two-feed forward fully convolution networks (FCNs): one for each motion primitive. Each FCN takes the heightmap image of the state s_t and creates the pixel-wise map of Q values with the same image size and resolution. Each Q value prediction at a pixel p represents the future expected reward of executing a motion primitive at that 3D location q.

The FCNs share the same architecture: two parallel 121-layer DenseNet channels pre-trained on the ImageNet image database, followed by channel-wise concatenation and 2 additional 1×1 convolutional layers interleaved with nonlinear activation functions (ReLU) and spatial batch normalization, then bilinearly upsampled (Deng et al., 2009). One of the parallel 121-layer DenseNet towers takes the colour channels as input. The other tower takes the depth channel as input. By rotating the heightmap of the scene with 16 different orientations before feeding it as an input for the FCNs, 16 different gripper orientations around the vertical axis for pushing and grasping are accounted for. This means affordances for top-down parallel-jaw grasping actions at each pixel p are determined with 16 forward passes of the FCN to generate 16 output affordance maps.

Likewise, pushing actions at each location p with 16 forward passes through the pushing FCN generates another 16 output maps. In total, 32 Q maps are made of the heightmap image. A visual summary of the perception module is provided in Figure 4.

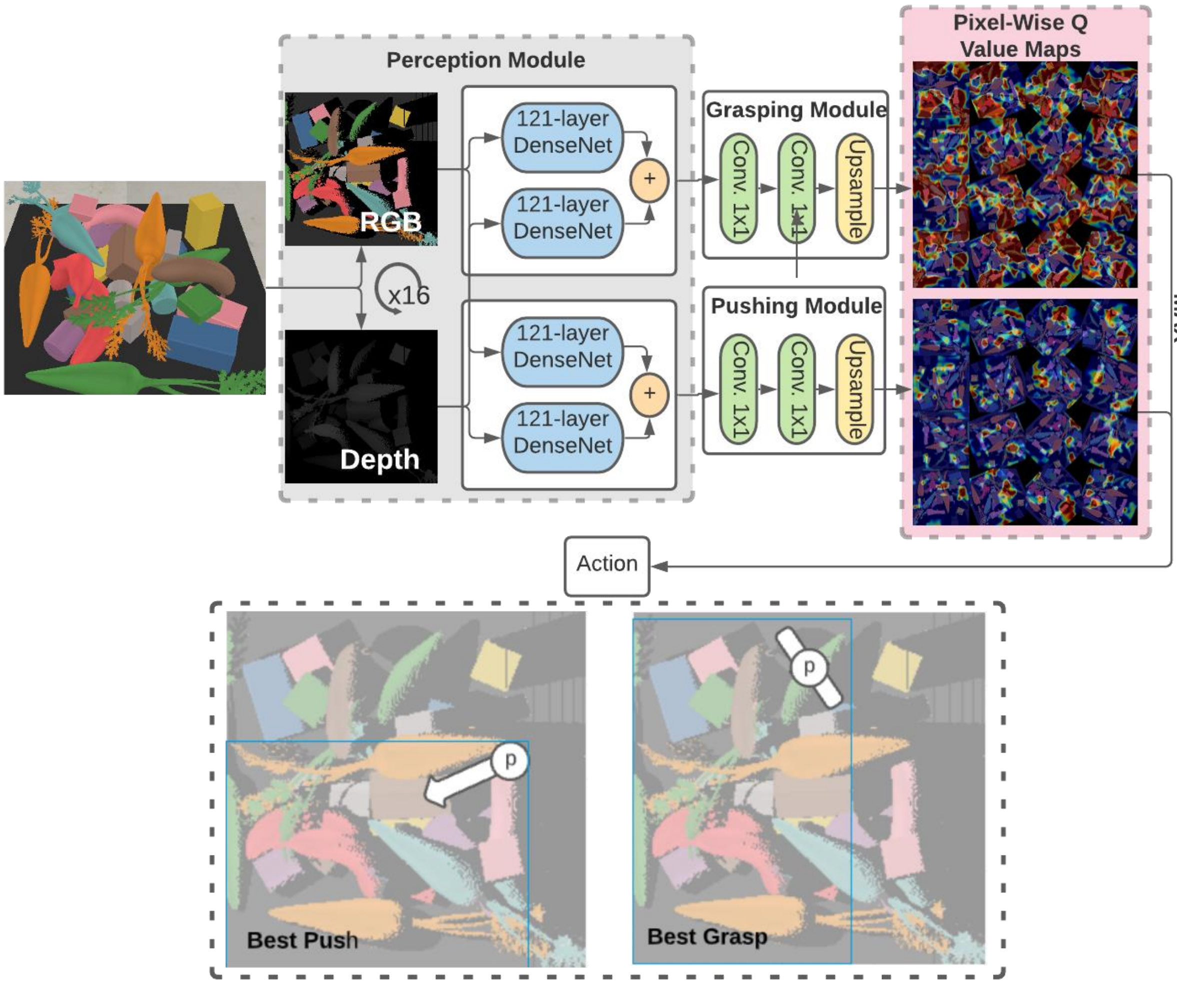


Figure 4: Overview of the system as it takes in RGB-D visual observations of the scene and predicts the next action primitive, pushing or grasping. The learning objective is to find a policy that always chooses the action that maximizes the expected success of current/future grasps for workspace clearance.

3.8 Pushing Module

A module for pushing is meant to predict the most beneficial pushing action for future grasps among a pixel-wise sampling of end-effector locations and orientations given s_t. The pushing action is a planar push parameterized as (x, β) where x denotes the location (x_x, x_y). With all pushes being planar, β is the rotated angle of the end-effector in the xy-plane. The push prediction network structure is made of a feed-forward FCN that maps the visual observation of the robot scene at time t to a dense pixel-wise Q-value map of the workspace. When a push is executed by the agent, the gripper is closed and moved to the selected location and pushed for 10cm in the selected direction. An example of an executed push is illustrated in Figure 5. The gripper height is held constant throughout the duration of the push to keep the action planar and stable. The length of the push is fixed to explore the synergy between the actions, but the direction is not.

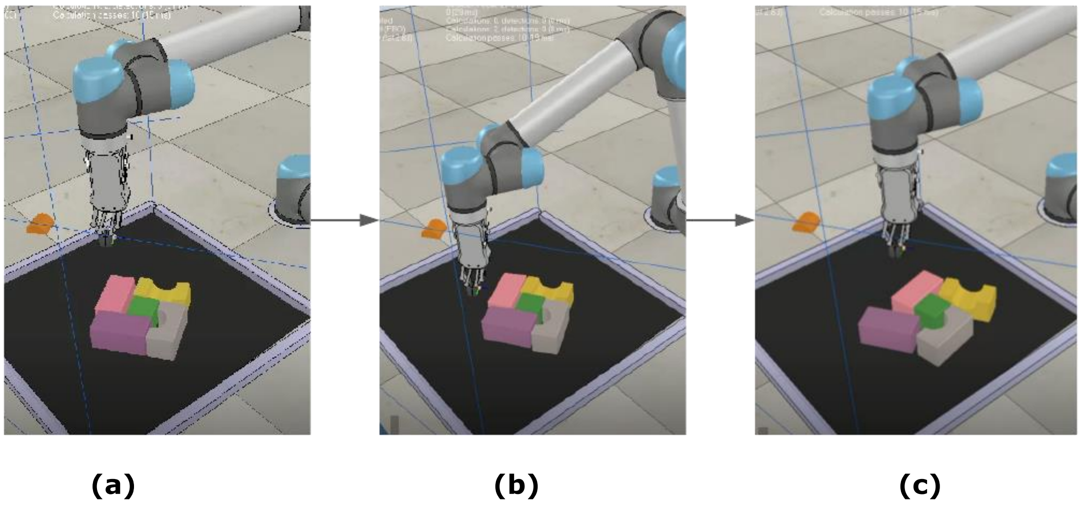


Figure 5: Snapshot images of pushing module in action. (a) The agent closes the girpper fingers, (b) goes to the 3D pushing pixel location, q, with fingertips of gripper 3mm below location, (c) executes planar push 10cm to the right with respect to the heightmap image orientation used to generate the specific pixel-wise Q prediction and subsequent pushing action.

3.9 Grasping Module

A module for grasping is meant to predict the most likely affordances for grasping actions among the pixel-wise sampling of end-effector locations and orientations given s_t. The grasping action is a top-down parallel-jaw grasp with the two-finger gripper parameterized as (x, β) where x denotes the location (x_x, x_y). Top-down grasping does well with picking up objects with smaller, irregular surfaces (e.g. deformable objects). With all grasps being top-down, β is the rotated angle of the end-effector in the xy-plane. The grasp prediction network structure is made of a feed-forward FCN that maps the visual observation from the perception module at time t to a dense pixel-wise Q-value map of the workspace. When a grasp is executed by the agent, the gripper is opened and both fingers attempt to move 3cm below q, the 3D location of the pixel, before closing the fingers. Once the fingers are closed, the robot lifts the object. An example of the UR5 arm performing this top-down grasping action is shown in Figure 6. If the antipodal distance of the fingers remains constant throughout the duration of the grasp, it is assumed that the grasp was successful.

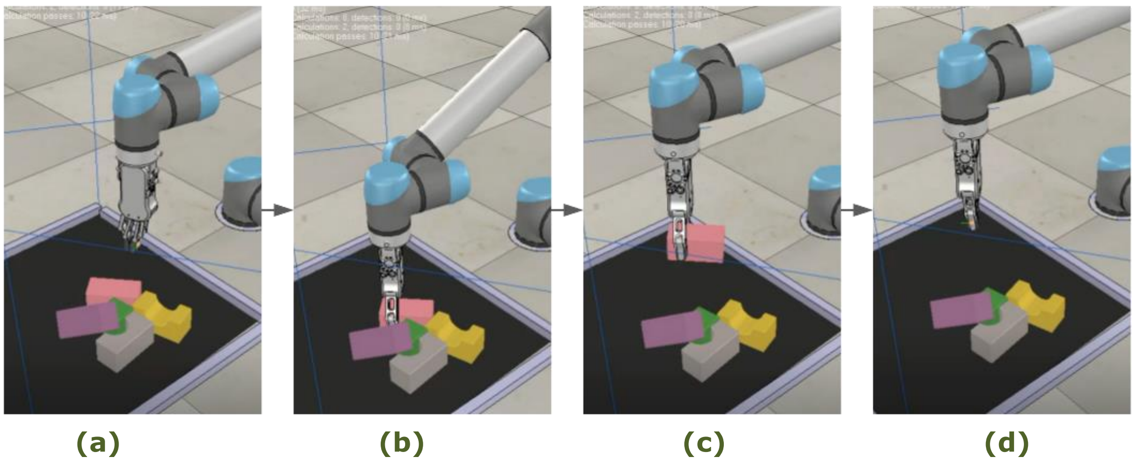


Figure 6: grasping module in action. (a) From starting location, (b) the gripper is opened and a top-down grasp begins centred of the specific pixel location, q, where the fingertips of the gripper stop 3mm below the location, q. (c) Constant force is applied to the object as it is lifted against the gravity direction. (d) Once the agent returns to its original height location, the object is recognized as successfully grasped and is removed from the workplace scene.

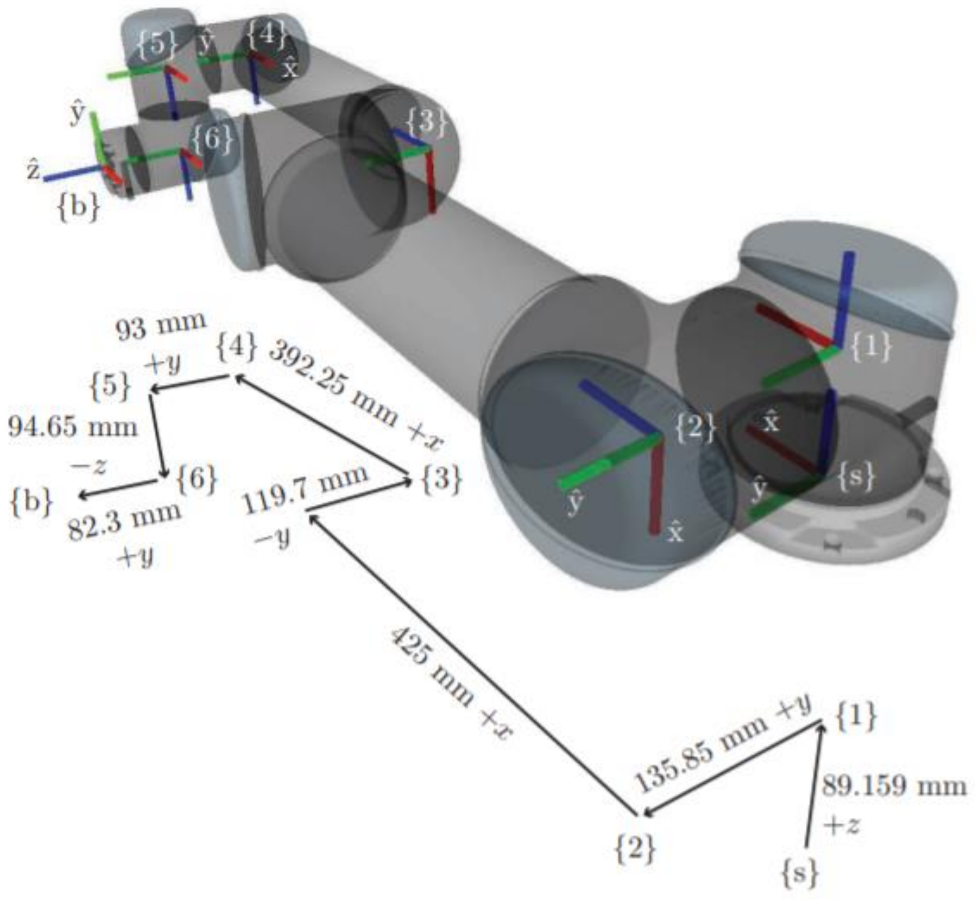
3.10 UR5 Robotic Arm

In this thesis, a UR5 robotic arm from Universal Robots was used (Universal Robots). A detailed list of the UR5 arms properties is provided in Table 1. The specific dimensions and parameters of the UR5 arm can be found in Figure 7. The collaborative robots from Universal Robots have various advantages that motivated the choice of the UR5 manipulator over an industrial robotic manipulator. To begin with collaborative robots—or cobots—including the UR5 arm are meant to work alongside human employees, while industrial robots are meant to work in place of those employees. A UR5 cobot can assist employees with work that may be too demanding or tedious for them to complete on their own without fully eliminating factory jobs. In opposition, industrial robots are used to automate the manufacturing process almost exclusively without human assistance on the processing lines. Cobots like the UR5 arm are more easily programmable than other manipulators and do not always need an engineer or programmer to write new code for any changes to be implemented. Factory workers can re-program a UR5 arm simply by moving the arm along the desired track. After this, the arm can store the movement action and "remember" it for later repeating the motion on its own. Finally, the sizes of UR5 arms mean that they are not designed for heavy manufacturing, which is the main area of application of industrial robots which can handle heavier and larger materials like in automobile manufacturing. The UR5 and other cobots, however, are safe enough to function around people and can easily manage the sizes and shapes of various agro-food processing settings. Given the desired situation described in Section 1.2 cobots are the best choice to maximize worker safety. These advantages motivated the choice of the UR5 cobot for this project.

Table 1: Specifications of the UR5 provided by Universal Robots.

Weight	18.4 kg
Payload	5 kg
Reach	850 mm
Joint Ranges	+/- 360° on all joints
Speed	Joint: Max 180°/sec. Tool: Approx. 1 m/sec.
Repeatability	+/- 0.1 mm
Footprint	149mm
Degrees of Freedom	6 rotating joints
Control box size (WxHxD)	475 mm × 423 mm × 268 mm
I/0 ports	10 digital in, 10 digital out, 4 analogue in, 2 analogue out
I/0 power supply	24 V 1200 mA in control box and 12 V/24 V 600 mA in tool
Communication	TCP/IP 100 Mbit: IEEE 802.3u, 100BASE-TX Ethernet socket & Modbus TCP
Programming	Polyscope graphical user interface on 12 inch touchscreen with mounting
Noise	Comparatively noiseless
IP classification	IP54
Power consumption	Approximately 200 watts using a typical program
Materials	Aluminium, ABS plastic
Temperature	Functional in 0-50°C

Power supply	100-240 VAC, 50-60 Hz
Calculated operating life	35,000 hours



Joint	Torsion angle (rad)	Rod length (mm)	Bias length (mm)	Joint angle (rad)
1	0	0	89.159	п/2
2	0	-425	0	0
3	0	-392.25	0	0
4	0	0	109.15	п/2
5	0	0	94.65	-п/2
6	0	0	82.3	0

Figure 7: Orientations of the frames {s} or the base, {b} or end effector, and joints {!} through {6} are illustrated on the translucent UR5 arm. {s} and {1} are aligned with each other, whereas {2} and {3} are aligned with each other, and {4} through {6} are aligned with each other. This explains why only the axes of {s}, {2}, {4}, and {b} are labeled. Below the image, the offsets of each link are indicated by the skeleton outline including distances directions. The specific Denavit-Hartenberg parameters of {1} through {6} are listed below the figure. Figure retrieved from Lynch & Park, 2017. Denavit-Hartenberg parameters taken from Universal Robots.

3.11 RG2 Gripper

The UR5 robot is combined with an RG2 gripper. The RG2 gripper is specially designed for robots from Universal Robots. The long-stroke capable of the gripper allows it to handle a variety of object sizes. The adjustable gripping forces enable it to handle both delicate and heavy objects. Below in The fingers of the gripper do not use any sensor data. The RG2 gripper provides antipodal distances between gripper fingers as feedback signals to the agent. This allows for checking of grasping success and determining if a grasp was successful or not. It is assumed that a grasp is successful if the feedback signal from the

RG2 gripper for finger antipodal distances does not change at any time during a grasping attempt. This assumption is true under the condition that the gripper constantly applies force on the grasped object. Under this condition, if the grasped object falls the gripper would close, therefore changing the distance between gripper fingers.

Table 2: Technical data for the RG2 gripper (RG2 Gripper Datasheet, 2015).

Technical data	Min	Typical	Max	Units
Total stroke (adjustable)	0	-	110	[mm]
Finger position resolution	0	0.1	-	[mm]
Repetition accuracy	-	0.1	0.2	[mm]
Reversing backlash	0.2	0.4	0.6	[mm]
Gripping force (adjustable)	3	-	40	[N]
Gripping force accuracy	+/-0.05	+/-1	+/-2	[N]
Gripping speed	55	110	184	[mm/s]
Gripping time	0.04	0.07	0.11	[s]
Operating voltage	10	24	26	[V DC]
Power consumption	1.9	-	14.4	[W]
Maximum current	25	-	600	[mA]
Product Weight	-	0.65	-	[kg]

3.12 CoppeliaSim

For the experiments, the CoppeliaSim platform, formerly known as V-REP, from Coppelia Robotics AG was used (Rohmer et al., 2013). CoppeliaSim is a free educational use platform that includes Python accessible via a remote API. Multiple models of robotic platforms are available in CoppeliaSim including the UR5 arm. The simulation environment was run using an NVIDIA GTX 1050Ti for computing and an Intel Core i9-10920X processor.

The simulation configuration utilizes a UR5 robotic arm manipulator with an RG2 gripper in CoppeliaSim with Bullet Physics 2.83 for dynamics. The internal inverse kinematics module of CoppeliaSim was utilized for robot motion planning. A statically mounted perspective 3D camera is simulated in the environment to capture perception data from an allocentric point of view. The resulting perception data is then rendered into RGB-D images of resolution 640 × 480 with OpenGL from the camera without noise models for depth or colour. The workspace for the system covers a 0.448^2m tabletop. Boundaries are placed around the workspace to limit failures from objects rolling outside of the camera's view. These boundaries are also meant to limit the negative impact of continuous pushing that may also remove objects from the agent's point of view. An example of the simulation setup for an intense natural clutter scenario is available in Figure 8.

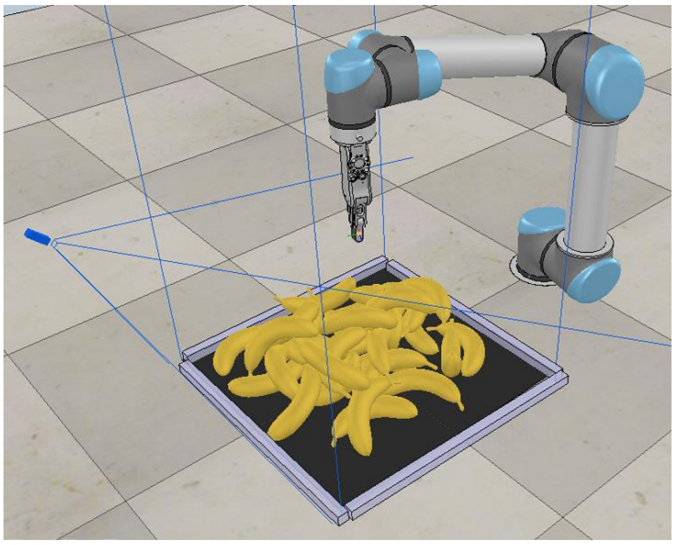


Figure 8: The simulation experimental setup for cluttered object clearance with pushing and grasping actions of the UR5 robot in CoppeliaSim. A simulated RGB-D camera is placed for visual inputs from an allocentric view of the workspace opposite of the UR5 arm.

PyTorch was used to program and train the neural networks. PyTorch can be programmed in Python which makes the connection between simulation and neural network structure easily achievable. PyTorch can also use a GPU to train the networks, which is a necessity due to the computational complexity of the reinforcement learning Q-mapping. The GPU usage unloads a lot of processing power from the CPU which is needed for processing the simulations. The declarative data parallelism of PyTorch means that multiple GPUs can be leveraged in training with minimal effort, which could further decrease the training times of neural networks. For this thesis, only one GPU was leveraged so the data parallelism was not fully taken advantage of.

3.13 Evaluation

To assess the different agents' performances, the speed and accuracy at which objects are picked up were examined with emphasis on task completion. Each trial is a new clutter of N objects in the workspace and the robot must remove all objects through grasping. Each test is executed for 30 trials or repetitions and the average evaluation scores are computed. These metrics will help identify training and reward strategies that work best for specific applications of the grasping and pushing agents. The performances are quantitatively evaluated with four evaluation metrics:

- **Completion:** a test trial is considered completed when the robotic agent successfully picks up every object without 10 consecutively failed primitive actions. This demonstrates the ability of the agent to finish the task and clear the workspace of objects. This is considered the primary metric of performance because, in industry-specific applications, total workspace clearance is the goal of bin picking and a robotic agent not capable of this will result in leftover objects creating inefficiencies in processing lines.

$$Completion\ Rate = \frac{Number\ of\ Cleared\ Trials}{30} * 100 \qquad [\%] \qquad (19)$$

- **Grasp Success (GS):** the ratio of all successful grasps in all grasp attempts to measure the accuracy of the grasping policy. All trials including uncompleted trials are considered to understand the impact of unfinished tasks on grasp success. Measures the accuracy of the grasping policy, while also considering which action choices led to uncompleted trials.

$$Grasp\ Success\ Rate = \frac{Number\ of\ Successful\ Grasps}{Number\ of\ Grasp\ Attempts} * 100 \qquad [\%] \qquad (20)$$

- **Action Efficiency (AE):** the ratio of the number of objects N to the number of actions executed to reach completion. Uncompleted trials are also taken into consideration to examine how the pushing policies impact task completion.

$$Action\ Efficiency\ Rate = \frac{Number\ of\ Objects\ in\ Test}{Number\ of\ Actions\ Before\ Completion} * 100 \qquad [\%] \qquad (21)$$

- **Grasp to Push Ratio (GtP):** the ratio of all grasp attempts to the total number of motion primitive attempts (both pushing and grasping actions). The frequency of grasping attempts to all actions will help identify reasons for failed task completion and create a stronger link between pushing and task completion.

$$Grasp\ to\ Push\ Ratio = \frac{Number\ of\ Grasp\ Attempts}{Number\ of\ Actions} * 100 \qquad [\%] \qquad (22)$$

3.14 Experiments

In this section, the methodology for how the simulation experiments were conducted within CoppeliaSim is discussed. In the simulation, two separate experiments regarding pushing reward structure and training object variation were conducted to see how best to modify the default method for a specific agro-food case. The default method or baseline for comparison is the VPG policy originally implemented by Zeng et al. 2018. In the pushing experiment, three variations of the pushing reward system were compared with the state-of-the-art method. For the training object variations, four different sets of training objects were used to train the pushing and grasping policies. The effect of the different training sets was compared with the default training to find the most appropriate training objects for greatest performance on agro-food objects while still maintaining an ability to generalize on novel arrangements not encountered within training.

3.14.1 Pushing Experiment

Several experiments were conducted to determine whether the pushing motion primitive is an effective method for enabling the grasping of objects that have a complex piling structure. For this, the performance of the default method will be evaluated and compared with three alternative reward structures for the pushing action and its respective policy. The default reward structure, which will be referred to as VPG, can be found in Equation (15).

- **Pushing No Reward (PNR):** The first alternative will be an agent that does not couple a reward to the pushing network, to be referred to as the Pushing No Reward (PNR) agent. In this case, all successful pushes that create a noticeable change in the robotic workspace receive no reward.

$$R_p(s_t, s_{t+1}) = \begin{cases} 0, if \sum(s_{t+1} - s_t) > \tau\,, \\ 0, otherwise. \end{cases} \qquad [-] \qquad (23)$$

- **Pushing Low Reward (PLR):** The second reward structure includes a pushing policy that returns a reduced reward compared to the default method. This method will be referred to as the Pushing Low Reward (PLR) agent. For this agent, the reward for successful pushes will be half that of the default agent.

$$R_p(s_t, s_{t+1}) = \begin{cases} 0.25, if \sum(s_{t+1} - s_t) > \tau\,, \\ 0, otherwise. \end{cases} \qquad [-] \qquad (24)$$

- **Higher Change Threshold (HCT):** The final alternative agent had the same reward value as the default VPG, but the way the reward was given when the agent is successful at performing the action was modified. In this reward structure, the noticeable change in the workspace caused by a successful push must create twice as much of an impact as the default case. The reason for this formulation is to create more difficult criteria for what is deemed a successful push allowing the agent to learn to only opt for pushing actions that are more likely to create space for grasping attempts and reduce overfitting to the pushing policy and its respective reward.

$$R_p(s_t, s_{t+1}) = \begin{cases} 0.5, if \sum(s_{t+1} - s_t) > 2 * \tau\,, \\ 0, otherwise. \end{cases} \qquad [-] \qquad (25)$$

Training Details

The four agents were trained with the same eight blocks with rigid geometries in a 10-object cluttered environment. In the 10-object training clutter, grasping and pushing actions were performed on episodes of randomly placed wooden blocks in the workspace delimited by grey borders. The agent automatically performs data collection through trial and error until the scene is empty of objects. When the scene is empty, a new 10-object random clutter is generated. This process was repeated until the agents were trained to 2500 action primitive iterations.

Testing Details

Once the agents are trained to 2500 iterations or motion primitives, each was evaluated on eleven different challenging 2D block arrangements referred to as Block Adversarial or BA test cases. These challenging scenarios are meant to represent adversarial clutter that was manually created to represent real-world picking scenarios like tightly packed boxes. Each test includes 3-6 objects placed in the workspace in front of the robot. These test cases consist of objects laid closely side by side in positions and orientations that even the best grasping policy would struggle with decluttering emphasizing the need for a secondary action to separate the arrangement before a single object can be grasped successfully. A single object is placed separately from the others as a sanity check for the agent’s learned policies. If an agent is not able to grasp the isolated object, then the agent’s policies have not been sufficiently trained and further training iterations are required. The BA test case scenes can be found in Figure 9.

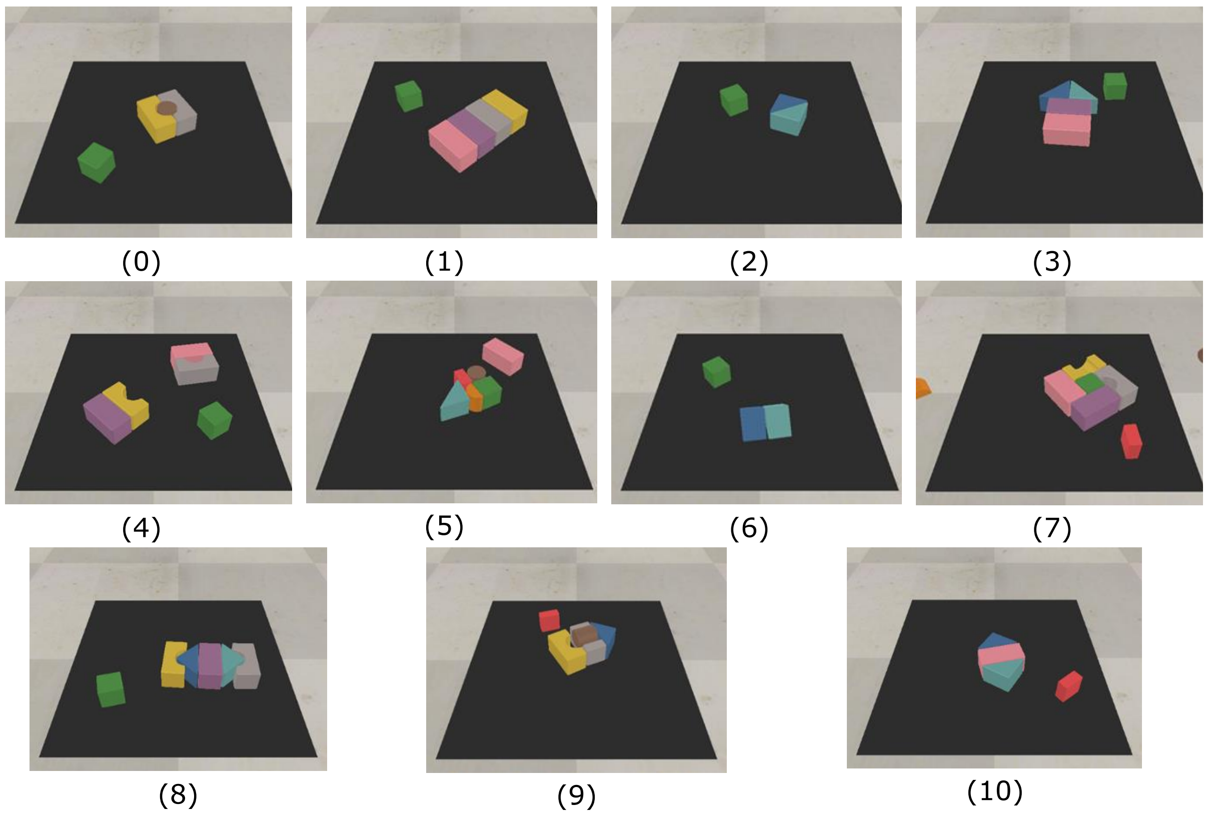


Figure 9: The eleven test cases for block adversarial (BA) clutter that is used as a baseline evaluation with VPG with each test trials scene index (0-10). All agents from pushing experiments, training object regimen experiments, and final AFV are evaluated on the BA cases.

3.14.2 Training Regimen Experiment

The second experiment performed in this thesis consisted of training the default VPG method on different sets of training objects to see the impact on final performance. In this experiment, the reward structure for pushing is not changed and the default setting for VPG is the same for all model comparisons, with the only difference between the trained agents being how they were trained.

Training Details

During training, the three new training object regimens and default VPG used the same episodic repetition of object scenes as previously described in the pushing experiment training detail in section 3.14.1.

- **10-Block:** The default training method uses the same set of training objects used in the previous pushing experiments with 10-object block random arrangements. This training method will be referred to as the 10-Block agent.
- **30-Block:** The first modified training method included the same eight wooden blocks but increased the number of training objects from 10 to 30 to represent a training regimen of intense natural clutter. This training regimen will be referred to as the 30-Block agent.
- **Block+Crop:** The second modified training regimen consisted of a 10-object scene like the default training scheme but includes two new crop shapes (banana and carrot). This increased the number of objects shape available for random scene generation from eight to ten. Having both block and crop training objects, this agent will be referred to as the Block+Crop regimen.

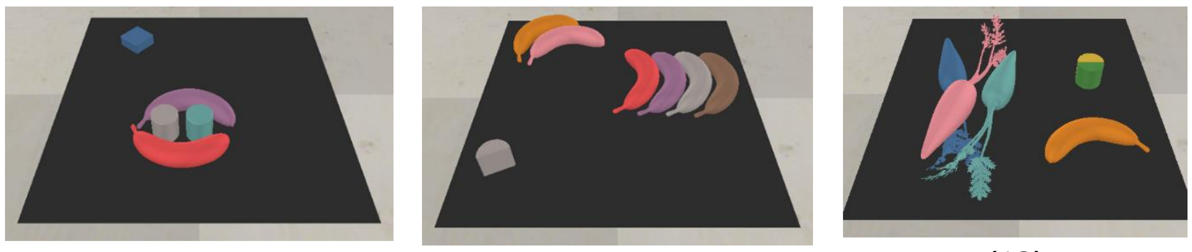
- **Crop-only:** The third modified training regimen reduces the number of objects shape available for random scene arrangement from eight in the default case to only the two crop objects. This agent is called the Crop-only regimen.
- **Same-Colour:** A final training regimen used the same training objects as the Block+Crop scenario, but all the training objects were the same colour. This Same-Colour training scenario was created to see the impact of training on objects all of the same colour, which would happen in the food processing industry (e.g., training a robot agent with only chicken/bananas in the real world the agent could not rely on colour information to differentiate the cluttered objects from one another).

These training object variations were performed to see how best to modify the default VPG training method for a specific agro-food case where only one type of product is meant to be manipulated in a scene. Each method was trained through trial and error to 2500 action primitives.

Testing Details

The four different training methods were then evaluated on the same eleven adversarial clutter arrangements described in the pushing experiment (BA test cases). These tests showed the generalization ability of the new and specific crop only training regimen to adversarial clutter scenarios not encountered during training.

In addition to these BA test cases, the four training regimens were evaluated on three manually engineering crop adversarial arrangements shown in Figure 10. Allowed for examination of specificity and generalization trade-off between simple and more challenging training scenarios. These test cases will be referred to as Block Crop Adversarial (BCA) test cases.



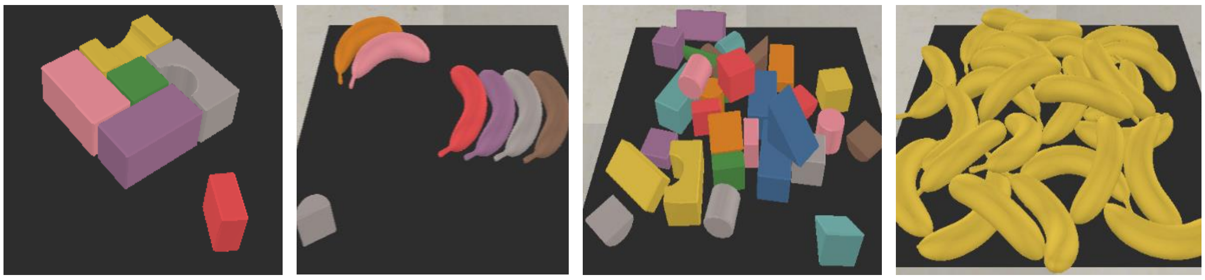
(11) (12) (13)

Figure 10: The three block and crop adversarial (BCA) clutter cases that are used to examine the specificity and generalization trade-off of training with more specific shapes that lack rigid geometries like in the BA cases (Figure 9). All agents from training object regimen experiments including VPG and the final AFV are evaluated on the BCA cases

3.14.3 Agro-Food Variant (AFV) Agent

From the pushing and training regimen experiments, the modifications in training and reward structure that resulted in the best performance compared to the default VPG agent were combined for a final Agro-Food Variant agent (AFV) of VPG. The training of the AFV agent included the training regimen that performed best in the training regimen experiments of adversarial clutter. Likewise, the pushing reward structure that improved upon VPG was included.

To evaluate the performances of AFV compared to VPG, the two agent formulations were tested on all previous BA and BCA tests. Another evaluation on intense clutter of 30 block objects was used to see the performance of the policies on natural clutter. A final evaluation was also performed to represent an agro-food specific scenario of heavy clutter bin picking that includes 30 bananas clutter with random arrangements. Examples of these cases are illustrated in Figure 11.



(a) **(b)** **(c)** **(d)**

Figure 11: (a) Adversarial clutters used by Zeng, et al. 2018 for fair comparison (BA test cases). All BA test scenes available in Figure 9*. (b) Manually engineered adversarial clutter with crop object shapes (BCA test cases). All BCA test scenes available in* Figure 10*. (c) Randomly generated block clutter with 30 objects (30 block test). (d) Randomly generated banana clutter for agro-food industry-specific clutter situation (30 banana test).*

*Table 3: Summary table of all variations of VPG and what specifically has been changed regarding the pushing reward system and training methodology. Test cases that will be used as evaluation tools for each agent variation are listed. *The τ_{old} is in reference to Equation* (15) *representing the manually set threshold for determining if a push was successful. **All training objects were yellow.*

Agent Name	**Push Reward**	**Training Object Shapes**	**Number Training Objects**	**Test Cases**
		Pushing Experiment		
VPG	+0.5	8 blocks	10	11 BA
PNR	+0.0	8 blocks	10	11 BA
PLR	+0.25	8 blocks	10	11 BA
HC T	+0.5 $(\tau_{new} = 2\tau_{old})$*	8 blocks	10	11 BA
		Training Regimen Experiment		
10-Block	+0.5	8 blocks	10	11 BA, 3 BCA
30-Block	+0.5	8 blocks	30	11 BA, 3 BCA
Block+Crop	+0.5	8 blocks, 2 crops	10	11 BA, 3 BCA
Crop-only	+0.5	2 crops	10	11 BA, 3 BCA
Same-Colour	+0.5	8 blocks, 2 crop**	10	11 BA, 3 BCA
		Final Agent Experiment		

VPG	+0.5	8 blocks	10	11 BA, 3 BCA, 30-block test, 30-banana test
AFV	+0.0 ($\tau_{new} = 2\tau_{old}$)*	8 blocks, 2 crop	30	11 BA, 3 BCA, 30-block test, 30-banana test

4 Results

In this section, the results of the experiments that were performed will be presented. In the first part, the results of the pushing reward experiments will be presented followed by the results of the training object regimen experiments and ending with the final comparisons of the default VPG method and the new AFV agent.

4.1 Pushing Experiment

For each agent variation of the pushing reward function, training was performed to 2500 iterations of motion primitive actions on the same set of 10 randomly placed block objects. The goal of this experiment was to assess the utility of pushing actions in the most extreme cases where decluttering is necessary. Therefore, the agents trained for the pushing experiments were only evaluated on the manually engineered block adversarial (BA) tests cases. All scenes of the BA test cases can be found in Figure 9. The training objects remained the same for all agent variations (Table 1).

4.1.1 Training Results

To begin this study, the results of the training performance of each variant was viewed on a plot of grasping performance versus training steps. The grasping performance is measured by the percent grasp success rate over the last 200 grasp attempts indicated by the graph in Figure 12. Numbers reported at earlier training steps (for example, *i < 200*) are weight by i/200. Since there is no defacto way to measure the quality of pushing motions for how well they benefit a model-free grasping policy, a second set of plots were made to display the grasp-to-push (GtP) ratio over the same training period with grasp performance for the VPG, PNR, and HCT agents (Figure 13). By doing so, periods of changing pushing frequencies, as indicated by lower GtP, can be related to changes in grasp success rate.

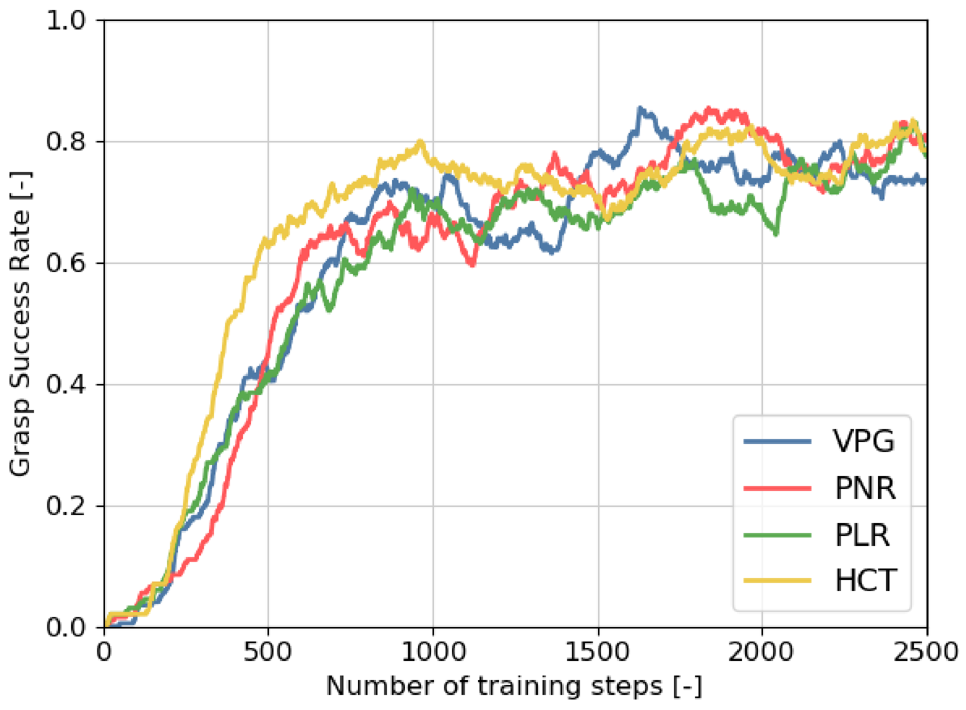


Figure 12: Comparing the performance of VPG policies trained with varied reward systems for pushing. Solid lines for each agent indicate percent grasping success rates.

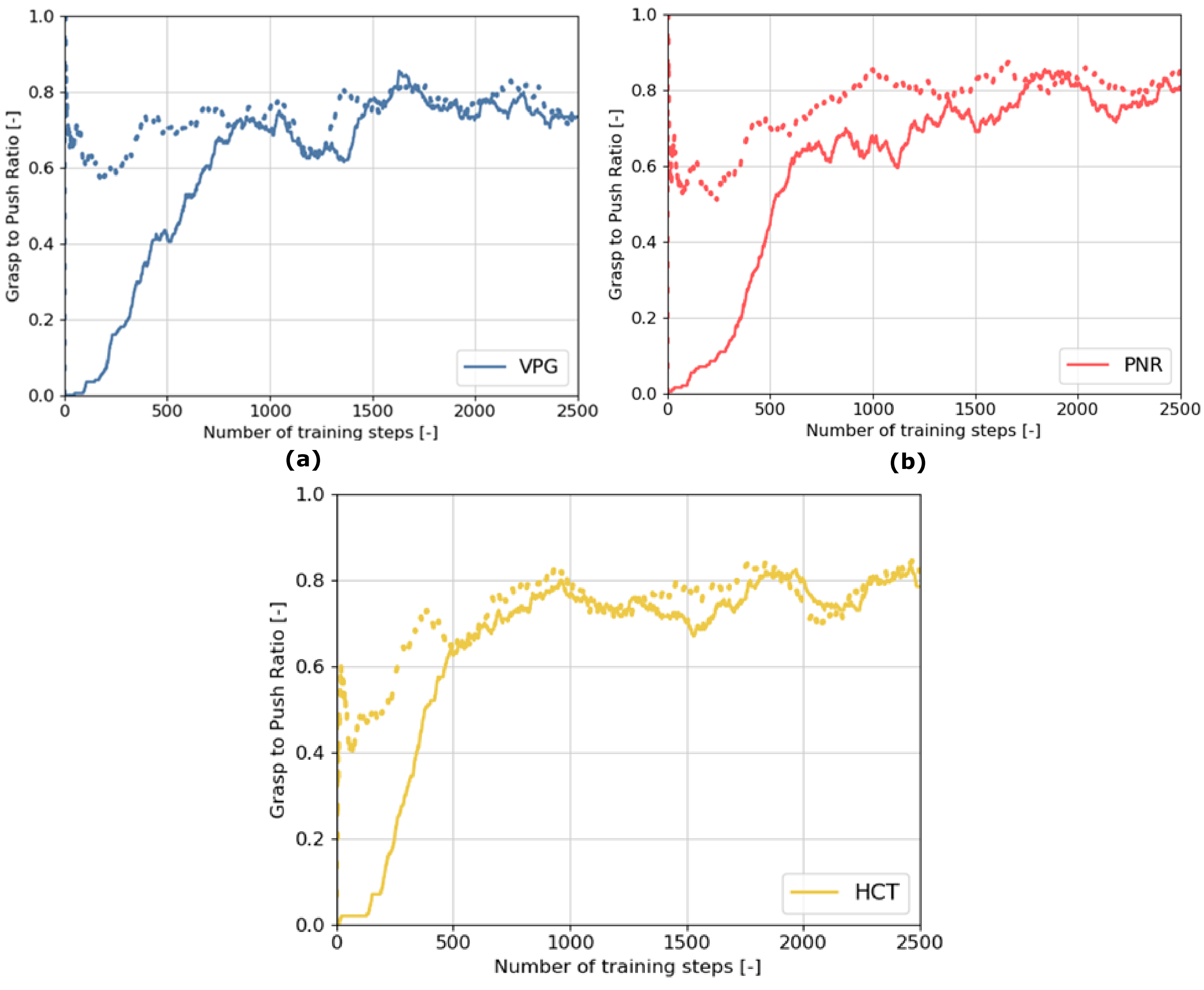


Figure 13: Grasp success rate lines (solid lines) overlaid by grasp to push ratios (dotted lines) of (a) Visual Pushing for Grasping, VPG (b) Pushing No Reward, PNR and (c) Higher Change Threshold, HCT.

From these training results, all the variants are capable of learning effective pushing and grasping policies and can achieve grasp success rates between 70-80%. From the grasp success plots, not many clear differences between the pushing reward variants can be identified. However, it is worth mentioning that the most stable and quickest to converge to 80% grasp success is the HCT variant. This stable training suggests that a higher manually set threshold for image scene detection helps the policy learn more effective and less redundant pushing actions than VPG with a threshold half the size. The rate of improvement in grasp success for PLR and VPG is the most erratic.

In Figure 13, the graphs show changing grasp success rate alongside the grasp to push ratio. The grasp to push ratio illustrates the frequency of grasping actions to total actions executed by the agents as illustrated in Equation (22). Plot (a) shows that the baseline method, VPG, has the largest variations of GtP ratio during training. Decreases in grasp success during the training of VPG, correspond to decreases in the GtP ratio as well. After 1000 training steps, all variants besides VPG have gradual convergence towards 80%

grasp success rate with minor fluctuations in GtP. In contrast, VPG at around 1000 training steps has a reduction in grasp success for almost 400 training steps. On the GtP plot in Figure 13(a) this reduction in grasp success corresponds to the lowest grasp-to-push ratio of all variants after 1000 iterations. These overlapping periods of decreased grasp success and decreased grasp frequency suggest an antagonistic relation between improved grasp success and encouraging more frequent pushing actions.

4.1.2 Block Adversarial (BA) Test Results

After these training observations, VPG was compared to the three pushing reward variants in simulation on the eleven BA test cases. Scene images and indexes of the eleven cases can be found in Figure 9. Each test case consisted of a configuration of 3-6 objects placed in the workspace in front of the robot. These configurations remain exclusive of the training procedure and reflect the most challenging picking scenarios. Across all these cases objects are laid side by side, in positions that even optimal grasping policies would be challenged without singulation of objects first. A single isolated object is also included to provide a sanity check if the agent has been properly trained. If a policy cannot grasp the isolated object, more iterations in training are necessary. Individual scene results for each block adversarial arrangement are portrayed in Figure 14. The average results are illustrated in Table 4.

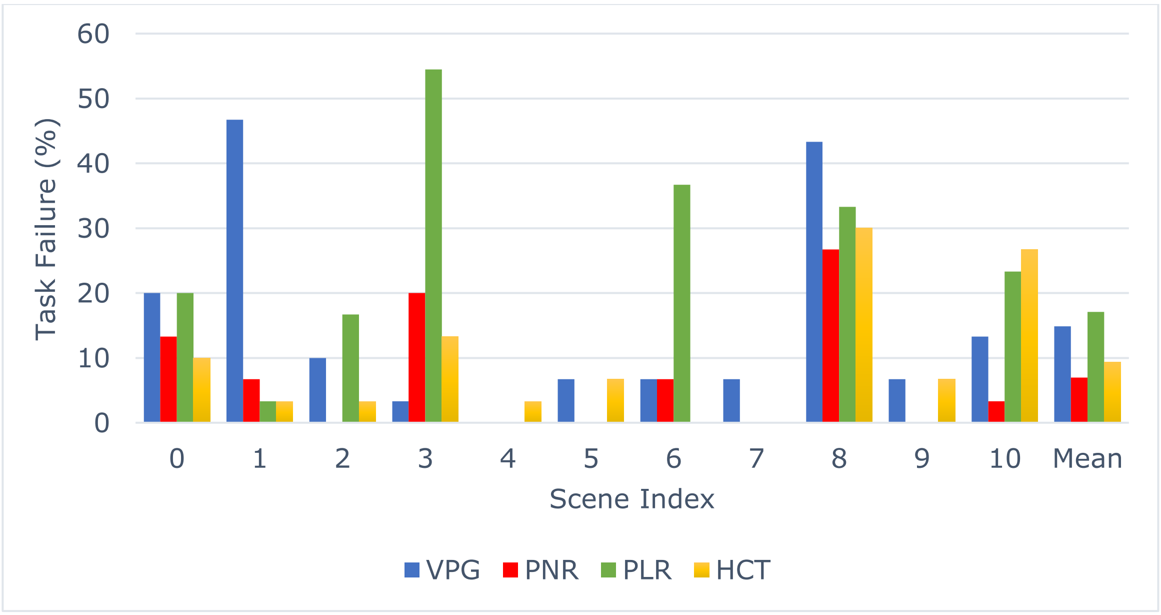


Figure 14: Failure rates out of 30 repetitions of each block adversarial test case scene. The scene images for each scene index on the x-axis can be found in Figure 9 *with the corresponding number.*

Table 4: Block adversarial task results with average percent completion rate, grasp success rate (GS), action efficiency (AE), and grasp-to-push (GtP) ratio for the four agents with different pushing reward functions.

Agent Policy	Completion (%)	GS (%)	AE (%)	GtP (%)
VPG	85.2	50.4	44.0	83.4
PNR	93.0	47.0	47.4	99.0
PLR	82.9	43.0	42.7	92.3

HCT	90.6	62.3	47.7	74.0

VPG: From the completion results, it is observed that the addition of a pushing reward slightly decreases the set of scenarios for which successful grasping can be performed. The policy with the second lowest performance is seen from the VPG baseline where it struggles to complete the picking tasks about 15% of the time. In particular, the VPG policy struggles in scenarios where large cuboids are laid closely side-by-side, scenes 1 and 8, with failure rates exceeding 40%. From the trends seen in Figure 15, the inclusion of an intrinsic reward for pushing leads to an increase in pushing frequency.

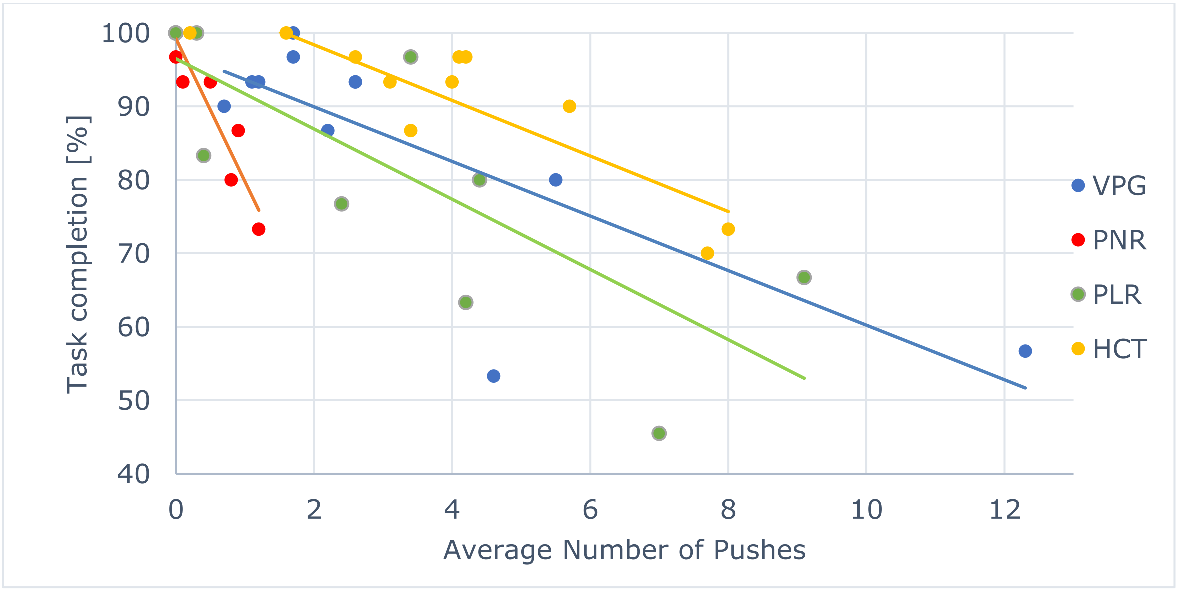


Figure 15: Task completion percentage of all eleven BA test cases plotted with respect to the average number of pushes per task. Linear trendlines were added for better visualization of how variations in pushing frequency compare to task completion.

PNR: The overall completion rate increases by approximately 8% when no pushing reward is included represented by policy PNR. These results show that without a pushing reward the policy can more easily complete the tasks despite a reduction in grasp success. Since no reward is attached to pushing, grasping is the dominant action chosen by this policy illustrated by the grasp to push ratio of 99.0%. For all tests, PNR without the intrinsic reward has an average push count between 0 and 1.7 pushes over the test cases. Additionally, the resistance to failure modes seen by the high completion rate across all eleven test cases in PNR indicates that without an intrinsic reward for pushing the policy can handle a larger set of complex and challenging scenarios of clutter than it can with an intrinsic reward. Due to this observation, the intrinsic reward will be removed from the final AFV policy.

PLR: Across the test cases, the PLR with a reduced pushing reward struggles the most to complete the picking tasks. This policy experiences the lowest completion, grasp success, and action efficiency of the four variants.

HCT: The improvements in completion rate, grasp success, and action efficiency of HCT indicate that the new threshold, τ_{new}, helps the agent learn a more accurate pushing policy without the redundant failure-prone pushes that result in lower completion rates of the VPG policy. This new threshold will be incorporated into the final AFV policy.

In total 44 test cases were performed across the four pushing reward variants (11 BA tests for each reward variant). 12 out of 44 test cases result in 100% completion. Of these 12 completed test cases, 8 reached completion with an average number of pushes per trial of 0. The four remaining fully completed tests had an average number of pushes between 0.2-1.7 per trial. All test cases where the average number of pushes exceeds 1.7 are not capable of clearing all 30 trials of the respective test. Clear trends are observable from the scatterplot of average pushes per trial versus test completion rate shown below Figure 15. As pushing increases in frequency regardless of trial configuration, the task completion rate for all four reward structures decreases.

The other three policies with the intrinsic reward have pushing counts ranging from 0 to 13 per test case. Regardless of the pushing reward system, 90% completion of a test case is only seen when the pushing frequency per trial is less than 6. It is not clear however if pushing frequency directly results in reduced task completion, but the HCT policy gives evidence that a more properly made reward criterion encourages pushing that is non-redundant and more resistant to failures seen by the standard VPG reward criterion.

The improvements in completion rate, grasp success, and action efficiency of HCT indicate that the new threshold, τ_{new}, helps the agent learn a more accurate pushing policy without the redundant failure-prone pushes that result in lower completion rates of the VPG policy. This new threshold will be incorporated into the final AFV policy. Additionally, the resistance to failure modes seen by the high completion rate across all eleven test cases in PNR indicates that without an intrinsic reward for pushing the policy can handle a larger set of complex and challenging scenarios of clutter than it can with an intrinsic reward. Due to this observation, the intrinsic reward will be removed from the final AFV policy.

4.2 Training Regimen Results

4.2.1 Training Results

After the pushing reward experiments, the impact of the training objects used to train the pushing and grasping policies were explored. Figure 16 shows the results of grasping success rate throughout the training of the five different scenarios. The performance of 10-Block over the training period has already been explained in the previous pushing experiment (refer to VPG in Figure 12).

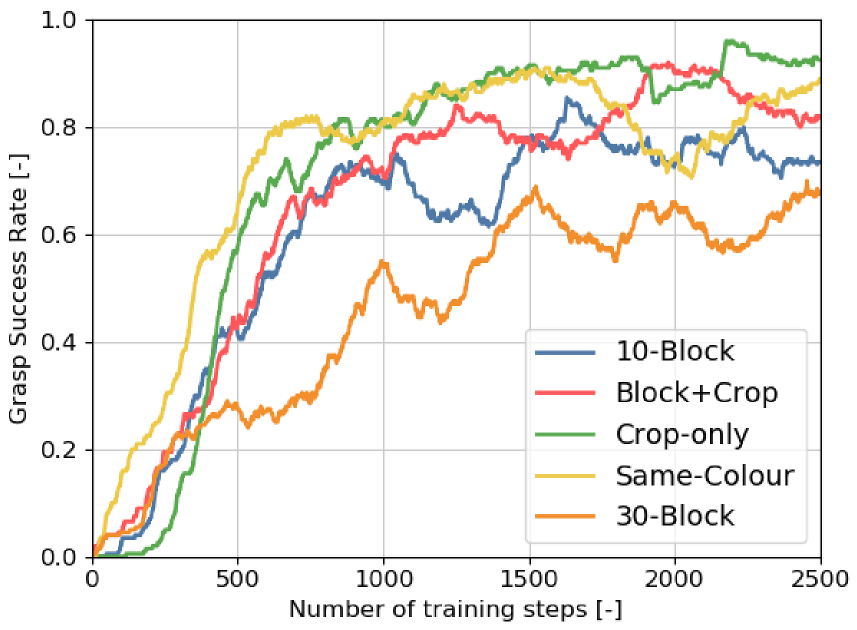


Figure 16: Comparing the performance of VPG policies trained with varied training objects.

Block+Crop: Final grasping performance for the Block+Crop object training scenario has an improvement of approximately 10% when compared with the standard 10-Block training agent. By including two irregularly shaped agro-food objects with the block objects with rigid geometries, better grasping can be achieved after 2500 iterations. The irregular crop objects may increase affordance values for grasping as the non-rigid shapes are less prone to falling into tightly packed challenging arrangements characteristic of the BA test cases. Introducing these complex shapes increases final grasp performance and allows for more stable learning of the policies. The stable learning can be seen when comparing the Block+Crop training performance with the 10-Block agent. The line of the Block+Crop policy has few moments of reductions in grasp success during training, but 10-Block has periods of decline in grasp success rate explained earlier because of redundant pushing. This redundant pushing appears to be less of a concern when objects without rigid geometries are included in training. Evidenced by the lines of Block+Crop, Crop-only, and Same-Colour when few reductions in grasp success occur.

Crop-only: The policy that has the quickest learning rate and highest grasp performance at the end of 2500 training steps is the Crop-only trained policy. For this agent, the policy for grasping is quickly learned because of the low number of object shapes used for training. The grasping policy is quickly learned by the agent and able to achieve grasping success rates of 90% on its training objects.

Same-Colour In the Same-Colour policy, the objects that were used for training were all the same colours. By doing this, the colour channels of the FCNs were not properly trained. Because of this, the agent cannot differentiate objects based on colour differences, and piled objects or objects close to one another are acted upon as if they are a single object. At the beginning of training, this directs the agent to attempt grasp actions at a much higher rate than any of the other agents. With limited pushing during the exploration phase (before 1000 training steps), the Same-Colour agent quickly learns the grasping policy with a high success rate.

30-Block: Both the 10-Block and 30-Block agents have regular periods during training that represent worsening grasp success. For 10-Block, this occurs between 1000 and

1500 training iterations and again for the last 1000 training steps as grasp performance gradually declines. For 30-Block, a period of worsening grasp success occurs three times between 1000 and 2500. When viewing the training performance of VPG with 30 objects, the learning rate at which the grasp success rate improves is much lower than the four other policies. This suggests that more than 2500 training steps are necessary for the policy to learn how to use collaborative pushing and grasping actions to manipulate and clear an intensely cluttered scene.

4.2.2 Block Adversarial (BA) Test Results

The individual results of the BA test cases for the training object regimens can be seen below in Figure 17. The results of the Crop-only trained policy are not included in the figure because of low completion rates. The results of these test cases for the Crop-only training scenario will be discussed later. Individual results for the 10-Block method of training are the same as they were for the pushing experiment (refer to VPG in Figure 14). The averaged results of task completion, grasp success, action efficiency, and grasp to push ratio are summarized in Table 5.

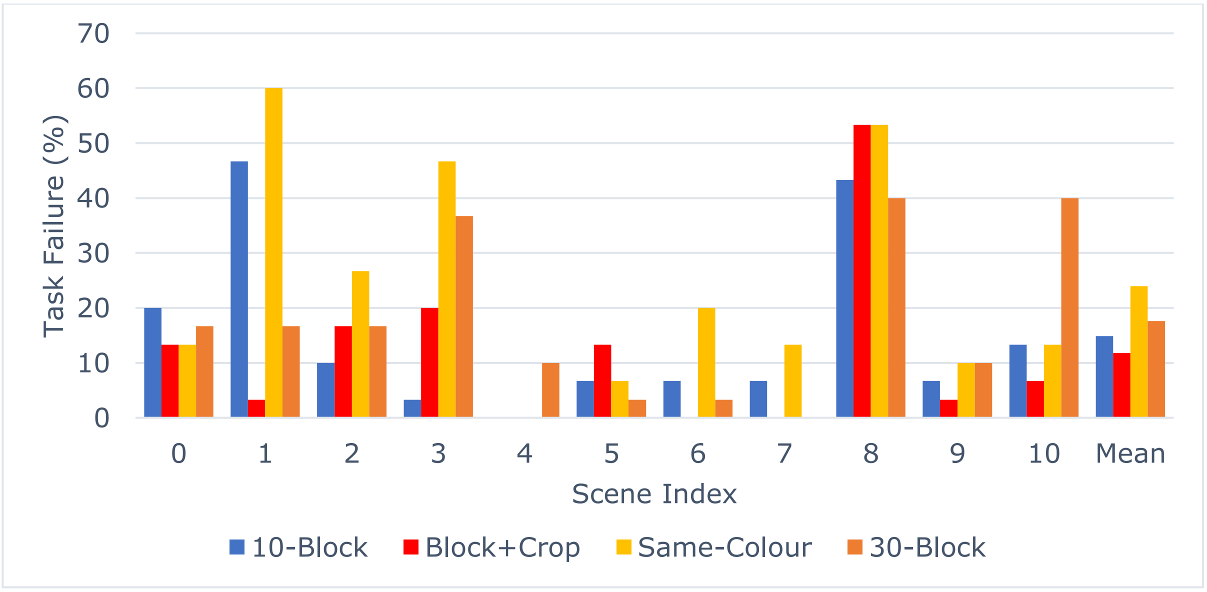


Figure 17: Failure rates out of 30 repetitions of each block adversarial test case scene. The scene images for each scene index on the x-axis can be found in Figure 9 *with the corresponding number.*

Table 5: Block adversarial task results with average percent completion rate, grasp success rate (GS), action efficiency (AE), and grasp-to-push (GtP) ratio for the four agents with different pushing reward functions.

Agent Policy	Completion (%)	GS (%)	AE (%)	GtP (%)
10-Block	85.2	50.4	44.0	83.4
Block+Crop	88.2	44.7	40.9	85.3
Crop-only	4.9	12.9	23.4	80.4
Same-Colour	76.1	48.7	33.5	61.8
30-Block	82.4	55.0	41.9	73.0

Block+Crop: In the Block+Crop training regimen, the failure rate exceeds 20% for one scene index, 8 which all other policies in this experiment struggle with resulting in failure occurrences above 40%. This scene contains large cuboid shapes closely packed together and saw high failure rates in the pushing experiment as well. Overall, this expanded training set has the highest completion rate compared to all other training policies,

Crop-only: As mentioned earlier, the Crop-only policy has the lowest completion rate on the eleven BA test cases with an average completion rate of 4.9%. The Crop-only policy never encountered the rigid block objects during training, which illustrates that the policy lacks the generalization ability to new objects and arrangements that is a strong characteristic of the VPG policy (Zeng et al. 2018A). Low completion rates by the policy are further reflected by poor grasp success and action efficiency.

Same-Colour: When comparing the Same-Colour training data in Figure 16 with the failure rates seen in Figure 17 a discrepancy in performance can be seen. Despite having the second-best grasping performance after 2500 training iterations, the Same-Colour policy has the worst overall performance in completion rates (not including the Crop-only policy). The Same-Colour agent has failure rates above 20% for four of the eleven test scenes with the highest failure percentage of 60% on scene 1. On average the lowest performing training scenario in task completion is the Same-Colour regimen. This seems contradictory considering the high success rate during training, but since the colour channels of this agent have not been trained failure modes are occurring that prevent the policy from task completion. Out of the eleven BA tests, the Same-Colour trained agent does not have a single completion rate of 100%.

30-Block: The third best performing policy behind the 10-Block and Block+Crop policies is the 30-Block policy trained with 30 block objects as opposed to only 10. As seen in Figure 16, a reason for this lack of success in task completion could be explained by an insufficient amount of training iterations indicated by the 30-Block agent having the lowest grasping performance after training.

The best performing policy regarding completion rates is the Block+Crop regimen, but the grasp success rate is lower than the other policies (excluding Crop-only). The highest grasp success rate comes from the 30-Block policy. In terms of action efficiencies, the 10-Block, Block+Crop, and 30-Block trained policies all have similar performance. It is worth mentioning that the completion rates for all these training regimen policies do not reach 90% or higher. Whereas the removal of an intrinsic reward (PNR policy) or a more properly defined successful push (HCT policy) can enable an agent to achieve higher completion results on the BA test cases (see Table 4).

4.2.3 Block Crop Adversarial (BCA) Test Results

After the training regimen policies were evaluated on the BA cases, they were evaluated on three additional Block Crop adversarial (BCA) arrangements. Individual failure rates for each test case are displayed in Figure 18 and a summary of all performance metrics are provided in Table 6.

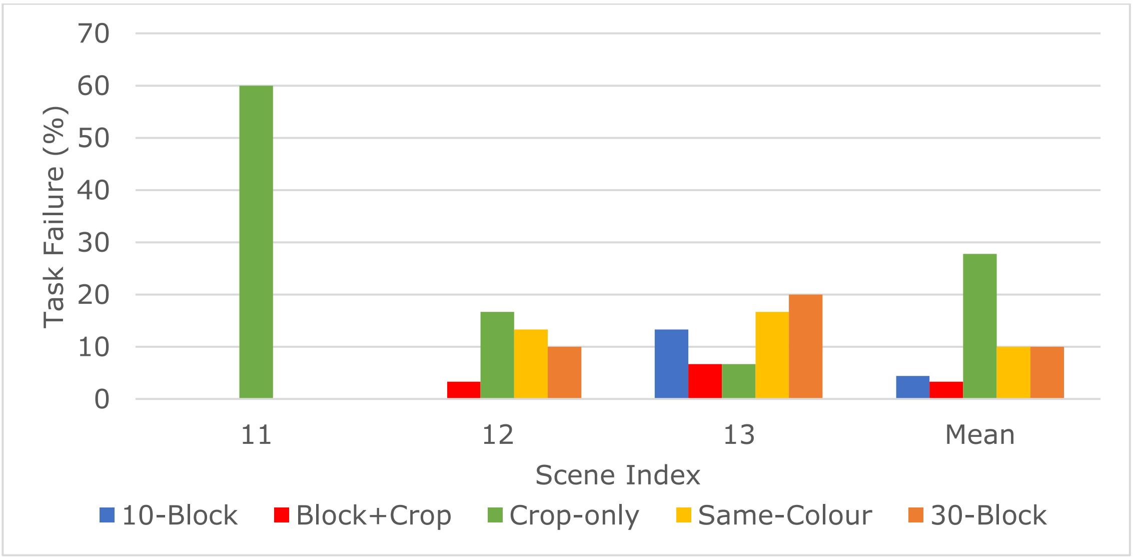


Figure 18: Completion results presented as failure rates for the individual Block Crop Adversarial test cases. The scene images for each scene index on the x-axis can be found in Figure 10 *with the corresponding number.*

Table 6: Block Crop Adversarial (BCA) task results with average percent completion rate, grasp success rate (GS), action efficiency (AE), and grasp-to-push (GtP) ratio for the four agents with different pushing reward functions.

Agent Policy	**Completion (%)**	**GS (%)**	**AE (%)**	**GtP (%)**
10-Block	95.6	77.3	72.4	94.8
Block+Crop	96.7	72.7	72.1	99.1
Crop-only	72.2	56.0	55.5	94.0
Same-Colour	90.0	71.6	67.7	93.6
30-Block	90.0	70.3	67.9	95.9

10-Block: The 10-Block baseline performs the best on scene 12 fully clearing all 30 repetitions, The performance of the 10-Block baseline drops slightly on scene 13, but it still has the second highest rate of completion for the three BCA cases. The successful performance of 10-Block on the BCA test cases highlights the strong generalization ability of the method to novel objects not encountered in training. Despite never learning how to grasp and push the unique crop objects, it can clear almost all the cases with similar completion rates to the policies that were trained with the crop objects. Even more, the grasp success rate for 10-Block, shown in Table 6, is the highest compared to all other trained policies.

Block+Crop: The Block+Crop trained agent failed to fully clear the workspace on one repetition of scene 12. The Block+Crop trained policy has a slight advantage when it came to the average completion of the BCA cases. A decrease in the grasp success rate of 4.6% can be seen when the grasp and push policies were trained with the expanded set of training objects. Despite the lower grasp success, the action efficiencies of the 10-Block and Block+Crop are similar meaning both can clear the training objects with a similar speed and number of actions. For both the BA and BCA tests, the Block+Crop policy opts for fewer pushes than the 10-Block trained policy seen by the higher grasp to push ratio. This suggests that when training with non-rigid shapes like the crop objects the expected future rewards for pushing determined by the Q-learning formulation are lower than they are when only rigid shapes are used. Specifically for the BCA cases, the grasp to push ratio increased for all policies. This gives further evidence that pushing is less necessary to properly grasp and clear crop objects from the scene.

Crop-only: Once again, the lowest performing policy is the Crop-only policy, but the completion rate increased from 4.9% on the eleven BA tests to 72.2% for the BCA tests. The policy had its lowest completion performance on scene 11, which of the BCA test cases has the largest amount of block objects. By not including rigid shapes in training, the generalization ability of VPG has been lost for this variant. It would be interesting to see how well this Crop-only policy could be generalized to non-rigid shapes like other crops not encountered during training. The Crop-only policy improved considerably on scene 12 where its performance is much closer to the other training regimen policies. For scene 13, the Crop only policy performs the best in terms of completion rate showing the policy can accurately handle workspace clutter when fewer block objects are included in the image scene. For scene 11, the Crop-only agent has a high failure rate while all other agents can reach 100% completion.

Same-Colour: In these test cases, the Same-Colour trained system does well compared to the earlier performance seen on the BA test cases (Table 5). The phenomena of the model not being able to identify the final object in the workspace does not occur in these test cases as they did previously. The Same-Colour agent can complete almost all tasks with an average failure of 10%. The reasoning for this discrepancy in performance between the BA tests and BCA tests is not entirely clear but could potentially be a result of the addition of the crop training objects. Since a majority of the BCA test objects consist of the two crop objects which are much larger than the rigid block shapes, the colour DenseNet channels are potentially not needed to identify the last object in the workspace. If a singular crop object is present at the end of a test trial, the objects may be large enough to provide enough depth information to the FCNs for object recognition and subsequent clearance from the workspace.

30-Block: The 30-block agent had a dip in performance on all metrics in the BCA cases compared to 10-block VPG. This is surprising considering how similar their performance was on the BA test cases, but when remembering the relatively low training performance of 30-block VPG after 2500 it seems reasonable for 10-Block VPG to perform better.

To summarize the results of the Training Regimen Experiment, the best training performances were seen in the Crop-only, Same-Colour, and Block+Crop trained agents. The Crop-only system converged to the highest grasping success rate after 2500 compared to all others. Despite these successes in training, it had significantly low performance on all metrics for the BA test cases. Inversely, the results for the BCA tests of the Crop-only policy showed a large performance improvement. The Crop-only agent performance metrics for BCA were still low compared to other policies since it has a low generalization ability for handling objects not seen within training. The Same-Colour agent despite strong training performance failed at a high rate with the BA test cases due to lack of training on colour information, which may be necessary to clear smaller block objects. It did show much better performance when tasks with the BCA tests. The Block+Crop had the strongest performance across the test cases, both BA and BCA, except for in grasp success rate. In terms of object clearance and test completion, it performed the best, which as the primary metric of performance makes it the optimal training scenario for an agro-food specific application.

Apart from these observations, it was clear that 2500 iterations were not sufficient to train more complex clutter training scenarios as seen by the training performance of the 30-block VPG training regimen (Figure 16). However, the performance of the 30-block regimen on the test cases was quite comparable to the 10-block VPG. This comparable performance is reason enough to extend a 30-object training scenario to the final AFV variant since the training iterations are increased for the final comparison of AFV to VPG.

4.3 AFV versus VPG

The final comparison with VPG took the characteristics of the best performing policies from the previous two experiments and combined them for a single agent. This comparison was made to provide an answer to the main research question of this thesis to see effective strategies for improving visual pushing for grasping for a specific agro-food scenario. The modified agent, referred to as the Agro-Food Variant (AFV), had no intrinsic reward for pushing, an increased threshold for image change $\tau_{new} = 2\tau_{old}$, and was trained on 30 objects with a total of 10 different training object shapes (8 rigid block objects and 2 crop objects). These characteristics are summarized in *Table 3*. The training for AFV was done till 5000 iterations, which was necessary due to 2500 iterations not being sufficient to train pushing and grasping policies with more intensely cluttered training iterations. This is visually represented in Figure 16 where the 30-Block, intense clutter training scenario, had the lowest grasp performance at the end of 2500 iterations. Likewise, the baseline VPG policy was also trained to 5000 iterations for a fair comparison. After training, the two policies were evaluated on the BA and BCA cases. Two final evaluations were performed on intense clutter scenarios; one scene with 30 randomly arranged block objects and a second with 30 randomly arranged bananas.

4.3.1 Training Results

Figure 19 illustrates the results of the grasping performance over the 5000 training steps of the AFV and VPG policies. Similar to the results of the pushing experiments, AFV has a faster improvement rate of grasping success near the beginning of training. At around 750 training steps, the grasping performance of VPG surpasses that of AFV. After this point, the AFV policy experiences oscillations in grasping performance like the 30-Block training scenario seen in Figure 16. Despite these oscillations, the grasping performance of AFV continues to gradually increase till the end of training. The final performance of AFV is slightly lower than VPG, but there are moments during the training where the grasping performance of AFV exceeds VPG demonstrating that both policies grasping policies have been well trained.

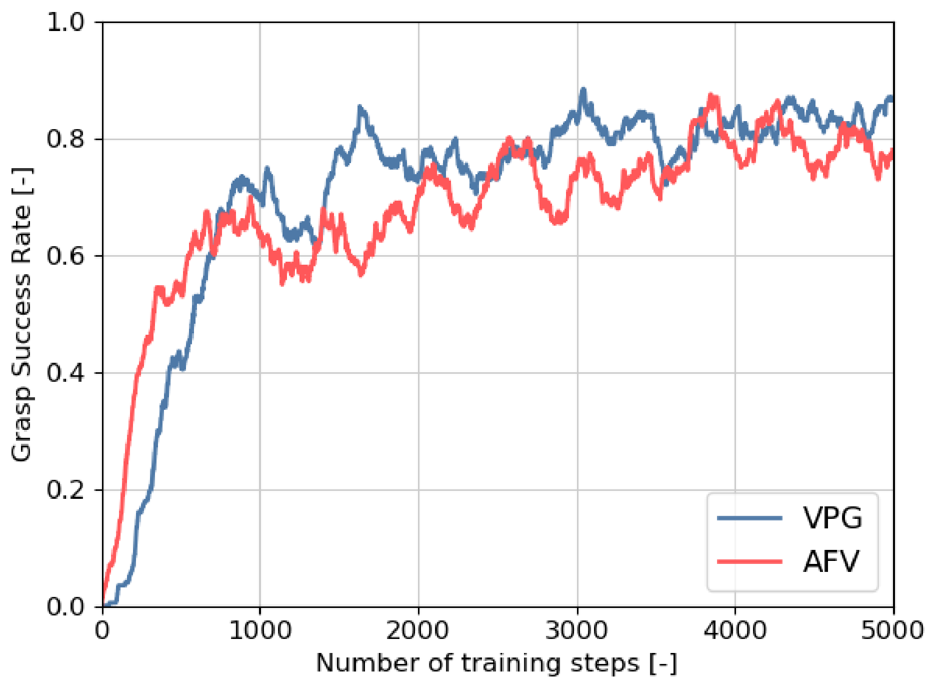


Figure 19: Grasp success performance of VPG and AFV over 5000 training iterations.

4.3.2 Block Adversarial (BA) Test Results

After training, the BA tests were used as an initial evaluation of the two policies. Individual test case results are available in Figure 20. A detailed summary table of the averaged evaluation metrics including grasp success rate, action efficiency, and grasp to push ratio can be found in Table 7. Across all test cases, the performance of VPG and AFV in terms of completion rate is similar. The only time the differences in success rate between the two policies exceeded 20% was for scene 6. In this test case, the AFV policy can clear all thirty repetitions of the scene whereas VPG fails to remove all objects on almost half the trials. AFV has its lowest performance on scenes 8 and 10. Of the eleven tests, AFV is capable of fully clearing all trials on five of the test cases. VPG on the other hand fully completes six of the eleven tests.

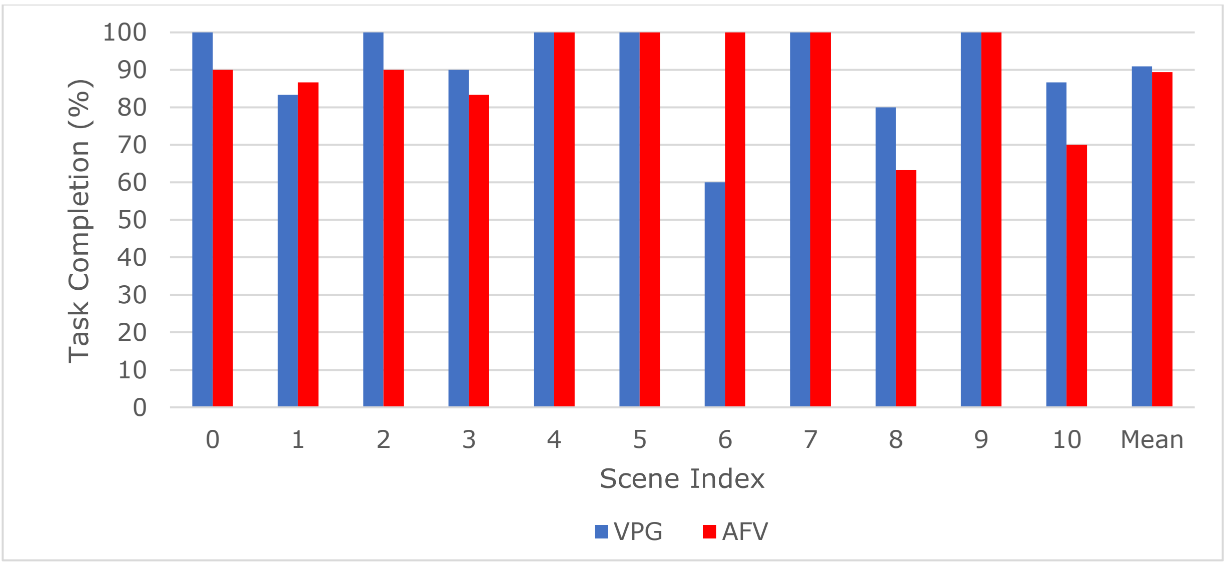


Figure 20: Individual scene completion rate results of the VPG and AFV policies when evaluated on the Block Adversarial (BA) test cases. The scene images for each scene index on the x-axis can be found in Figure 9 *with the corresponding number.*

To understand how the pushing actions can influence the completion rate, the average number of pushes for each test case was found and plotted against the test completion rate. These results are illustrated in Figure 21. Of the 22 test cases (11 test cases for each AFV and VPG), 11 resulted in full completion—5 by AFV and 6 by VPG. For the five tests that were fully completed by AFV, only one test case required the policy to use pushing for 100% completion. The pushing used was minimal; 0.2 pushes per trial meaning that only six pushing actions were used for all 30 repetitions of the scene which required a few hundred actions to fully complete. The other four tests saw the AFV fully clear all objects with zero pushes per trial. For the VPG policy, six of the test cases were fully completed. All six of these cases saw the VPG policy use zero pushes per trial, but full completion still occurred. Whenever pushing was used by VPG, the completion rate was 90% or lower. As illustrated by the graph below, as pushing frequency increases for both policies a trend can be seen in decreased completion rates.

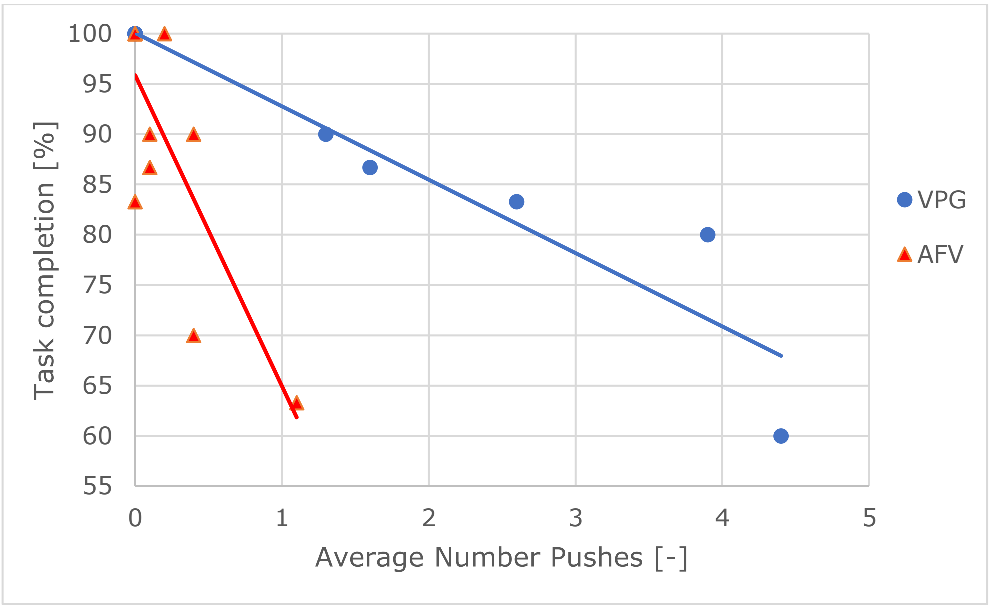


Figure 21: Test completion versus the average number of pushing actions for each test case of the AFV and VPG policies.

4.3.3 Block Crop Adversarial (BCA) Test Results

After the BA test cases, AFV and VPG were also evaluated on the BCA test cases. The individual results of the completion rates for these three cases are portrayed in Figure 22. The average results are also provided. For these test cases with crop objects included, the VPG policy has the same or higher completion rates in comparison to AFV. While surprising, this demonstrates the strong generalization ability of VPG to objects—in this instance the crop objects—not encountered during training. Also, the difference in the averaged completion rate of the two policies on the BCA test cases is minimal, 3.3% as given by Table 7. These high completion rates of both the BA and BCA test cases demonstrate that both policies can clear cluttered, challenging arrangements of both block and crop objects.

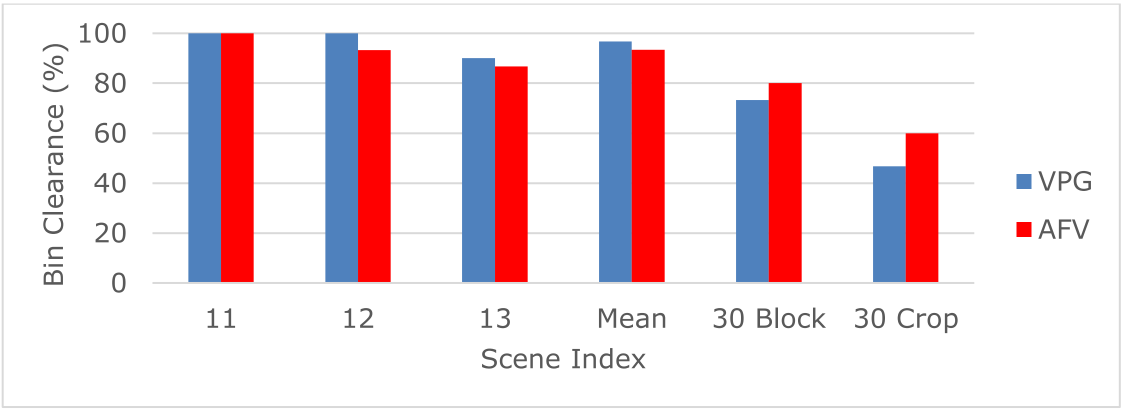


Figure 22: Completion rates of VPG and AFV for the BCA test cases, 30 block random arrangement test case, and 30 banana random arrangement test case. The scene images for each scene index on the x-axis can be found in Figure 10.

The plot in *Figure 21* is not recreated for the BCA test cases because pushing actions were only used in a single test case—VPG scene 13. The low frequency of pushing is illustrated in Table 7 with the averaged grasp to push ratio for the three BCA test cases. For AFV, not a single pushing action is used across the three tests, but a high grasp success rate and action efficiency are achieved. For VPG, the policy can achieve a high completion rate with almost exclusively grasping actions. For scene 13, VPG uses three pushing actions across 30 trials and still achieves a high clearance rate. Despite a higher completion rate than AFV, the VPG policy has a lower grasp success and action efficiency than AFV across the BCA cases.

4.3.4 Final Intense Clutter Evaluations

The final evaluations used to compare AFV and VPG were two scenes of 30 randomly placed block and crop objects repeated 30 times. The results are shown in Table 7. For the 30 clutter block scenario, the performance of VPG concerning grasp success and action efficiency is slightly higher than AFV, 1.7% and 0.5% respectively. In contrast, the completion rate of AFV is 7.7% higher than VPG. This minimal difference in grasp success and action efficiency combined with a higher completion rate shows that AFV is more capable of decluttering intense naturally occurring clutter better than the baseline VPG. Another result of interest is the low pushing frequency used to declutter the random arrangements. Both policies have a higher grasp to push ratio than they did for the BA test cases where the blocks were manually arranged.

After the 30 block test, a 30 banana test was carried out. In this test, the resulting completion rates of VPG and AFV were lower than any of the previous testing scenarios. Despite this, the grasp success rate and action efficiency of both policies for this 30 banana clutter surpass their respective grasping success and action efficiency seen on the BA test cases. For the 30 repeated banana clutter scenarios, AFV can clear all 30 bananas 60.0% of the time whereas VPG can only clear all 30 bananas 46.7% of the time. The AFV policy's stronger performance on the 30 banana scenario is further indicated by the grasp success rate and action efficiency, which both exceed the respective metrics of VPG by approximately 16%. In this final test scenario, pushing actions are never used by AFV and only minimally by VPG.

Table 7: Summary of averaged results of the AFV and VPG policies for all four testing scenarios.

Agent	**Completion (%)**	**GS (%)**	**AE(%)**	**GtP (%)**

11 Block Adversarial (BA) Test Cases				
VPG	90.9	52.7	52.0	95.2
AFV	89.4	50.2	51.4	98.9
3 Block Crop Adversarial (BCA) Test Cases				
VPG	96.7	76.4	76.4	99.9
AFV	93.3	86.9	87.3	100.0
30 Block Random Arrangement Test				
VPG	73.3	68.0	67.5	97.8
AFV	80.0	66.3	67.0	99.6
30 Banana Random Arrangement Test				
VPG	46.7	68.4	70.0	99.8
AFV	60.0	84.3	86.0	100.0

5 Discussion

In this section, a discussion between the results of this thesis will be related to current practices and studies that lie at this intersection of reinforcement learning, computer vision, and robotic manipulation. The results from the previous section will be compared to current state-of-the-art methods while emphasising the strengths and weaknesses of the new formulation of VPG as the Agro-Food Variant (AFV). The comparisons will be made with the original VPG report from Zeng et al., (2018A) and two new formulations of the clearance method through robotic manipulation and reinforcement learning (Ren et al., 2021; Tang et al., 2021). The work by Tang et al. (2021) extended VPG with the replacement of top-down grasping with 6-DoF grasping while removing any direct rewards for pushing actions.

In the work by Ren et al. (2021), a comparable method to VPG was developed with the addition of another motion primitive of shifting. The shifting action is like the pushing motion described in this work but contact from the gripper occurs at the top of the object as opposed to the side. Additionally, a masking function is introduced for the determination of rewards for the non-grasping actions. The function still depends on pre-set thresholds for heightmap image change detection like in Equation (15), but it enables the agent to learn more specific information on how pushing or shifting actions can better scatter objects for effective future grasps. In summary, the main components are the same as the work in this thesis besides the introduction of an additional motion primitive and mask function for the reward systems.

5.1 Pushing Reward

In the recent works mentioned above, none show the frequency of pushing actions. Without this frequency or knowledge of how many pushes are used it is difficult to assess the utility of secondary motion primitives. However, including pushing or secondary motion primitive frequencies allows for an assessment of pushing utility as shown in Figure 15 and Figure 21. From the results found in this project, a link can be drawn between high failure rates and excessive pushing.

Additionally, the recent methods prioritize grasp success rate as the main metric of interest, while minimizing the importance of task completion by only using fully completed tasks for the metric calculations. All failed tasks are disregarded in determining grasp success rate and action efficiency. By doing so, the negative impacts of pushing are largely ignored as once again seen in Figure 15 and Figure 21. In this thesis, all failed tasks are included in metric calculations so direct quantitative comparisons in regards to grasping success and action efficiency are difficult to make between the methods.

In the pushing experiments performed, it was found that coupling a reward to pushing increases the frequency of pushing actions in training and testing. When the reward was removed for the pushing actions, failure modes occurred at a lower rate evidenced by the outperformance of the PNR agent compared to the baseline VPG seen in Section 4.1.2. This is an interesting result that may indicate a limitation of the VPG method. In the original work by Zeng et al. (2018), the comparison of a system with a reward for pushing and a system without the reward stopped after the training results (Figure 23). Despite the quicker training of the rewarded system, the difference between the two grasp success rates is only 10% after 2500 iterations. If this was all the evidence used for determining that a coupled reward is better, then it appears incomplete without examining both systems on test cases.

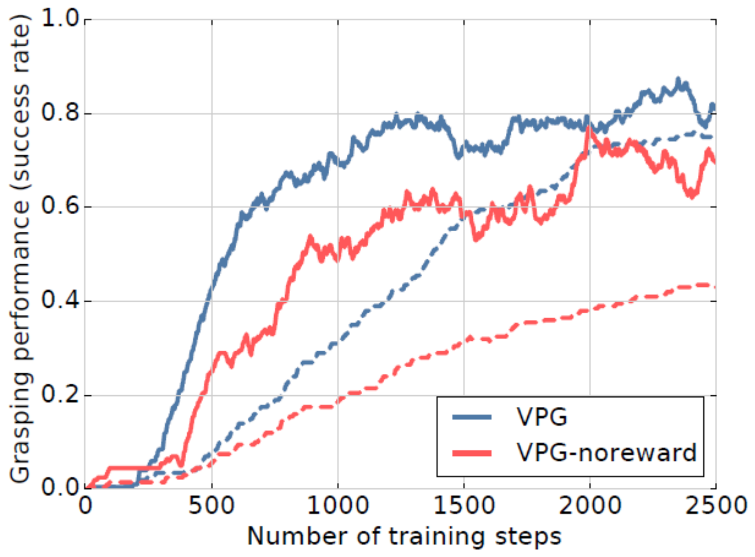


Figure 23: Comparing the performance of VPG policies trained with and without reward for pushing. Solid lines indicate % grasp success rates and dotted lines indicate % push-then-grasp success rates over training steps.

This incomplete evaluation is further evidenced by the work of Tang et al. (2021) where the reward for pushing was removed altogether like the well-performing PNR agent presented in this thesis. Their method was also evaluated on the same BA test cases as this thesis and Zeng et al. (2018A). The best performing pushing experiment agents—PNR and HCT—are compared to the results of Tang et al. (2021) and Zeng et al. (2018A) in Table 8. Both PNR and HCT completed more trials than the literature version of VPG (Zeng et al., 2018A). Their respective grasp success and action efficiencies cannot be easily compared to the literature results because failed test cases are considered in this work. However, since the primary metric of performance of this thesis is the completion rate the removal or revision of the reward system can directly improve task completion. The method presented by Tang et al. (2021) outperformed on all metrics, but the exact reason for this cannot be directly attributed to the removal of the reward for pushing. Since both 6-DoF grasping and no reward for pushing are included in this methodology, either could be the reason for the high performance on the BA cases.

*Table 8: Averaged results of the 11 Block Adversarial (BA) Test Cases of the pushing experiment best performers, Pushing No Reward (PNR) and Higher Change Threshold (HCT) from Table 4, compared with literature values. PNR and HCT values can also be found in Figure 15. *Calculations of the grasp success (GS) and action efficiency (AE) do not include failed test cases and may therefore be inflated.*

Agent Policy	**Iterations**	**Completion (%)**	**GS (%)**	**AE (%)**	**GtP (%)**
PNR	2500	93.0	47.0	47.4	99.0
HCT	2500	90.6	62.3	47.7	74.0
Zeng et al. (2018A)	2500	82.7	77.2*	60.1*	N/A
Tang et al. (2021)	2500	100.0	87.0	65.2	N/A

The strong performance of the HCT agent compared to literature VPG shows that a revision of the reward system for pushing can also lead to increased completion rates (Zeng et al., 2018A). In Ren et al. (2021), a more detailed revision of the pushing reward system is included using a masking function for clutter quantization and image change detection. While their method does expand on the original work by Zeng et al. (2018A), it is only compared with that method on grasp success rate on a real-world robot. Despite the limited comparison, an increase in grasp success of 26% on randomly cluttered blocks is a large improvement. This improvement shows that revision of the reward system for secondary motion primitives needs further investigations as arbitrarily set thresholds, e.g. τ in Equation (15), are not enough to train an agent to learn useful pushing actions. In other words, only doubling the threshold of image change detection, τ_{new}, for the HCT agent is insufficient for the revision of the reward system. It should be noted that since an additional secondary motion primitive of shifting is included along with pushing in this new method, the increase in grasp success rate cannot be attributed to only the revision of the reward system.

Table 9: Comparison of state-of-the-art grasping algorithms on grasping randomly cluttered block objects

Agent Policy	**Iterations**	**GS (%)**
Zeng et al. (2018A)	2500	68
Ren et al. (2021)	2500	94

To summarize, the results from the pushing experiments for the PNR and HCT agents are backed by recent extensions of the VPG methodology by Zeng et al. (2018A). The PNR agent and method used by Tang et al. (2021) both improve task completion compared to the original method. Another method for improving VPG was the revision—as opposed to removal—of the pushing reward as seen in this project by the HCT agent and in recent literature (Ren et al., 2021). These two modifications of the pushing policy present two opportunities for further agent and method development.

5.2 Training Objects

All three of the methods from the literature train on the same 8 different block shapes as in this thesis. No new objects were introduced in training, unlike the training regimen experiment. These other methods focused on the generalization of grasping and secondary motion primitive skills to new objects or scenarios not seen in training, while the objective of this thesis is to build the foundation for an agent that can use these skills in a specialized food processing use case. This preference for generalization ability over specificity in current state-of-the-art methods motivated the experiments performed in Section 4.2.

The previously mentioned methods all train on a limited amount of training objects, but none as limited as the Crop-only training regimen which trains on only two different crop object shapes. Examining Figure 23, the highest grasp success rate achieved by VPG is around 85% near the end of the 2500 iterations (Zeng et al., 2018A). When comparing this to Figure 16, the Crop-only trained agent reaches this grasp success rate 1000 iterations before the original VPG. Strong training performance of the specified Crop-only agent provides insight on how these types of robotic manipulation methods should be trained for specific use case scenarios. For example, since the end goal of the FlexCRAFT group is to apply a similar method to the bin picking of chicken pieces only, then only the types of chicken pieces that would appear in the bins should be used for training the agent. This will limit the generalization ability of the agent, but since the training can be done much quicker when the number of training objects is reduced retraining another agent for different chicken pieces or other specific processing tasks is not time-consuming or computationally exhaustive.

With the high performance of the Crop-only agent in training comes a low performance in generalization to the BA test cases. This specific training regimen limits the objects the Crop-only model is capable of handling indicating a low-level ability to generalize to novel objects and scenarios. Overall, this does not mean the Crop-only training regimen is bad but illustrates the existence of a generalization-specialization trade-off. To improve generalization, the results of the training regimen experiment would suggest expanding the number of training objects used to train the agent. To do so, it is likely that more training iterations are necessary for the agent to learn the manipulation strategies required for the expanded set of training objects. Inversely to reduce training times required by the agent, the results of the experiment suggest reducing the set of training objects to as few as possible. This should only be done when the number of different object shapes acted upon in practice or real-world is small.

5.3 Agro-Food Context

A large performance improvement was made to the original visual pushing for grasping method when incorporating the three modifications in the AFV agent when comparing the two methods. This was shown in Figure 22. By removing the intrinsic reward for pushing, increasing the manually set threshold for pushing success, and training on an expanded set of training objects enabled the AFV agent to outperform the original VPG on the agro-food clutter scenario seen in Figure 11(d).

Despite this clear performance improvement, the results found in this comparison are limited and need verification and validation by applying the methods on a real-world robot. Because the results are based on simulated trials, product variations common in agro-food were not able to be considered since the simulated agro-food objects are identical.

6 Conclusions

In this section, the conclusion to the thesis project is provided with answers to the research questions. The answers to the sub-questions will be provided first because the information obtained from answering them formed the evidence that led directly to finding and answering the main research question.

6.1 Answers to Sub-Research Questions

How does the reward system for pushing affect performance?

Coupling a reward system to pushing will always lead to an increase in the frequency pushing actions used by the robotic agent. This increased frequency of pushing can lead to undesired effects seen by the lower completion rates when a reward is given for pushing actions deemed successful. The lower completion rate occurs due to excessive pushing potentially caused by the agent overfitting to the pushing reward seen in Table 4, which is easier to achieve than the grasping reward. In other words, the criterion for a successful push is too simple and does not train effective pushes to enable grasping. This directly leads to pushing actions being overused, which creates failure modes by objects being removed from the workspace scene by pushing actions as seen in Figure 15.

How do the training objects affect final performance?

When a large amount of training object shapes are used, the performance in completion rate across all test cases also improves Table 5 and Table 6. When a limited number of training objects are used, training time can be significantly improved as seen in the Crop-only policy training illustrated in Figure 16. However, this creates a specialized agent that cannot perform well on test cases that have objects never before seen in training. This results in low performance for all evaluation metrics seen in Table 5.

6.2 Answer to Main Research Question

What are effective strategies for improving VPG (visual pushing for grasping) performance when applying the method to a cluttered agro-food scenario?

Effective strategies for improving visual pushing for grasping on cluttered agro-food scenarios are the removal of a direct reward for pushing actions, doubling the threshold required for determining successful pushes, and training on an expanded set of training objects that include objects that will be seen in the agro-food testing scenario. These individual strategies showed improvements in completion rates when individually compared to the baseline strategies of VPG. In Table 4, the removal of pushing reward (PNR) and increased threshold for successful pushes (HCT) are shown to improve performance in completion rate which is the highest priority metric for agro-food clutter clearance. In Table 5 and Table 6, the improved completion rate of the expanded training set (Block+Crop) on the different evaluations illustrates improvements compared to the baseline VPG training. When all three of these strategies were combined for a single modified agent of VPG (referred to as AFV), the performance metrics of this new agent compared to the baseline-unmodified VPG improved by 13.3% in completion rate, 15.9% in grasp success rate, and 16.0% action efficiency when applied to an agro-food cluttered scenario as seen in Table 7.

7 Recommendations

Current methods that lie at the same intersection of computer vision, robotic manipulation, and reinforcement learning as discussed in this thesis are focused on improving generalization and expanding the abilities of robotic agents to mimic human manipulation. While this is promising for robotics and automation, the level of specialization these methods seem capable of suggests they should be applied to more specific use cases like poultry processing. To assess if it can work in these use cases, experimentation on a real-world robot is necessary because product variation could not be included in simulation for the manipulation objects.

If this method were to be continued, major reworks of the existing code are necessary. Not only does the code need to be updated for newer robots, but the current code also does not work with the firmware of current cobots (UR5 more specifically). Additionally, the existing code for the method lacks modularity so extensions like a masking function, new motion primitives (Ren et al.), or 6-DoF grasping (Tang et al.) will not be possible till the method is reworked or the other methods become open source. With the current limits of top-down grasping, the method is not recommended for soft body products like many types of chicken meat. Only chicken pieces with rigid structures in them would be acceptable manipulation objects for the robotic agent.

References

Bauza, M., & Rodriguez, A. (2017). "A probabilistic data-driven model for planar pushing". *IEEE International Conference on Robotics and Automation*, 3008-3015.

Beuchat, L. R., & Ryu, J. H. (1997). Produce handling and processing practices. *Emerging Infectous Diseases*, 459-465.

Bohg, J., Morales, A., Asfour, T., & Kragic, D. (2013). "Data-driven grasp synthesis--a survey". *IEEE Transactions on Robotics*, vol. 30. no. 2. 289-309.

Boularias, A., Bagnell, J. A., & Stentz, A. (2015). *Learning to manipulate unknown objects in clutter by reinforcement.* AAAJ.

Bryavan, A., & Fox, D. (2017). "SE3-nets: Learning rigid body motion using deep neural networks". *IEEE International Conference on Robotics and Automation*, 173-180.

Buckenhüskes, H. J., & Oppenhäuser, G. (2014). DLF-Trend Report: Robots in the Food and Beverage Industry. In *DLG Lebensmittel, 9(6)* (pp. 16-17).

Calderone, L. (2013). *Food Processing Without the Human Touch.* Opgehaald van Robotics Tomorrow: http://roboticstomorrow.com/content.php?post_type=1851

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). *Imagenet: A large-scale hierarchical image database.* CVPR.

Depierre, A., Dellandréa,, E., & Chen, L. (2018). "Jacquard: A large scale dataset for robotic grasp detection". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3511-3516.

Diankov, R. (2010). *Automated construction of robotic manipulation programs.* CMU RI.

Dogar, M. R., & Srinivasa, S. S. (2012). *A planning framework for non-prehensile manipulation under clutter and uncertainty.* Autonomous Robots.

Fang, H. S., Wang, C., Gou, M., & Lu, C. (2020). "Graspnet-1billion: a largescale benchmark for general object grasping". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 11,444-11,453.

Finn, C., & Levine, S. (2017). "Deep visual foresight for planning robot motion". *IEEE International Conference on Robotics and Automation*, 2786-2793.

Ghadirzadeh, A., Maki, A., Kragic, D., & Bjorkman, M. (2017). "Deep predictive policy training using reinforcement learning". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2351-2358.

Girshick, R. (2015). Fast R-CNN. *IEEE International Conference on Computer Vision*, pp. 1440-1448.

Goldfeder, C., Ciocarlie, M., Dang, H., & Allen, P. K. (2009). *The colombia grasp database.* ICRA.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Granta. (2021). *What is the difference between automation and robotics.* Opgehaald van Granta Automation: https://www.granta-automation.co.uk

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). "Mask R-CNN" . *IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.

Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). *Densely connected convolutional networks.* CVPR.

Ignasi, C., Held, D., & Abbeel, P. (2017). "Policy transfer via modularity". *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Joffe, B., Walker, T., Gourdon, R., & Ahlin, K. (2019). *Pose estimation and bin picking deformable products.* Atlanta, GA, USA: Georgia Tech Research Institute.

Joffe, S., & Szegedy, C. (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 448-456.

Karpathy, A. (sd). *CS231n Convolutional Neural Networks for Visual Recognition*. Opgehaald van Github.

Kumra, S., Joshi, S., & Sahin, E. (2020). "Antipodal robotic grasping using generative residual convolutional neural network". *iEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Lau, M., Mitani, J., & Igarashi, T. (2011). Automatic learning of pushing strategy for delivery of irregular-shaped objects. *IEEE International Conference on Robotics and Automation*, 3733-3738.

Li, Y., Wang, G., Ji, X., Xiang, Y., & Fox, D. (2018). "Deepim: Deep iterative matching for 6d pose estimation". *Proceedings of the European Conference on Computer Vision (ECCV)*, 683-698.

Long, J., Shelhamer, E., & Darrell, T. (2015). *Fully convolutional networks for semantic segmentation.* CVPR.

Luijkx, J. (2020). *Robotic Grasping of Deformable Food Objects.* TU Delft.

Lynch, K. M., & Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control.* New York, NY, USA: Cambridge University Press.

Lynch, K., & Mason, M. (1999). "Dynamic nonprehensile manipulation: Controllability, planning, and experiments". *International Journal of Robotics Research*, no. 1, pp. 64–92.

Maricli, T., Veloso, M., & Akm, H. L. (2015). *Push-manipulation of complex passive mobile objects using experimentally acquired motion models.* Autonomous Robots.

Mataric, M. J. (1994). "Reward functions for accelerated learning. *Machine Learning Proceedings*, 181-189.

Meshram, B. D., Shaikh, A., & Suvartan, R. (2018). Robotics: An Emergin Technology in Dairy and Food Industry: Review. *International Journal of Chemical Studies, 6(2)*, 440-449.

Minh, V., Kavukeuoglu, K., Silver, D., Rasu, A. A., & et al. (2015). *Human-level control through deep reinforcement learning.* Nature.

Miyazawa, K., Maeda, Y., & Arai, T. (2005). "Planning of graspless manipulation on rapidly-exploring random trees". *The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing*, 7-12.

Nair, V., & Hinton, G. E. (2010). "Rectified linear units improve restricted Boltzmann machines". *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 807-814.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Big Data*.

NFU. (2020, 11 17). COVID-19: Travel quarantine exemption granted for seasonal poultry workers. United Kingdom.

Omrcen, D., Böge, C., Asfour, T., Ude, A., & Dillmann, R. (2009). Autonomous acquisition of pushing actions to support object grasping with a humanoid robot. *Humanoids*.

P6: Food-Processing Robotics. (sd). Opgehaald van FlexCRAFT: https://flexcraftprogram.com/2018/12/17/project-6food-processing-robotics/

Paszke, A., & et al. (2016). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Opgehaald van Software available from pytorch.org

Prasad, S. (2017). *Application of Robotics in Dairy and Food Industries: A Review.*

Purnell, G., & Further , G. I. (2013). Robotics and automation in meat processing. *Robotics and Automation in the Food Industry*, pp. 304-328.

Ranger, P., Ottley, G., & Smith, N. (2004). The Pitfalls of Purchasing. *Food Processing*.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). "You Only Look Once: Unified Real-Time Object Detection". *IEEE Conference on Computer Vision and Pattern Recognitiion (CVPR)*, 779-788.

Ren, D., Ren, X., Wang, X., Tejaswi Digumarti, S., & Shi, G. (2021). *Fast-Learning Grasping and Pre-Grasping via Clutter Quantization and Q-map Masking.* arXiv.

Ren, G., Lin, T., Ying, Y., Chowdhary, G., & Ting, K. C. (2020). *Agricultural robotics research applicable to poultry production: A review.* Computers and Electronics in Agriculture, 169.

RG2 Gripper Datasheet. (2015). On Robot Ap5.

Rohmer, E., Singh, S. P., & Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework. *IROS*.

Rohmer, M., & Singh, S. (2013). *V-rep: a versatile and scalable robot simulation framework.* IROS.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, vol. 323*, 533.

Sain, M., Sing, R., & Kaur, A. (2020). Robotic Automation in Dairy and Meat Processing Sector for Hygienic Processing and Enhanced Production. *Journal of Community Mobilization and Sustainable Development*, 543-550.

Salganicoff, M., Metta, G., Oddera, A., & Sandini, G. (1993). *A vision-based learning method for pushing manipulation.* University of Pennsylvania.

Shaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). *Prioritized experience replay.* ICLR.

Shi, S., Wang, X., & Li, H. (2019). "Pointrcnn: 3d object proposal generation and detection from point cloud". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-779.

Striffler, S. (2007). *Chicken: The Dangerous Transformation of America's Favorite Food.* Yale University Press.

Sutton, R., & Barto, A. (2014). *Reinforcement Learning: An Introduction.* The MIT Press.

Tang, B., Corsaro, M., Konidaris, G., Nikolaidis, S., & Tellex, S. (2021). *Learning Collaborative Pushing and Grasping in Dense Clutter.*

Timmers, R. (2018). *Learning to Grasp Objects with Reinforcement Learning.* University of Groningen.

Universal Robots. (sd). *Universal Robots e-Series User Manual.* Opgehaald van https://www.universal-robots.com/support/.

Walker, S., & Salisbury, J. K. (2008). "Pushing using learned manipulation maps". *IEEE International Conference on Robotics and Automation*, 3808-3813.

Wang, S. C. (2003). Artificial Neural Network. In *Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743.* Boston, MA: Springer.

Watkins, C. (1989). *Learning from delayed rewards.* PhD Thesis, University of Cambridge, England.

Watters, N., Zoran, D., Weber, T., Battaglia, P., Pascanu, R., & Tacchetti, A. (2017). "Visual Interaction networks: Learning a physics simulator from video". *Advances in Neural Information Processing Systems*, vol. 30.

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2018A). *Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning.* IEEE.

Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2020, February 4). *Visual Pushing and Grasping Toolbox*. Opgehaald van GitHub: https://github.com/andyzeng/visual-pushing-grasping

Zeng, A., Song, S., Yu, K.-T., Donlon, E., Hogan, F. R., & Bauza, M. (2018B). *Robotic pick-and-place of novel objects in cluttter with multi-affordance grasping and cross-domain image matching.* ICRA.

Zeng, A., Yu, K.-T., Song, S., Suo, D., Walker, E., Rodriguez, A., & Xiao, J. (2017). *Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge.* ICRA.