

A Machine Learning Approach to Grasp Planning for Forestry Log-loading

Elie Ayoub

Department of Mechanical Engineering
McGill University
Montréal, Québec, Canada

August 15, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Science

©2023 Elie Ayoub

Abstract

Automation in the forestry sector is largely dependent on the possibility of automating log-loading processes. Although currently proposed techniques have shown potential, they target individual logs rather than piles which neglects possible interactions between logs and extends the amount of time and energy required to complete log-loading tasks. In this context, the main objective of this research is to develop a vision-based pipeline to detect, cluster and localize piles of logs, and most importantly, predict the most efficient grasping location for the log-loader to pick up the logs. Nonetheless, successful log grasping depends on the state of the log-loader's grapple; therefore, a compound multi-purpose neural network is also developed to detect the grapple and determine its opening width. The network is further augmented to count the number of logs that the grapple is carrying after logs were picked up. This thesis presents the developed machine learning techniques for grasp planning and grapple detection with its derivatives, in addition to simulated and experimental results for the vision-based grasp planning pipeline.

Abrégé

L'automatisation dans le secteur forestier dépend largement de la possibilité d'automatiser les processus de chargement des grumes. Bien que les techniques actuellement proposées aient montré leur potentiel, elles ciblent les grumes individuelles plutôt que les tas, ce qui néglige les interactions possibles entre les grumes et prolonge la durée et l'énergie nécessaires pour effectuer les tâches de chargement des grumes. Dans ce contexte, l'objectif principal de cette recherche est de développer un pipeline basé sur la vision pour détecter, regrouper et localiser les tas de grumes, et surtout, prédire l'emplacement le plus efficace pour que la grue de chargement de grumes puisse les saisir. Néanmoins, la réussite de la saisie des grumes dépend de l'état de la pince de la grue de chargement de grumes ; par conséquent, un réseau de neurones composé multi-usages est également développé pour détecter la pince et déterminer sa largeur d'ouverture. Le réseau est en outre augmenté pour compter le nombre de grumes transportées par la pince après que les grumes ont été saisies. Cette thèse présente les techniques d'apprentissage automatique développées pour la planification de la saisie et la détection de la pince, ainsi que leurs dérivés, en plus des résultats simulés et expérimentaux pour le pipeline de planification de la saisie basé sur la vision.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor Prof. Inna Sharf for her constant guidance, insights and invaluable discussions during my time at McGill.

I would also like to thank James Anawati, Hunter Song, Jad Wehbeh, Jackson Empey, Anthony Maalouly, Juan Carlos Hernandez Ramirez and all my colleagues in the Aerospace Mechatronics Laboratory for all the professional and personal discussions, and for making spending almost everyday for the past two years in a lab an enjoyable experience. I am furthermore grateful to the undergraduate students that contributed to my work, namely George Sideris, Oscar Cardenas Gomez (Data labeling and processing), and Junrui Huang (CAD modelling) .

Moreover, I am thankful to FPInnovations for allowing me to use their facilities for this work's experimentation and for the opportunity to intern and gain valuable work experience at their organization through the MITACS Accelerate Program.

I am grateful for the trust that the graduate students of the mechanical engineering department put in me by giving me the opportunity to serve as the president of the Graduate Association of Mechanical Engineering Students, and for all the executive members on the GAMES executive board who went above and beyond when it came to management, event planning, and representing graduate students in Mech. Eng..

Finally, I would like to express my appreciation to my father and my mother for all the sacrifices that they made for me to be where I am today and their constant support, and my brother for always looking up to me and pushing me to be the best role model that I can be.

The research of this thesis was made possible by the financial support of the NSERC Canadian Robotics Network (NCRN).

This work was supported by Mitacs through the Mitacs Accelerate program.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Objective	2
1.3	Literature Review	4
1.3.1	Grasp Planning with Deep Neural Networks	4
1.3.2	Manipulation with Reinforcement Learning	6
1.3.3	Generative Adversarial Networks for Log-loading	8
1.3.4	Log-loading Related Non-learning Methods	9
1.4	Thesis Outline	11
2	CNN for Grasp Planning	12
2.1	Grasp Definition	13
2.2	Convolutional Neural Network	16
2.2.1	Network Architecture	16

2.2.2	Network Training	18
2.2.3	Evaluation metrics	21
2.3	Performance and Comparison	22
2.3.1	Training and Evaluation	22
2.3.2	Simulation Grasping	25
2.3.3	Experimental Grasping	28
3	Grasp Planning Pipeline	31
3.1	Log Segmentation	32
3.2	Log Piles Clustering	33
3.3	Virtual Environment	35
3.4	Convolutional Neural Network	38
4	Simulation and Experimental Validation	42
4.1	Log-loading Test-bed	43
4.1.1	Crane	43
4.1.2	Hardware and Software Architecture	45
4.1.3	Testing Environment	47
4.2	Simulation Environment	48
4.3	Experimental Procedure	50
4.4	Results	52

4.4.1	Simulated Results	52
4.4.2	Experimental Results	54
4.5	Virtual Camera Placement and Grasp Planning	56
4.5.1	Logs at the Edges of Input Images	56
4.5.2	Non Intuitive Grasps on Individual Logs	60
5	Log-Counting and Grapple Opening	65
5.1	U-Net	66
5.2	Grapple Opening and Log Counting Architecture	68
5.3	Training and Evaluation	71
5.3.1	Simulation Environment and Model	71
5.3.2	Training Datasets	73
5.3.3	Network Training	75
5.4	Experimental Validation and Results	80
5.4.1	Simulated Results	80
5.4.2	Experimental Results	82
6	Conclusion	86
6.1	Research Summary	86
6.2	Future Work	89

List of Figures

2.1	World space grasp	14
2.2	Image space grasp	14
2.3	Grasp map with target image space grasp	15
2.4	Network architecture summary	17
2.5	Cornell grasping dataset sample	19
2.6	IoU metric	22
2.7	Orientation difference metric	22
2.8	Networks training loss	24
2.9	Networks validation loss	24
2.10	Networks validation accuracy	25
2.11	Gazebo simulation setup	26
2.12	Gazebo objects set	26
2.13	Kinova Jaco2 robotic arm	28
2.14	Household objects set	28
3.1	Grasp planning pipeline schematic	32

3.2	Logs segmentation sample, FPInnovations log-loading test-bed	33
3.3	Clustering sample output	35
3.4	Replication of sample chosen cluster in Gazebo virtual environment	37
3.5	World space grasp on log	38
3.6	Image space grasp on log	38
3.7	Grasp map from log image and predicted optimal grasp	39
3.8	Training and validation loss	41
3.9	Validation accuracy	41
4.1	FPInnovations log-loading test-bed	44
4.2	Hardware architecture	46
4.3	Software architecture	47
4.4	Log-loading test-bed environment	47
4.5	Logs sample	48
4.6	Log pile sample	48
4.7	Log-loading test-bed Isaac Sim simulation environment	49
4.8	Input depth image from 3 meters above logs	57
4.9	Quality Distribution for camera placement of 3 meters above logs	57
4.10	Input depth image from 4 meters above logs	57
4.11	Quality Distribution for camera placement of 4 meters above logs	57
4.12	Inner bounding box	59

4.13 Canny algorithm edge detection	59
4.14 Failed, Non Intuitive, and Optimal grasp planning samples	61
5.1 Original U-Net architecture	67
5.2 Multi-branch network block diagram	68
5.3 Our U-Net architecture	69
5.4 Log-counting network branch architecture	70
5.5 Grapple opening network branch architecture	71
5.6 ZED 2i grapple camera view	72
5.7 Isaac Sim grapple camera view	72
5.8 Dataset 1 sample	74
5.9 Dataset 2 sample	74
5.10 Isaac Sim grapple images dataset samples	74
5.11 Real grapple U-Net training and validation losses	76
5.12 Simulated grapple U-Net training and validation losses	76
5.13 Real grapple segmentation sample	77
5.14 Simulated grapple segmentation sample	77
5.15 Log-counting branch training and validation losses	78
5.16 Log-counting branch training and validation accuracies	78
5.17 Grapple opening training and validation losses	80
5.18 Grapple with three grasped logs, one of which is hidden	84

List of Tables

2.1	Network architecture layers summary	18
2.2	Our network vs. GG-CNNs 10-Fold cross validation results	23
2.3	Chosen final network models	25
2.4	Gazebo simulation grasping attempts results	27
2.5	Experiment 1 grasping attempts results	29
2.6	Experiment 2 clutter grasping results	30
4.1	Denavit-Hartenberg parameters table	44
4.2	Log-loader joint ranges table	45
4.3	Log configurations for simulated and experimental trials	50
4.4	Simulated grasps outcomes	53
4.5	Experimental grasps outcomes	55
4.6	Log characteristics	61
4.7	Log 1 orientation test results	62

4.8	Log 2 orientation test results	62
4.9	Log 3 orientation test results	62
4.10	Log 4 orientation test results	62
4.11	Grasp planning results based on camera orientation	64
5.1	Grapple opening and log-counting simulated results	81
5.2	Simulated log inventory counting results	82
5.3	Sim-to-real grapple opening and log-counting experimental results	83
5.4	Experimental log inventory counting results	85

Chapter 1

Introduction

Automation in the forestry industry has been receiving increased attention over the past few years, as witnessed by the number of recent works that address robotics and AI related topics in the context of forestry operations. While these operations are numerous, the process of collecting felled trees, termed "log-loading", presents itself as a key task and one of the challenging forestry operations to automate. This chapter provides the necessary context and background to introduce the log-loading automation work of this thesis and delineates the objectives that will be pursued throughout its chapters.

Some of the contents of this chapter are extracted from [1].

1.1 Motivation

The lack of skilled machine operators, with many in the labour force close to retirement, coupled with low student participation in educational and training programs for forestry machine operators, make the shift from human-operated to automated forestry machines a pressing necessity [2, 3]. Efforts have been made to assist operators and to enhance human-machine cooperation through automating some decisions for the operator [4] or enabling teleoperation [5], but few works aim to achieve complete autonomy for the tasks of tree felling and log loading operations. For the log loading problem specifically, which is one of the key operations in delivering timber from forest to mill, majority of existing literature tackles the problem of path planning for the log-loader's crane [6,7]. While few works attempt autonomous log grasping [7–9], the results have always been demonstrated for single, isolated logs, rather than a diversity of possible configurations ranging from cluttered logs to large log piles. The recent heightened interest in automation for forestry and the absence of a robust grasp planning framework to produce grasps that allow single or multiple log grasping in complex log configurations motivates the work of this thesis.

1.2 Objective

Given the aforementioned context, the objective of this work at large is to decrease operator-dependency in the log-loading process through the automation of the grasp planning process

that is typically performed by the operator, while maintaining key capabilities like targeting complex log arrangements and log piles.

The thesis' detailed objectives can be broken down as the following:

- Develop and evaluate a convolutional neural network that leverages computer vision to produce optimal grasps and that can be adapted to forestry log-loading scenarios.
- Integrate the developed CNN into a full grasp planning pipeline for log-loading, starting from identifying target logs in a log-loader's environment based on RGB-D image data to predicting exact grasping coordinates to grasp detected logs.
- Validate the full grasp planning pipeline on a realistic log-loader simulation and through experimentation on a real log-loading test-bed.
- Enhance the degree of automation in log-loading through the development of a compound CNN-based architecture that uses the same RGB-D input data from the grasp planning pipeline to segment a log-loader's grapple and its grasped logs, and predict the grapple's opening and the number of logs that it has collected.

1.3 Literature Review

1.3.1 Grasp Planning with Deep Neural Networks

The inherent difficulties in robotic grasping have pushed recent research efforts in the direction of deep learning methods. Convolutional Neural Networks (CNNs), in particular, are a common choice for grasping solutions based on visual input [10, 11] due to their feature extraction abilities. Several state-of-the-art CNN multipurpose grasping approaches have been successfully used for accurate object picking and placing tasks [12–18]. For example, the approach from [12] relies on identifying good grasping candidates and ranking them using neural networks: An RGB-D image is divided into multiple grasping rectangles that are exhaustively tested by a first neural network to find potential good grasping candidates. The candidates are furthermore fed to a second larger and more complex neural network to determine the optimal grasping rectangle. This type of methods typically suffers from long inference times due to the high computational complexity that is required to sample and rank grasping candidates, and therefore, they are normally limited to open-loop grasping (10 second inference time in [12]).

A different kind of deep learning approach is that of [19] which creates two lightweight and fully convolutional neural networks termed GG-CNN and GG-CNN2 that rely on depth images only to generate dense pixel-level grasp distributions that assign a quality, grasping angle and width for each pixel in the input image rather than ranking sampled grasps. These

networks' lightweight architecture allows for grasp predictions in real time, thus enabling grasping that follows targeted objects.

When it comes to deep neural networks' use in forestry, the approach in [20] showcases the use of CNN from [13] to perform object segmentation and generate single-log grasp predictions. A backbone made of a residual neural network and a feature pyramid network is used to extract features from RGB visual data that are passed to two separate branches for grasp prediction and segmentation: the first branch contains a CNN based on the work in [13] to predict grasping candidates in the scene with a grasping confidence score that indicates chances of grasping success. As for the segmentation branch, it contains a segmentation backbone followed by a semantic and an instance segmentation blocks. The semantic segmentation block separates the graspable objects from the background of the input image, while the instance segmentation block utilizes the grasp prediction information from the grasp prediction branch in addition to depth information from the input image to detect and separate the different objects in the scene. The output of both segmentation blocks is combined to obtain an image that separates individual graspable items. The object with the highest success chance is then chosen as the grasping target. Although this approach targets individual logs, it proved to be successful at distinguishing wood logs in the scene and reached a single-log grasping success rate of 95%.

Another deep learning approach applied to forestry is that of [8]. RGB-D images are collected from the scene. The RGB data is fed to a CNN that predicts good grasping

candidates on individual logs in 2D space. The network relies on the architecture from [21] where a ResNet-50 feature detector [22] is used to extract descriptive features from the RGB data that are then fed to a network that performs 2D grasp prediction. The grasp prediction network provides grasping candidates first and further refines them by adjusting their parameters (grasping center, width, height, angle). The depth information from the input RGB-D data is then used to convert the 2D grasp candidates into the 3D world. This approach was tested in simulation and on a down-scaled log-loading crane with respective grasping success rates of 88% and 92%.

1.3.2 Manipulation with Reinforcement Learning

Reinforcement Learning (RL) methods have been intensely explored for general manipulation tasks involving robotic arms. These techniques rely on modelling cumulative reward functions and allowing the robot to take multiple actions that train a network to maximize that reward. RL approaches for grasping typically rely on visual data as input for the agents and perform end to end training on their networks. End to end learning refers to mapping current states of the robot's environment to joint motor commands directly, which allows RL techniques to bypass the need for controllers, motion planning, and object dynamics' knowledge [23]. To list a few examples, the work in [24] showcases a manipulator (agent) that receives RGB-D images from wrist and overhead cameras, in addition to the robot's joint positions to train a Q-learning network that outputs motor

commands and a grasp success probabilities. It was tested through grasp attempts on simple shapes (cube, sphere, cylinder) and achieved grasping success rates of 91%, 83%, and 89% respectively. Another approach in [23] shows the possibility of using RGB visual data from an over the shoulder camera only to generate gripper motion commands, with a grasp success rate of 96%. Moreover, this proposed technique allows for pre-grasp manipulations to reposition target objects to more graspable states. Even though these RL approaches have a great potential of grasping success, they require complex modelling of reward functions and long training times with potentially hundreds of thousands of grasp attempts (the method in [23] required several weeks of training on 7 robots, with 580 thousand attempts). Furthermore, although RL methods have been heavily applied to general manipulation tasks, few works attempt to implement such techniques on large scale hydraulic machines like excavator arms [25–28] and forestry cranes [7, 29]. The approach from [7] looks into learning successful actuator-space control policies to grasp single wood logs using a scaled-down redundant forestry crane. The agent receives information about the pose (position and orientation) of the target log and the end to end learned policy directly outputs the motor commands. This approach leverages a form of transfer learning called curriculum learning, where the agent learns several simple sub-tasks one by one. The learned knowledge from these “lessons” that form the curriculum are then transferred to the policy that governs the full complex task. Successful grasps, for the sake of reward modeling, are judged based on whether the target log was lifted off the ground and whether

the grapple claws are tightly closed around it. This technique was implemented in simulation and resulted in an accuracy of 97% when the reward function did not include elements to minimize energy consumption, and 93% when it did. As for the approach in [29], two supervised neural networks are trained to map between hydraulic cylinders' displacement and angular joint positions, and solve the forward kinematics of the crane. A reinforcement learning agent is then used to learn, in simulation, the inverse kinematics of the crane and a model-independent task-space control policy that provides joint angles at every instant based on the target end-effector location. A sim-to-real transfer of the learned policy is then performed to deploy the policy on a down-scaled log-loading crane model with an arm length of 2.5 m. The learned policy was tested on a trajectory tracking task in simulation and on the real crane and resulted in tracking errors that were up to 3 cm in simulation and 8 cm during the sim-to-real approach for each of the x,y, and z position components.

1.3.3 Generative Adversarial Networks for Log-loading

A different kind of learning approach to log grasping is that of [30], where a Generative Adversarial network (GAN) is used to output graspability heatmaps that predict whether logs are graspable or not based on an input RGB or RGB-D image. Images of clusters of wood logs are generated in simulation and a simple algorithm that detects intersection between logs is used to create the ground truth graspability heatmaps indicating for each

log whether it is graspable or not. A log is deemed graspable if it is higher than all the logs that it has points of intersection with. The generated images and labels are then used to train the generator and the discriminator of the GAN. The generator’s job is to take the image input and generate graspability heatmaps that are not distinguishable from the ground truth labels, while the discriminator’s job is to accurately detect whether these heatmaps are generated or are real ground truth heatmaps. Both networks are trained in parallel; the generator gets better at creating heatmaps that are indistinguishable from the ground truth data and the discriminator gets better at determining which images are synthetic. This approach achieved an accuracy of 100% on the training data and 97% on the evaluation data in a simple simulation environment that is not representative of a real forest. Furthermore, with this kind of approach, the generator and discriminator networks tend to not reach a state of convergence, which makes it tricky to find an adequate point to stop the training.

1.3.4 Log-loading Related Non-learning Methods

Several works attempt to tackle other log-grasping related issues, like detecting the presence of logs in the grapple. The work in [31] presents a grapple design with cheap, easy to manufacture, and accurate integrated proximity sensors that are capable of detecting the distance to logs around the grapple and the presence of logs between its tongs based on change in capacitance. This design was implemented in [9] on a down-scaled crane model with the CNN from [21] that predicts good grasp candidates based on RGB-D images. The

purpose of this implementation was to augment the log-loading crane with a sense about the quality of the grasps that it performs.

Another problem that has been addressed is the sway of the crane's grapple due to its passive joints. The approach from [32, 33] tackles minimizing grapple oscillations during slewing motions that rotate the entire crane. High order polynomials were used to generate smooth crane trajectories that result in continuous acceleration profiles and the optimal trajectory was selected experimentally to minimize the transient effects of the slewing torque at the endpoint of the crane's trajectory.

Although most of the mentioned approaches aim for automation of log-loading machines, some have explored other modalities, like teleoperation. The solution in [5] uses structural light cameras for real-time log detection and localization in a virtual environment. The operator can then teleoperate the log-loading crane through feedback from said virtual environment and long distance remote control to pick up the logs.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 introduces a grasp planning CNN, evaluates its performance and establishes a comparison between its performance and that of the networks that it was based on.
- Chapter 3 presents the grasp planning pipeline that integrates the CNN from Chapter 2 to perform grasp planning on logs, along with its components.
- Chapter 4 presents the simulation and the log-loading test-bed that were used for experimentation, along with the simulated and experimental results of the performed trials to validate the grasp planning pipeline of Chapter 3.
- Chapter 5 presents a compound network architecture that detects the log-loader's grapple and the logs that it is carrying through segmentation, and provides an estimate of the grapple's opening along with the number of logs that it is carrying.
- Chapter 6 concludes the thesis with a summary of the developed methods and their notable results, and presents possible future directions to further expand this work.

Chapter 2

CNN for Grasp Planning

In this chapter, we present a convolutional neural network that performs general grasp planning and we compare its performance with the GG-CNN networks from [19] using simulated and experimental grasping trials. Section 2.1 defines a grasp along with its parameters in the world and image space. It also introduces grasp maps, optimal grasps and conversion of grasps from image space to world space. Section 2.2 presents the developed CNN, its architecture, its training process and the evaluation metrics that were used to measure its performance. The CNN's performance is evaluated and compared with those of the GG-CNNs through theoretical evaluation metrics, simulated grasps on a Gazebo virtual objects set and experimental grasps on a household objects set in Section 2.3.

2.1 Grasp Definition

To train a convolutional neural network to perform grasp planning, a grasp must first be expressed as parameters that describe it. Therefore, we define a grasp, similarly to [19], as:

$$g = (q, p, \theta, w) \quad (2.1)$$

In (2.1), grasp quality q is a scalar in the range $[0,1]$ that represents the chances of performing a successful grasp, where a quality of 1 indicates the highest likelihood of grasping success. Due to the complexity of analytically assessing the chances of grasping success at different locations on objects that vary in shape, human-labeled datasets or simulation trials are generally used to provide ground-truth information on locations with high expected grasping quality on a given set of objects. The grasping location $p = (x, y, z)$ represents the 3D location with Cartesian coordinates (x, y, z) in world space where the grasp is to be performed. The grasping angle θ represents the angle that is required of a parallel gripper to perform the grasp, and the grasping width w represents the required opening width of said gripper. Figure 2.1 shows a graphical representation of a grasp in world space.

Since the work presented in this thesis relies on computer vision and depth imagery for

grasp planning, a grasp is also defined in the 2D image space to be:

$$\tilde{g} = (q, \tilde{p}, \tilde{\theta}, \tilde{w}) \quad (2.2)$$

where quantities associated with the image space are denoted as $(\tilde{\cdot})$. The grasp quality's symbol remains q in both the world and image space since it is invariant between the two. The grasping location in image space \tilde{p} indicates a pixel and is expressed as $\tilde{p} = (\tilde{x}, \tilde{y})$ where (\tilde{x}, \tilde{y}) are the 2D pixel coordinates. An image space grasp is illustrated in Figure 2.2.

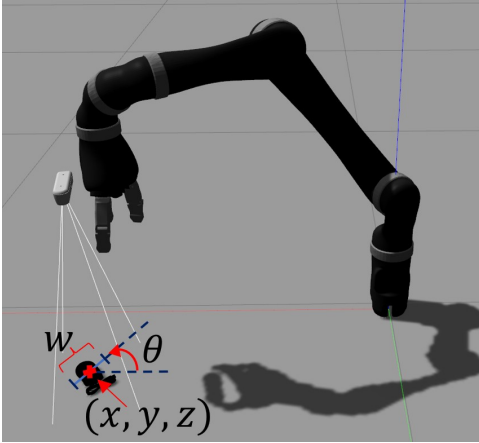


Figure 2.1: World space grasp

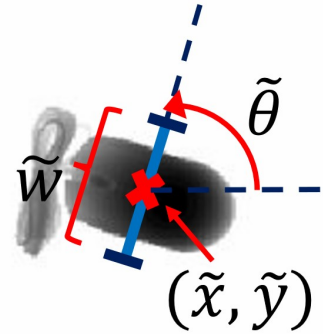


Figure 2.2: Image space grasp

Given the definition of a grasp in image space, a grasp map can be further defined to include the collection of image space grasps from every pixel of an image, such that:

$$\tilde{G} = (Q, \tilde{\Theta}, \tilde{W}) \in \mathbb{R}^{3 \times \tilde{X} \times \tilde{Y}} \quad (2.3)$$

$Q, \tilde{\Theta}, \tilde{W} \in \mathbb{R}^{\tilde{X} \times \tilde{Y}}$ where \tilde{X} and \tilde{Y} are the height and width of the image respectively; $Q, \tilde{\Theta}$, and \tilde{W} are distributions that represent respectively the grasp quality, image space grasping angle and the image space grasping width at every pixel of the image.

Given a grasp map, the image space target grasp \tilde{g}^* is defined by the pixel coordinates $(\tilde{x}^*, \tilde{y}^*)$ that correspond to the highest pixel value of the grasp quality distribution from the grasp map, the angle at the same pixel coordinates in the angle distribution, and the width at the same pixel coordinates in the width distribution, that is:

$$(\tilde{x}^*, \tilde{y}^*) = \underset{(\tilde{x}, \tilde{y})}{\operatorname{argmax}} Q$$

$$\tilde{g}^* = (q^*, \tilde{p}^*, \tilde{\theta}^*, \tilde{w}^*) = (Q_{(\tilde{x}^*, \tilde{y}^*)}, (\tilde{x}^*, \tilde{y}^*), \tilde{\Theta}_{(\tilde{x}^*, \tilde{y}^*)}, \tilde{W}_{(\tilde{x}^*, \tilde{y}^*)}) \quad (2.4)$$

Figure 2.3 presents a grasp map along with the target grasp based on that map.

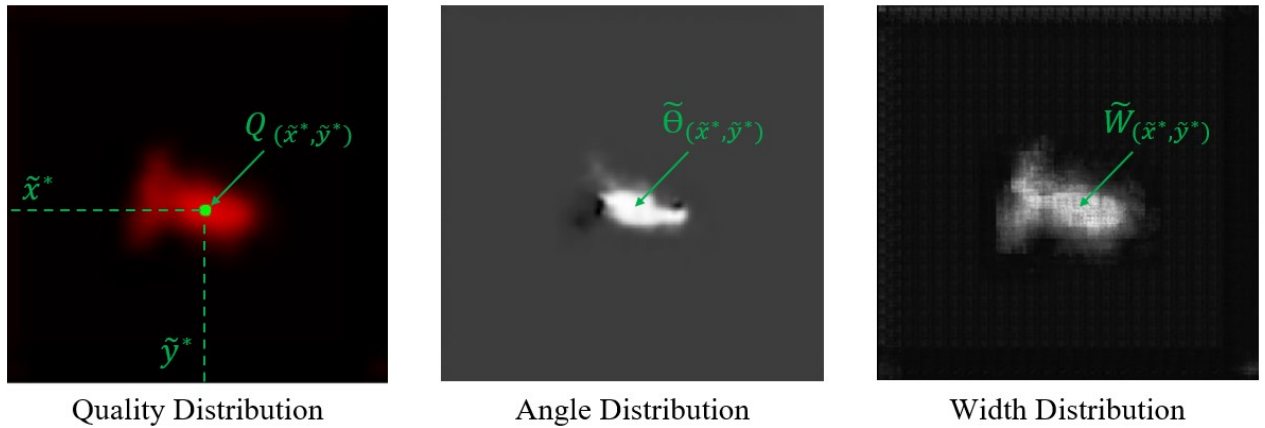


Figure 2.3: Grasp map with target image space grasp

To convert from the image space to the world space, two transformations T_I^C and T_C^W are defined. T_I^C is the transformation from the image space to the employed depth camera's

base frame and T_C^W is the homogeneous transformation from the depth camera’s base frame to the world space frame (most commonly the grasping robot’s base frame). A grasp can then be converted from image space to world space using:

$$g = T_C^W T_I^C(\tilde{g}) \quad (2.5)$$

2.2 Convolutional Neural Network

2.2.1 Network Architecture

The grasp planning problem was formulated, similar to [19], as learning a function M to convert a depth image I to a grasp map \tilde{G} that predicts the grasp quality Q , the grapple orientation angle $\tilde{\Theta}$, and the grapple width \tilde{W} distributions over the pixels of a given depth image. The CNN presented here is based on the GG-CNNs from [19]; it will take a 300×300 depth image as input and provide three 300×300 images as output. Therefore, the CNN’s architecture is fully convolutional. The chosen CNN architecture was inspired by the GG-CNN networks presented in [19] because of their lightweight topology. However, for our network, we added, subtracted, and manipulated some layers based on a series of trials where the number of convolutional layers, type of operations, number, size of filters, and other network attributes were sequentially varied to achieve the highest test accuracy rate and a faster prediction rate than in [19].

The optimal resulting network possesses a fully convolutional architecture, where the input depth image is scaled down to 100×100 from the original 300×300 size. The scaled down image is subjected to seven intermediary convolutional layers. Throughout these layers, a max pooling operation is performed, reducing the image size to 50×50 in the middle. Image expansion is performed using two dimensional bilinear upsampling, and final 2D bilinear upsampling operation is applied after the image goes through the convolutional layers to re-scale the prediction outputs to the original 300×300 input image size. In totality, the network comprises one initial average pooling operation, seven convolutional layers (two of which have dilation), one max pooling operation, an intermediary 2D bilinear upsampling operation, and a final pre-output 2D bilinear upsampling operation. Therefore, the network requires 78 820 parameters and its architecture is summarized in Figure 2.4 and Table 2.1.

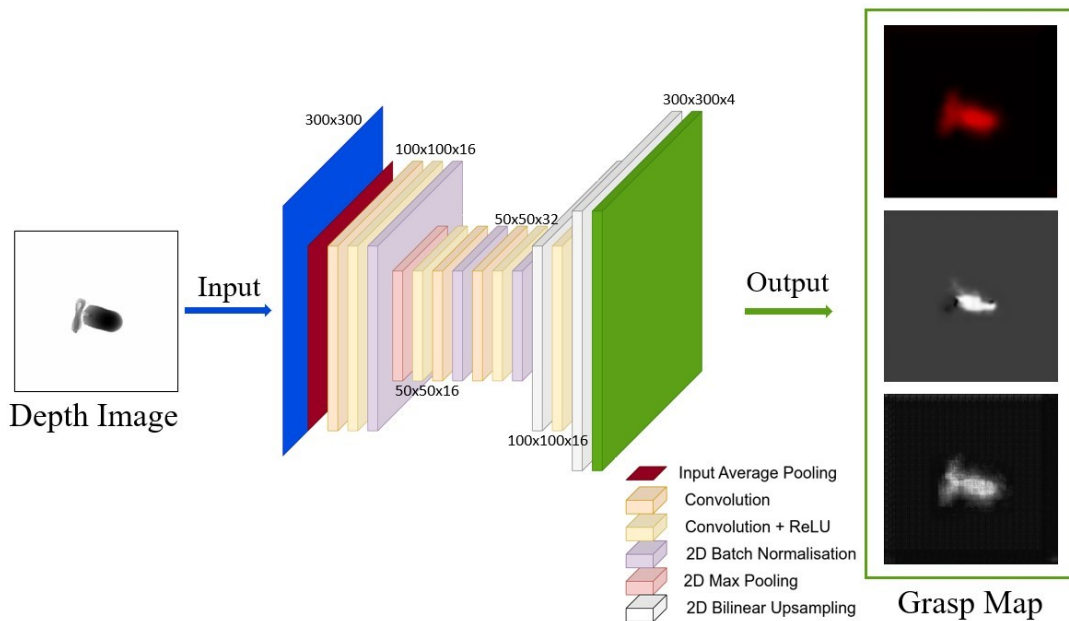


Figure 2.4: Network architecture summary

Layer	Output shape	No. parameters
Input	(1, 300, 300)	0
AvgPool2D	(1, 100, 100)	0
Conv2D	(16, 100, 100)	1952
Conv2D	(16, 100, 100)	12560
ReLU	(16, 100, 100)	0
BatchNorm2D	(16, 100, 100)	32
MaxPool2D	(16, 50, 50)	0
Conv2D	(16, 50, 50)	6416
ReLU	(16, 50, 50)	0
Conv2D	(16, 50, 50)	6416
BatchNorm2D	(16, 50, 50)	32
Conv2D	(32, 50, 50)	12832
Conv2D	(32, 50, 50)	25632
ReLU	(32, 50, 50)	0
BatchNorm2D	(32, 50, 50)	64
UpsamplingBilinear2D	(32, 100, 100)	0
Conv2D	(16, 100, 100)	12816
ReLU	(16, 100, 100)	0
UpsamplingBilinear2D	(16, 300, 300)	0
Conv2D	(1, 300, 300)	17
Conv2D	(1, 300, 300)	17
Conv2D	(1, 300, 300)	17
Conv2D	(1, 300, 300)	17

Table 2.1: Network architecture layers summary

2.2.2 Network Training

A labeled grasping dataset is required to train our neural network to produce grasp predictions. For this work, we chose to use one of the available standardized datasets, the Cornell Grasping dataset¹: it contains RGB-D images of 885 real objects with multiple

¹http://pr.cs.cornell.edu/grasping/rect_data/data.php — As of August 15, 2023, the website is down.

grasp labels per object denoted by rectangles for a total of 5110 labeled grasps. A sample image from the dataset is shown in Figure 2.5.

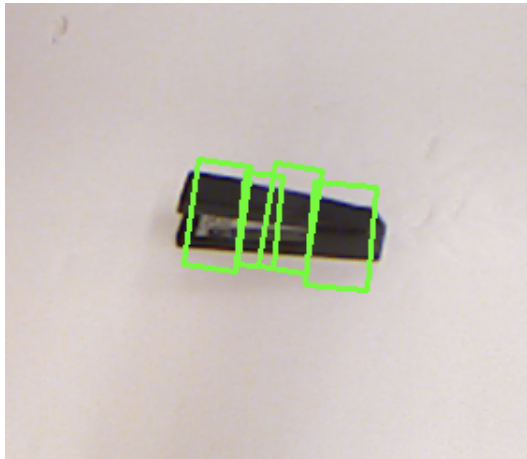


Figure 2.5: Cornell grasping dataset sample

Following the data pre-processing and training framework in [19], the dataset was augmented 10 times with random zooms, crops, and rotations to obtain 8850 images and 51100 labeled grasps. The 640×480 images from the dataset were cropped at their center to form 300×300 images which served as the input to the CNN. The true labels were generated by setting the grasp quality inside the Cornell grasp rectangles to 1 and 0 elsewhere. The grasping angles were based off the orientations of the Cornell rectangles and were decomposed into their respective $\cos 2\theta$ and $\sin 2\theta$ components, as in [19]. Instead of directly outputting the angle distribution, the CNN outputs a cosine and a sine distributions that can be used to infer the grasping angle distribution. This decomposition

facilitates the learning process and allows for the generation of unique angles in the range of $[-\pi/2, \pi/2]$ that can be used by a symmetric robotic gripper. Given $\cos 2\theta$ and $\sin 2\theta$ predictions by the network, the angle is deduced using:

$$\theta = \frac{1}{2} \tan^{-1}\left(\frac{\sin 2\theta}{\cos 2\theta}\right) \quad (2.6)$$

As for the gripper's width, it is inferred from the rectangle label's width by normalizing it from the range of $[0, 150]$ pixels to $[0, 1]$.

The network training was performed by minimizing the total mean squared error:

$$\begin{aligned} MSE_{total} &= MSE_Q + MSE_{\cos 2\tilde{\Theta}} + MSE_{\sin 2\tilde{\Theta}} + MSE_{\tilde{W}} \\ &= \frac{1}{NN_p} \sum_{n=1}^N \sum_{i=1}^{N_p} [(\hat{Q}_i - Q_i)_n^2 + (\cos(2\hat{\Theta}_i) - \cos(2\tilde{\Theta}_i))_n^2 \\ &\quad + (\sin(2\hat{\Theta}_i) - \sin(2\tilde{\Theta}_i))_n^2 + (\hat{W}_i - \tilde{W}_i)_n^2] \end{aligned} \quad (2.7)$$

where variables denoted by a hat represent predicted values and those without a hat represent true labels, N represents the number of training examples and N_p represents the number of pixels per image, $N_p = 90000$ for our CNN.

2.2.3 Evaluation metrics

A grasp’s success was judged by expressing the resulting prediction as a rectangle similar to the Cornell dataset label. A grasp was classified as successful if it fulfilled the conditions for both the Intersection over Union (IoU) and Orientation Difference metrics described below:

- **IoU:** To compute the IoU metric, the areas of overlap and union between the predicted grasp’s rectangle \hat{R}^* and the ground truth rectangle from the Cornell dataset \tilde{R} are determined as shown in Figure 2.6. The area of overlap is then divided by the area of union of the two:

$$IoU = \frac{\hat{R}^* \cap \tilde{R}}{\hat{R}^* \cup \tilde{R}} \quad (2.8)$$

For our network, the IoU metric condition was considered to be satisfied if $IoU \geq 25\%$.

- **Orientation Difference:** The orientation difference is defined as the absolute value of the difference between the angle of the predicted grasp’s rectangle $\hat{\theta}^*$ and the angle of the ground truth rectangle from the Cornell dataset $\tilde{\theta}$ (see Figure 2.7):

$$\Delta\theta = |\hat{\theta}^* - \tilde{\theta}| \quad (2.9)$$

For our network, the orientation difference metric condition was considered to be satisfied if $\Delta\theta \leq 30^\circ$.

These evaluation metrics are commonly used for grasp evaluation in [15–19] which also tackle

grasp prediction and synthesis.

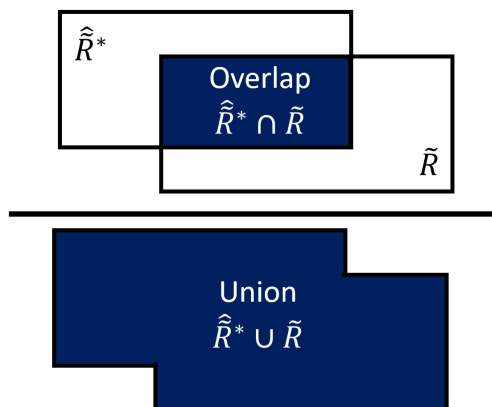


Figure 2.6: IoU metric

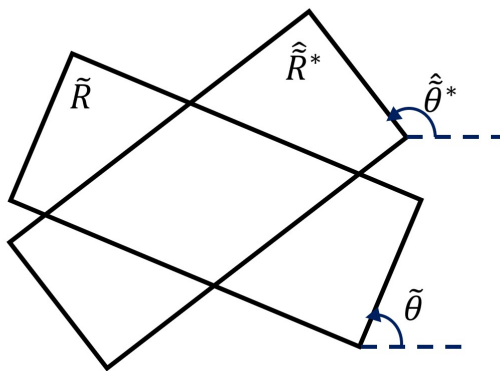


Figure 2.7: Orientation difference metric

2.3 Performance and Comparison

Our network was implemented alongside the GG-CNN and the improved GG-CNN2 models from [19] and the performance of the three networks was compared. The results of training and testing using the introduced evaluation metrics are demonstrated, along with simulated and experimental grasping results using the three networks.

2.3.1 Training and Evaluation

The three networks were trained on the original and the augmented version of the Cornell grasping dataset. The Adam optimizer [34] with a learning rate of 0.001 was used for training purposes. Testing was performed using 10-fold cross validation and the resulting accuracies

based on the IoU and orientation difference metrics are presented in Table 2.2.

Network	Train Augmented	Test Augmented	Accuracy
GG-CNN	No	No	78.4%
GG-CNN2	No	No	80.7%
Our Network	No	No	84.7%
GG-CNN	No	Yes	41.4%
GG-CNN2	No	Yes	44.9%
Our Network	No	Yes	46.9%
GG-CNN	Yes	No	75.8%
GG-CNN2	Yes	No	77.2%
Our Network	Yes	No	84.9%
GG-CNN	Yes	Yes	64.6%
GG-CNN2	Yes	Yes	73.4%
Our Network	Yes	Yes	83.8%

Table 2.2: Our network vs. GG-CNNs 10-Fold cross validation results

As seen in Table 2.2, our network outperforms the GG-CNNs in all test cases. When the dataset was not augmented, all three networks performed somewhat similarly with our network leading with an accuracy of 84.7% while GG-CNN and GG-CNN2 achieved accuracies of 78.4% and 80.7%, respectively. However, when the dataset was augmented, our network was significantly ahead, with an accuracy of 83.8% compared to 64.6% and 73.4% for GG-CNN and GG-CNN2, respectively.

Table 2.2 also highlights the importance of data augmentation in our case, since all three networks’ performance was significantly degraded (up to $\sim 40\%$ decrease in accuracy) when they were trained on the original unaugmented Cornell dataset and evaluated on a test set from the augmented dataset. The three networks performed similarly in this case: our

network had a slightly higher accuracy of 46.9% than GG-CNN and GG-CNN2 that had respective accuracies of 41.4% and 44.9%. These results suggest that data augmentation is necessary for the network’s predictions to be generalizable and to achieve successful grasping in a real-world scenario where objects can be cluttered, in different orientations, and at different distances.

To produce the final network model for each of the three networks, they were trained for 50 epochs on 90% of the augmented Cornell dataset and validated on the remaining 10%. The networks’ training losses, validation losses and accuracies are presented in Figures 2.8, 2.9 and 2.10, respectively. As the figures demonstrate, our network achieved training and validation losses that are consistently lower than those of the GG-CNNs and a validation accuracy that was generally higher than that of the GG-CNNs.

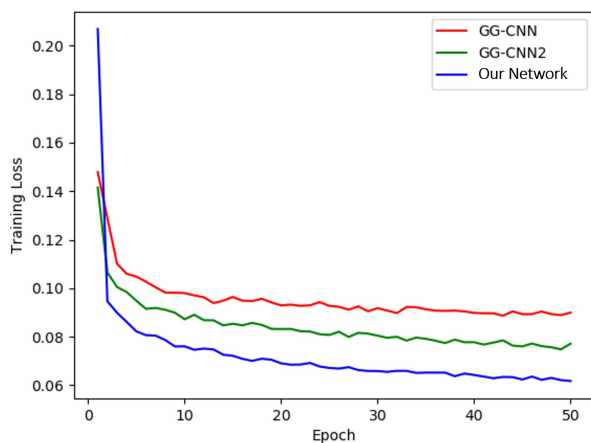


Figure 2.8: Networks training loss

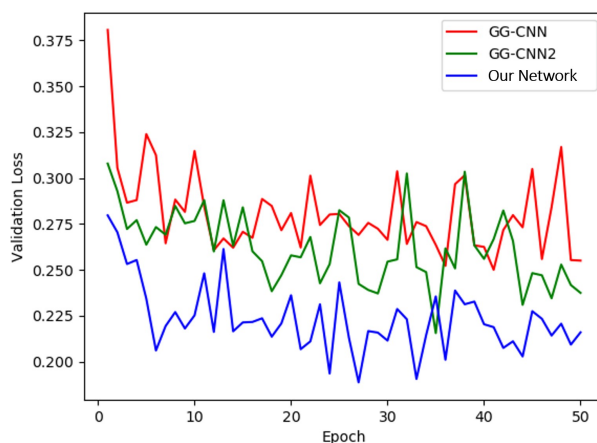


Figure 2.9: Networks validation loss

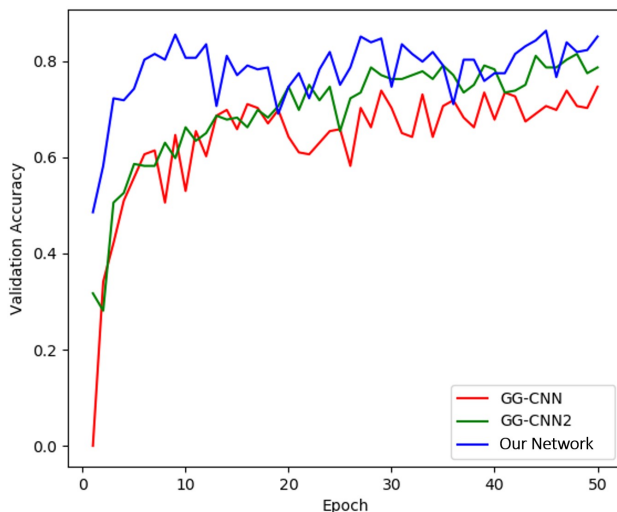


Figure 2.10: Networks validation accuracy

The specific epochs for the final models of the networks were chosen based on a trade-off between validation loss and accuracy. They are presented in Table 2.3.

Network	Epoch	Val Acc
GG-CNN	41	73%
GG-CNN2	35	78%
Our Network	27	85%

Table 2.3: Chosen final network models

2.3.2 Simulation Grasping

To further test the performance of the developed network model, a Gazebo simulation was set up with a 6-DoF Kinova Jaco 2² manipulator arm with a Kinova KG-3 3-finger gripper (model

²<https://www.kinovarobotics.com/product/gen2-robots>

j2n6s300). This robot was chosen for our simulation since it is available in our laboratory (Aerospace Mechatronics Laboratory) and was therefore used for the experimental grasping attempts of the next section. As for the perception, a virtual Intel RealSense D435 depth camera was attached to the manipulator's wrist to capture the depth images that will serve as the input to the neural network. The Gazebo simulation setup is illustrated in Figure 2.11.

To perform grasping attempts, a synthetic Gazebo objects set was created. It contains 10 simulated items that were chosen at random from the Gazebo models database. The objects from the Gazebo set are presented in Figure 2.12. Note that the choice of items to include in the Gazebo set was random with the only condition being that the items must fit within the robot's gripper fingers.

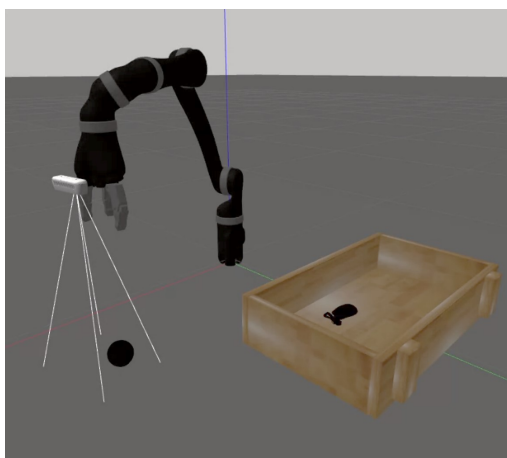


Figure 2.11: Gazebo simulation setup



Figure 2.12: Gazebo objects set

To generate our simulated results, 10 open-loop grasp attempts were performed on each of the 10 items of the Gazebo set using each of the three network models. In other words, 100 grasps were performed using each network, for a total of 300 grasp attempts for our simulated results. A grasp was considered to be successful if the robot was able to grasp the object and transport it to a marked drop-off location. The total grasping accuracy along with the hardest object to grasp (object with the lowest number of successful grasps) were recorded for each of the networks and are presented in Table 2.4.

Object	Successful Grasps	Failed Grasps
GG-CNN	77%	Computer Mouse
GG-CNN2	80%	Boat
Our Network	86%	Shoe

Table 2.4: Gazebo simulation grasping attempts results

As the table shows, our network achieved the highest success rate of 86% followed by GG-CNN2 and GG-CNN at 80% and 77% respectively. Although the success rates of the three networks were somewhat similar, it is noticeable that GG-CNN2 was only able to improve on GG-CNN’s success rate by 3% while our network presented a 9% increase in success rate compared to GG-CNN and a 6% increase compared to GG-CNN2. As for the hardest objects for the networks, it seems that GG-CNN and GG-CNN2 face difficulties with objects with complex and intricate geometry like the computer mouse and the boat, while our network struggles while attempting to grasp objects with thin edges like the shoe.

2.3.3 Experimental Grasping

To validate the simulated results experimentally, a new household objects set was created from 10 randomly selected items that are likely to be found in a household or in a pile of clutter. Once again, only the size of the items was taken into consideration when they were selected, since they must fit between the fingers of the Kinova Jaco 2's KG-3 gripper that was used for the experiments. An Intel RealSense D435 depth camera was attached to the Jaco 2 arm's wrist and ROS was used as middleware to facilitate communication between the software and hardware components. The household set is illustrated in Figure 2.14 and the Jaco 2 manipulator that was used is presented in Figure 2.13.



Figure 2.13: Kinova Jaco2 robotic arm



Figure 2.14: Household objects set

Using the household set, two experiments were performed:

- **Experiment 1:** Similarly to the simulation tests, 10 grasp attempts were performed

on each of the 10 items of the household set, resulting in a total of 100 grasps that were performed by each of the three networks. Grasps were considered to be successful if the robot grasped the item and transported it to a designated drop-off basket. The success rate and the hardest object to grasp for each of the networks were recorded and are presented in Table 2.5.

Network	Grasping Success	Hardest Object
GG-CNN	73%	Pliers
GG-CNN2	75%	Pliers
Our Network	83%	Mug

Table 2.5: Experiment 1 grasping attempts results

Similarly to the simulated results, our network achieved the highest success rate of 83%, followed by GG-CNN2 and GG-CNN with success rates of 75% and 73%, respectively. As for the hardest objects to grasp, the GG-CNNs had the lowest chances of success with the pliers while our network faced the most difficulties with the mug. These findings consolidate the conclusion that was made in simulation: the GG-CNNs struggle with complex geometry while our network struggles with thin edges.

- **Experiment 2:** To compare the networks’ performance on grasping in clutter, the household set’s items were placed in a bin to form a cluttered pile. The bin was furthermore shaken to randomize the arrangement of the items inside and the robot was tasked to empty out the bin by transporting all its items to a designated drop-off

location. We recorded the number of grasps it took for the robot to transport all the items out of the bin along with the amount of time that was taken to perform this task. The results are presented in Table 2.6.

Network	Time	Number of Grasps
GG-CNN	8m 31s	17
GG-CNN2	7m 49s	15
Our Network	5m 12s	10

Table 2.6: Experiment 2 clutter grasping results

As shown in the table, our network was significantly more efficient than the GG-CNNs at clearing the pile of clutter. It was able to grasp all 10 items with 10 grasp attempts, which is 41% fewer grasps than GG-CNN (17 attempts) and 33% fewer grasps than GG-CNN2 (15 attempts). It is worth noting that although our network required 10 grasps to empty the pile, not all grasps were successful. One grasping attempt on the mug item failed but the robot was able to pick up two items with a single grasp, which compensated for the failed attempt. As for the time taken to empty the bin, our network was the fastest with a time of 5m 12s; that is 39% faster than GG-CNN (8m 31s) and 33% faster than GG-CNN2 (7m 49s).

Chapter 3

Grasp Planning Pipeline

This chapter introduces the grasp planning pipeline, which takes a set of log parameters as input (pose, length, diameter) and outputs the location and orientation for a log-loader's grapple to move to where the grapple performs the grasping operation by closing its tongs. The presented work relies on segmentation to generate the aforementioned input parameters to the pipeline and a modified version of the CNN from Chapter 2 to decide on the output grapple pose. To progress from input to output, logs are clustered and a virtual environment created in Gazebo is employed as an intermediate step, where logs are recreated, post-processed, and positioned based on their input parameters. A virtual depth camera is then positioned in the virtual Gazebo world at a point that provides good exposure of the logs to generate a target grasping position and orientation for the log-loader's grapple. A schematic of the pipeline is presented in Figure 3.1.

The contents of this chapter are heavily based on [1].

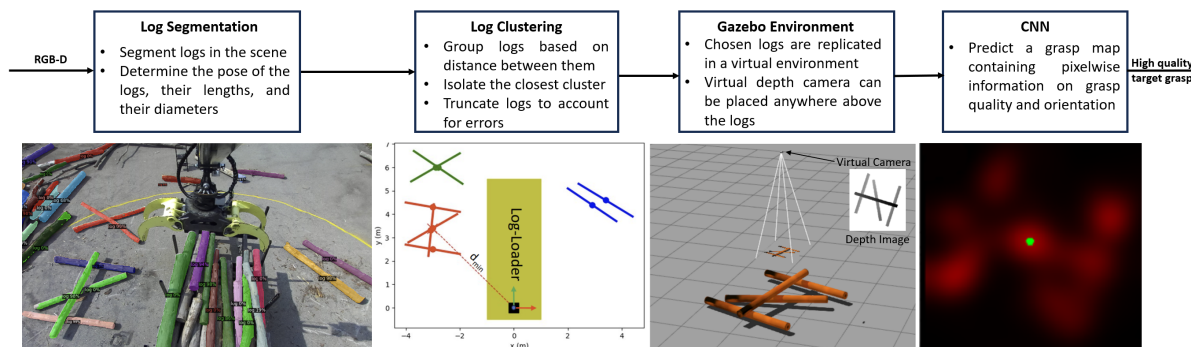


Figure 3.1: Grasp planning pipeline schematic

3.1 Log Segmentation

Since the grasp planner requires the pose and characteristics of individual logs in order to replicate them in the virtual environment, log detection and segmentation were selected as one of the more suitable methodologies to extract this information. The work in [35] introduces a log segmentation dataset named TimberSeg 1.0 that contains 220 images of 2500 segmented logs with their bounding boxes and mask annotations. The majority of images were collected through dash-cams that were placed on forwarder machines during their operation in forests over several months and through various weather conditions. Conveniently, the Mask2Former segmentation network from [36] was trained on the TimberSeg dataset and implemented by the authors of [35] on the FPInnovations log-loading test-bed employed for implementation and experimentation on our grasp planning pipeline. Their implemented approach relies on a Swin-B transformer backbone that is pre-trained on the ImageNet-22k dataset [37] to extract features from RGB images

and generate segmented masks. The Detectron2 library from Facebook was used by the authors of [35] to train the model on TimberSeg 1.0 and limit its application to the single-class detection task of identifying wood logs. The segmented log masks coupled with a point-cloud obtained from a depth camera are then combined to infer the length and diameter of the logs, and localize them in the scene around the log-loading test-bed. This log segmentation and characterization solution was fully developed and implemented on the FPInnovations log-loading test-bed by the authors of [35]. Figure 3.2 shows a sample case of segmentation on logs at FPInnovations log-loading test-bed as seen from a camera on a log-loader.



Figure 3.2: Logs segmentation sample, FPInnovations log-loading test-bed

3.2 Log Piles Clustering

Once the logs in the scene have been identified and characterized, they undergo clustering to separate piles from each other based on the distance between them. To perform the

clustering operation, the Agglomerative Clustering algorithm from the scikit-learn¹ python library is employed. In this hierarchical method, the algorithm starts by defining every data point (the centroid of every log) as its own cluster. Individual clusters are then recursively combined to create larger clusters based on their similarity. For our application, similarity is based on the euclidean distance between log centroids specifically since we are working in 3-dimensional space. Using a maximum linkage criterion, the distance between two clusters is defined as the euclidean distance between the two furthest points of the clusters. This criterion was chosen since it provides the possibility to ensure that all logs in the same cluster are within a certain threshold distance from each other. For our case, a maximum distance threshold of two meters is used. In other words, only clusters whose furthest points are less than two meters apart will be merged and therefore only logs within a two meter radius of each other will be part of the same cluster at the end.

After all clusters have been formed, the center of each cluster is computed by averaging the positions of the logs that belong to the cluster. The distances from the world frame's origin to each of the cluster centers are then calculated and the cluster with the shortest distance is chosen to be the target for grasp planning. Logs belonging to non-chosen clusters are ignored beyond this point in the pipeline. Figure 3.3 shows a sample clustering output.

The clustering operation was not performed in the grasp planning pipeline's evaluation in Chapter 4 since only single clusters of logs in different configurations were considered.

¹<https://scikit-learn.org/>

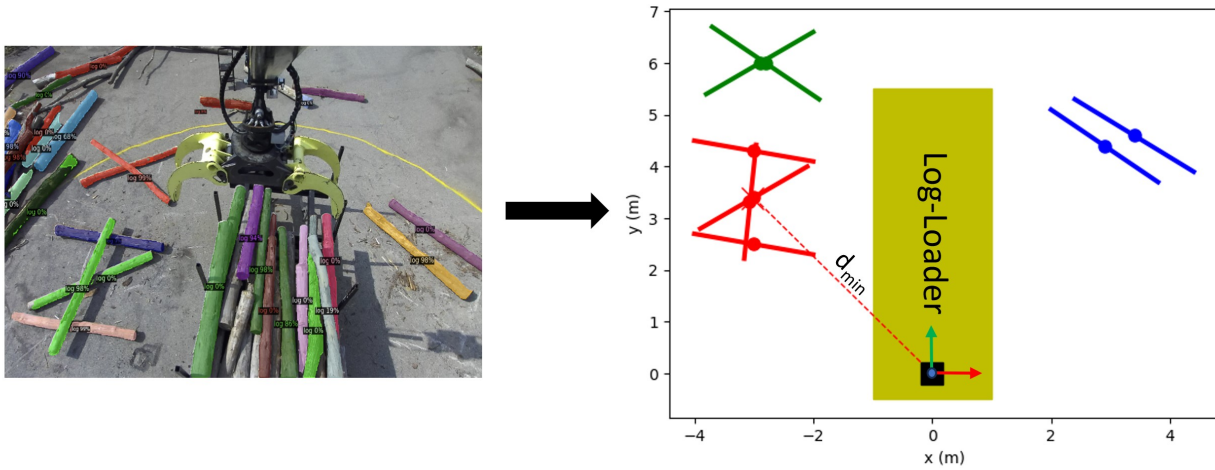


Figure 3.3: Clustering sample output

3.3 Virtual Environment

Once the targeted cluster has been identified, virtual logs are instantiated in a blank Gazebo world as perfect cylinders with the poses, lengths, and diameters that correspond to the logs in that cluster. The logs' lengths are reduced by the grapple's width and an extra 10% that was empirically found to avoid potential grasps at their edges, and to partially mitigate segmentation inaccuracies in length estimation. A random striped orange-like material is assigned to the cylinders for the sole purpose of visual clarity, since grasp planning relies exclusively on depth information. In the Gazebo world, the log-fixed frames coincide with their centroids and the world frame is set to represent the crane's base frame; thus, the positions and orientations of virtual logs in Gazebo match those of real logs from the chosen cluster with respect to the machine's base frame.

As noted earlier, one of the unique features of our grasp planning scheme is that planning is carried out in the virtual (Gazebo) environment, with a virtual camera. To determine the placement of the camera in the Gazebo world, we proceed as follows. First, we define a right-hand log-fixed frame at the geometric center of each log, with the x -axis pointing along the length of the log. The grasping operational area is then defined by computing the bounding box that contains the entirety of the generated virtual logs. This is done by determining the positions of the ends of each log \mathbf{p}_i^{GW} ($i = 1, 2$ for the two ends) in Gazebo world frame (GW).

$$\mathbf{p}_{1,2}^{GW} = \mathbf{p}_L^{GW} + \mathbf{R}_L^{GW} \mathbf{p}_{1,2}^L \quad (3.1)$$

where \mathbf{p}_i^L denotes the corresponding end positions in the log frame ($\mathbf{p}_1^L = \begin{bmatrix} -L/2 & 0 & 0 \end{bmatrix}^T$ and $\mathbf{p}_2^L = \begin{bmatrix} L/2 & 0 & 0 \end{bmatrix}^T$), \mathbf{p}_L^{GW} is the position of the log-fixed frame and \mathbf{R}_L^{GW} is the rotation matrix from the Gazebo world frame to the log-fixed frame. The latter is obtained through the XYZ Tait-Bryan angles parameterizing the orientation of the log such that:

$$\mathbf{R}_L^{GW} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \quad (3.2)$$

where ϕ , θ , and ψ are the angles of rotation around the world frame's x -, y -, and z -axis respectively.

Having computed all extremity locations, we locate the minimum and maximum position component ranges in the three directions, $\{x, y, z\}_{min/max}$, and form a cuboid with

sides of length $(x_{max} - x_{min})$, $(y_{max} - y_{min})$, and $(z_{max} - z_{min})$. A virtual depth camera can now be created above the logs, at any desired location. Alternatively, it can swiftly follow a specified path while taking continuous depth shots to allow for grasp planning from multiple angles and averaging grasp predictions as new depth images are processed. The log-loader can then navigate to the target location immediately, once virtual grasp planning is completed. For the simulations and experiments presented in Chapter 4, a RealSense D435 depth camera was instantiated at a static position \mathbf{p}_C^{GW} , 3 meters above the centroid of the cuboid defining the grasping operational area. This camera placement provides an

uncut view of the logs from a reasonable distance. i.e., $\mathbf{p}_C^{GW} = \begin{bmatrix} (x_{max} + x_{min})/2 \\ (y_{max} + y_{min})/2 \\ (z_{max} + z_{min})/2 + 3 \end{bmatrix}$ m.

Figure 3.4 shows the replicated logs of a sample chosen cluster in Gazebo.

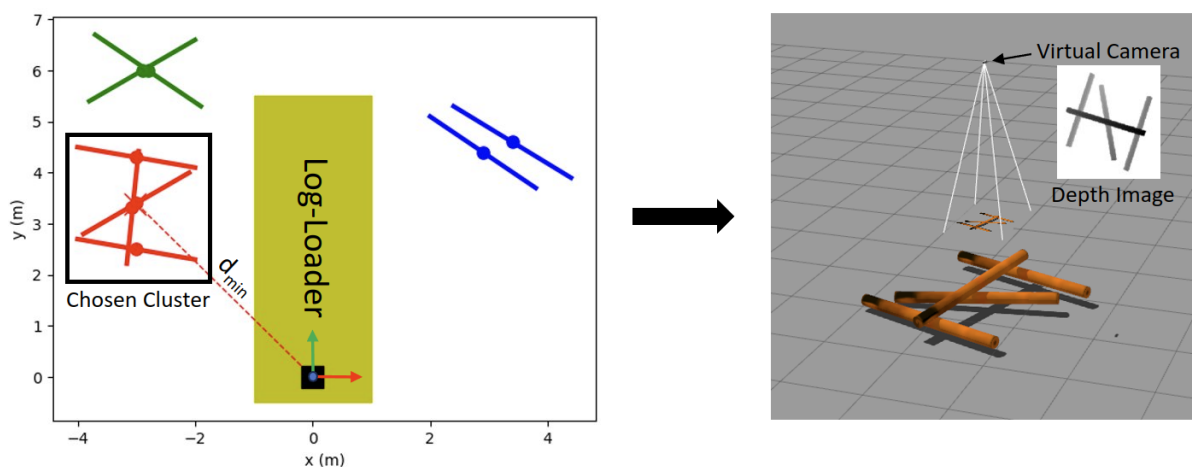


Figure 3.4: Replication of sample chosen cluster in Gazebo virtual environment

3.4 Convolutional Neural Network

Since log-loaders do not typically allow for controlling the opening of their grapple beyond fully open and fully closed, the grasp definition introduced in Chapter 2 is modified to exclude the width information. In this context, a grasp's definition in world space becomes:

$$g = (q, p, \theta) \quad (3.3)$$

and in the image space:

$$\tilde{g} = (q, \tilde{p}, \tilde{\theta}) \quad (3.4)$$

A sample grasp on logs is presented in the world space and in the image space in Figures 3.5 and 3.6, respectively.



Figure 3.5: World space grasp on log

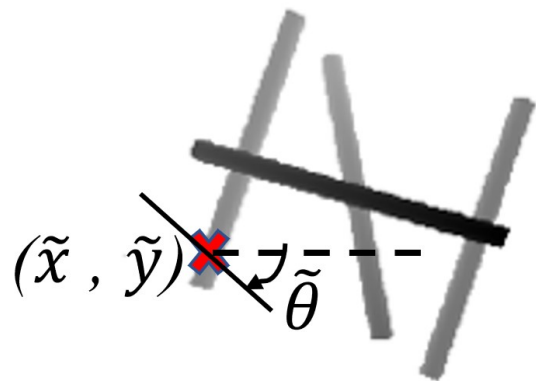


Figure 3.6: Image space grasp on log

The resulting grasp map representing the collection of image space grasps at every pixel

can then be represented as:

$$\tilde{G} = (Q, \tilde{\Theta}) \in \mathbb{R}^{2 \times \tilde{X} \times \tilde{Y}} \quad (3.5)$$

Furthermore, the predicted optimal grasp after omitting the grapple's width becomes:

$$\tilde{g}^* = (q^*, \tilde{p}^*, \tilde{\theta}^*) = (Q_{(\tilde{x}^*, \tilde{y}^*)}, (\tilde{x}^*, \tilde{y}^*), \tilde{\Theta}_{(\tilde{x}^*, \tilde{y}^*)}) \quad (3.6)$$

A sample grasp map and predicted optimal grasp are shown in Figure 3.7

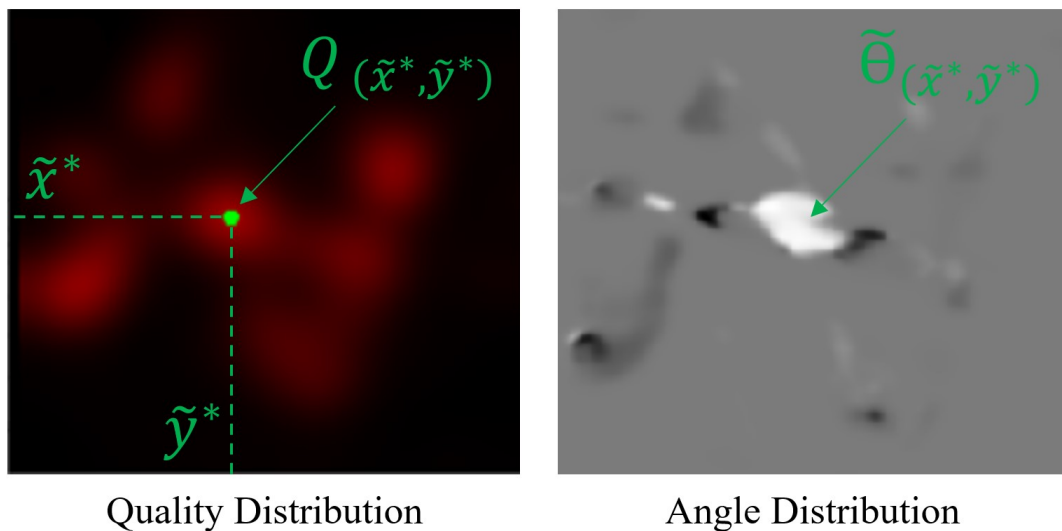


Figure 3.7: Grasp map from log image and predicted optimal grasp

Therefore, for the log-loading task, the CNN takes a 300×300 depth image as input and outputs two 300×300 images representing the pixelwise quality and angle distributions. Besides the output layers, the architecture of the network is identical to the one from Chapter 2. The nature of the input and output images make it that the grasp planning network is

completely agnostic to the fact that it is trying to grasp logs and to the number of logs that are present in the input image. Therefore, the number of grasped logs is a stochastic outcome of the grasping maneuver, i.e., there is no way to know how many logs will get grasped.

The modified network was trained on the same augmented data from the Cornell general grasping dataset that was presented in Chapter 2. In this case, training was performed by minimizing the following total mean squared error:

$$\begin{aligned}
 MSE_{total} &= MSE_Q + MSE_{\cos 2\tilde{\Theta}} + MSE_{\sin 2\tilde{\Theta}} \\
 &= \frac{1}{NN_p} \sum_{n=1}^N \sum_{i=1}^{N_p} [(\hat{Q}_i - Q_i)_n^2 + (\cos(2\hat{\Theta}_i) - \cos(2\tilde{\Theta}_i))_n^2 + (\sin(2\hat{\Theta}_i) - \sin(2\tilde{\Theta}_i))_n^2]
 \end{aligned} \tag{3.7}$$

The Adam optimizer with a learning rate of 0.001 was used for training. To test the accuracy of the network before employing it in our grasp planning pipeline, 10-fold cross validation was applied on the augmented Cornell dataset and resulted in an 83.8% accuracy based on the IoU and orientation metrics. To generate the final model to be used, the network was trained on 90% of the augmented Cornell dataset and validated on the remaining 10%. The training and validation losses are provided in Figure 3.8 while the validation accuracy is provided in Figure 3.9.

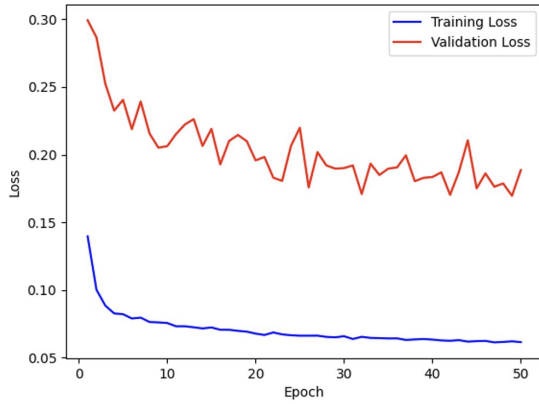


Figure 3.8: Training and validation loss

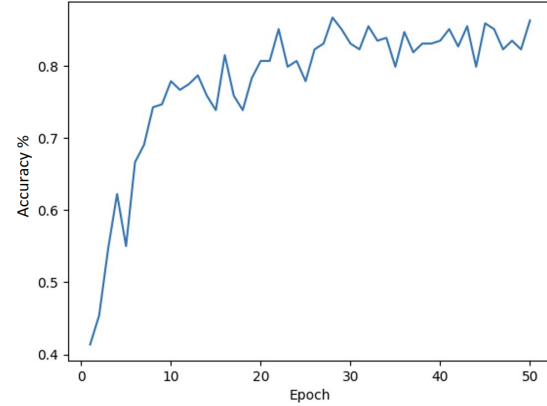


Figure 3.9: Validation accuracy

The model from epoch 45, with a validation accuracy of 86%, was chosen to be used for the grasp planning pipeline's evaluation in Chapter 4. Note that although the difference between the training and validation losses appears to be significant in Figure 3.8, it is consistent with the difference observed during the training and validation of the GG-CNNs, as seen in Figures 2.8 and 2.9.

Chapter 4

Simulation and Experimental

Validation

This chapter presents the trials that were performed to validate the grasp planning pipeline from Chapter 3. Section 4.1 provides a general description of the log-loading test-bed that was employed during the experimentation, along with its components, topology, and the environment that it can interact with. In addition, an overview of the hardware and software components that ensure the operation of the test-bed is presented. Section 4.2 presents the Omniverse Isaac Sim simulation environment that was used during the simulated trials. The experimental procedure that was followed through the trials is introduced in Section 4.3, and the simulated and experimental results are reported in Section 4.4. Finally, Section 4.5 demonstrates the influence of virtual camera placement on the grasp planning outcome.

The contents of this chapter are heavily based on [1].

4.1 Log-loading Test-bed

4.1.1 Crane

The experiments and simulations presented in this thesis were conducted on the log-loading test-bed (The Crane) at FPInnovations, Pointe Claire, Quebec (see Figure 4.1). This facility was built up from the originally purchased commercial system PALMS 4.71¹. The arm, referred to as the crane of the test-bed, is integrated on a fixed-base platform which houses the hydraulic system to drive the crane; a bunk (also referred to as trailer or basket) holds the logs. The crane itself has the topology typical of a log-loading machine, such as the forwarder: it is a seven degree-of-freedom under-actuated arm with the first four joints (RRRP) to position the tip of the boom, followed by two passive rotary joints to support the grapple (C4 model from PALMS) and the last joint (the rotator) to orient the grapple with respect to the logs. The grapple tongs are actuated open-loop to fully open or close the grapple. The crane has a maximum reach of 7.1 m. The joint and frame assignments are shown in Figure 4.1, the DH parameters are stated in Table 4.1, and the joint ranges are stated in Table 4.2.

¹<https://www.palms.eu/forest-cranes?productID=58>



Figure 4.1: FPInnovations log-loading test-bed

Joint	$\alpha[\text{rad}]$	$a[\text{m}]$	$d[\text{m}]$	$\theta[\text{rad}]$	Joint Type
1	0	-0.06	1.364	θ_1	Revolute
2	$\pi/2$	0	0	θ_2	Revolute
3	0	3.049	0	$\theta_3 + \pi/2$	Revolute
4	$\pi/2$	0	$d_4 + 2.16$	0	Prismatic
5	$\pi/2$	0	0	G	Revolute - Passive
6	$\pi/2$	-0.165	0	$\pi/2$	Revolute - Passive
7	$-\pi/2$	0	0.452	θ_7	Revolute

Table 4.1: Denavit-Hartenberg parameters table

For the passive joints, when the grapple has no sway, $\theta_5 = G$ where G is the required angle to keep \vec{z}_7 pointing along the direction of gravity and $\theta_6 = \pi/2$ rad.

Joint	Min Value	Max Value
θ_1 [rad]	-1.745	1.745
θ_2 [rad]	-0.384	1.309
θ_3 [rad]	-3.086	0.035
d_4 [m]	0.13	1.8
θ_5 [rad]	n/a	n/a
θ_6 [rad]	n/a	n/a
θ_7 [rad]	$-\pi$	π

Table 4.2: Log-loader joint ranges table

To enable closed-loop joint control, the actuated revolute joints have been instrumented with absolute joint encoders (Magres–EAM360-B-CANopen²) measuring the output (link) angle. The extension of the fourth (prismatic) joint is measured with a magnetic sensor/band combination (MSA501/MBA501).³ In addition, the crane is instrumented with a ZED 2i stereo camera⁴ rigidly mounted on a stick link (see Figure 4.1); the camera provides the image (RGB-D) data necessary for log segmentation.

4.1.2 Hardware and Software Architecture

The control hardware enabling the closed-loop joint control of the crane includes a laptop communicating through Ethernet with a PLC. The PLC receives information on desired joint trajectories from the laptop and current joint states (positions and velocities) from the joint encoders, and in turn generates the commands to the hydraulic valves of the crane using

²<https://www.baumer.com/us/en/product-overview/rotary-encoders-angle-sensors/industrial-encoders-absolute/36-mm-integrated/eam360-b/eam360-b—canopen/p/27928>

³<https://www.siko-global.com/en-ca/products/magline-magnetic-linear-and-angular-measurement/magnetic-sensors/msa501>

⁴<https://store.stereolabs.com/products/zed-2i>

a PD control law through the CANopen communication protocol. The ZED camera relays RGB-D images to the laptop where log segmentation and characterization is performed, and the grasp planning pipeline is applied to determine a target grasping spot. Since ROS Noetic was used as middleware for communication between the software components, path planning and trajectory generation were done through the "MoveIt"⁵ package once a grasping target has been determined. The "RRTstar" algorithm from the Open Motion Planning Library in MoveIt is employed to perform path planning and generate trajectories that avoid self-collisions and collision with the log-loader's basket. The hardware and software architectures are illustrated in Figure 4.2 and 4.3 respectively.

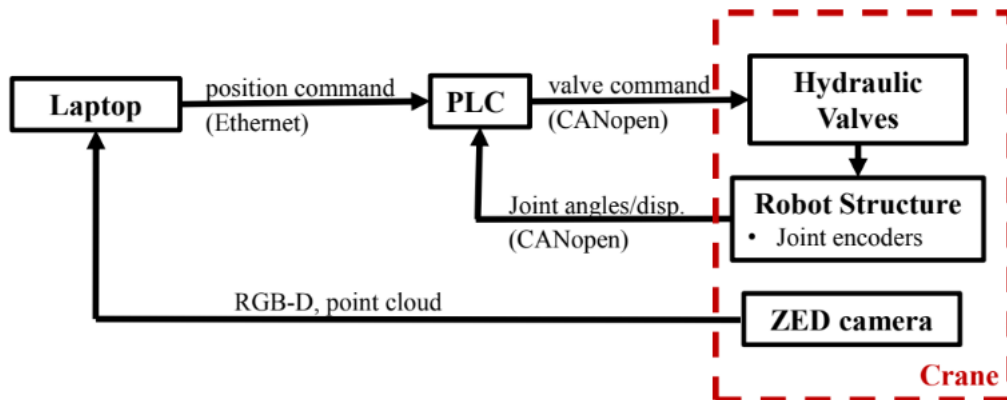


Figure 4.2: Hardware architecture

⁵<https://moveit.ros.org/>

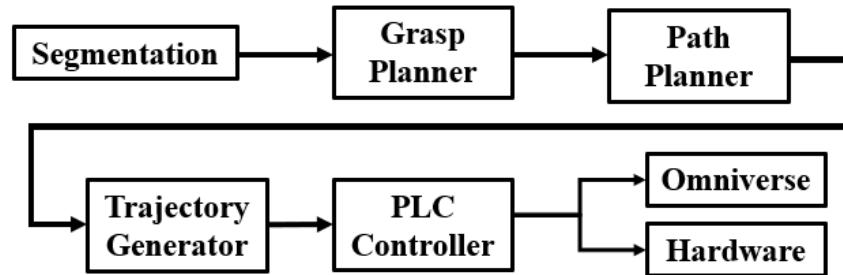


Figure 4.3: Software architecture

4.1.3 Testing Environment

The crane test-bed is located in a fenced outdoor parking area of FPInnovations that contains a few dozen wood logs. The testing environment can be seen in Figure 4.4.



Figure 4.4: Log-loading test-bed environment

The set of logs employed for the grasping experiments (samples shown in Figures 4.5 and 4.6) comprised around a dozen logs of dry red pine, with lengths in the range of 2.5-2.8 m and diameters in the range of 0.1-0.3 m. The logs were placed on asphalt ground around the log-loading test-bed. Only logs that are to be grasped were kept within the reach of the crane, while the others were pushed out of the working area.



Figure 4.5: Logs sample



Figure 4.6: Log pile sample

4.2 Simulation Environment

A model of the test-bed was created using Isaac Sim robotic simulator from the NVIDIA Omniverse platform. Starting with the 2D CAD models of the crane components, 3D models of the parts were generated, meshed and saved in STL format. The STL meshes were then used to build a URDF file describing the architecture of the log-loader. In order to be

compatible with the Isaac Sim environment, the URDF file was subsequently converted into a USD scene using the Isaac Sim URDF importer extension to create the virtual test-bed. The masses of the main components of the crane were obtained from the manufacturer and the other inertial parameters estimated based on the mass and geometry information. Hydraulics of the test-bed are not modelled; hence, the joints' positions are directly controlled using a PD control law. A camera with matching properties to the ZED 2i was positioned in the latter's place and configured to relay RGB-D data through an Isaac Sim built-in ROS bridge. Information on joints' positions and velocities, in addition to joint commands were also relayed from and to Isaac Sim within ROS topics through built-in ROS bridges. Figure 4.7 shows the simulated log-loading test-bed in the Isaac Sim simulation environment.



Figure 4.7: Log-loading test-bed Isaac Sim simulation environment

4.3 Experimental Procedure

To quantify the success of the presented grasping pipeline in Chapter 3, we define 12 test configurations to replicate common arrangements of logs that a log-loader might encounter during its operation (see Table 4.3). These configurations present opportunities for grasping a single log in some cases and multiple logs (crossed logs and mini-piles) in other cases. The configurations were reviewed by a forestry researcher⁶ and confirmed as representative of routine log-picking operations in the forest.




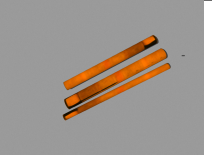

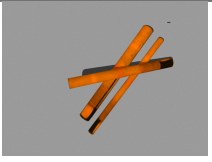
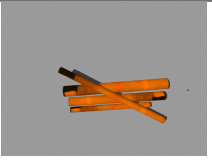
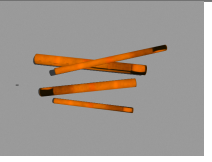
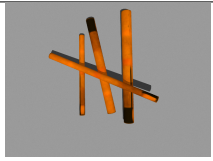
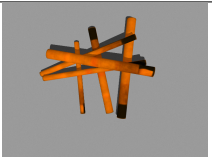
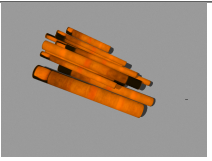
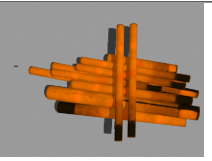
Log Config.	1	2	3	4
Picture				
Log Config.	5	6	7	8
Picture				
Log Config.	9	10	11	12
Picture				

Table 4.3: Log configurations for simulated and experimental trials

⁶Peter Hamilton reviewed all configurations. He is a Senior Researcher at FPInnovations, Pointe-Claire, QC H9R 3J9, Canada

For every configuration, five grasping attempts are performed for a total of 60 grasping tests. A single test includes execution of the grasp planning pipeline, the crane repositioning to the grasping target location, the grapple commanded to perform the grasp, and finally, the crane raising the grasped log(s). For each of the five attempts per configuration, the logs were arranged to maintain the configuration, but placed at random locations and orientations. The positions of the logs were also cycled between each other during same-configuration attempts (a log that was on top of a pile during an attempt may be at the bottom of the pile on the next attempt). The lengths and diameters of the logs that were used for different configurations were chosen at random. A grasp was considered to be successful if by the end of a test cycle, the log-loader had grasped and raised at least one log above ground level, and no parts of the grasped log(s) are touching the ground. Furthermore, successful grasps were manually categorized into optimal and sub-optimal grasps by visual inspection. Optimal grasps make the most sense for a certain configuration and are likely to be chosen by an operator, such as, grasps of the top log in configurations 3 and 9, or grasps at the geometric center of single and parallel logs in configurations 1 and 2. In addition, the grapple angle is aligned with the direction of logs to be grasped during an optimal grasp. As for sub-optimal grasps, they are considered in three categories:

1. **Non Intuitive (NI)**: for grasps that log-loading operators would not normally perform, such as logs that are under other logs and grasps at the edges of single or parallel logs.

2. **Segmentation Error (SE)**: for grasps that were not optimal because of segmentation inaccuracies, such as, logs missing or inaccurate log localization.
3. **Misaligned Angle (MA)**: for grasps with grapple angles that did not align well with the direction of the targeted logs.

4.4 Results

4.4.1 Simulated Results

Table 4.4 presents the number of successful simulated grasp attempts out of five for every configuration and their optimal/sub-optimal categorization.

Out of the 60 performed attempts, the log-loader succeeded at grasping and lifting a log or a bunch of logs during 59 (sum of successful grasps in Table 4.4) tries (98.33% success). Multiple log grasping was consistently observed for configurations that allowed it: all logs in configurations 2, 4, 5, 6, 7, and more than one log in configurations 10, 11, and 12. Out of the 59 successful attempts, 48 were optimal and 11 were sub-optimal, the latter including 4 NI and 7 SE cases, with no MA occurrences. Segmentation errors were expected to be dominant in the simulated trials since the segmentation network was trained on images of real wood logs. The virtual wood material and the resolution of the simulation's image also played a role in deteriorating the segmentation quality. Observed segmentation errors included logs shifted from their real positions, logs completely missed and unsegmented,

individual logs that were segmented as multiple logs because part of them was obstructed, and logs with widely inaccurate lengths or diameters. Note that segmentation inaccuracies occurred frequently in the simulated trials but were only reported when they caused sub-optimal grasps. Indeed, the grasp planner was able to produce optimal grasps even with multiple segmentation problems. For the only failed grasp attempt for configuration 5, the combination of a non intuitive grasp location at the edge of the logs and a segmentation error in the form of a shift in the logs' positions caused the grapple to completely miss its target.

Log Config.	1	2	3	4	5	6
Grasp Success	5	5	5	5	4	5
↳ Optimal	5	3	4	5	3	4
↳ Sub-Optimal	n/a	SE, NI	NI	n/a	SE	NI
Log Config.	7	8	9	10	11	12
Grasp Success	5	5	5	5	5	5
↳ Optimal	3	5	4	2	5	5
↳ Sub-Optimal	SE, SE	n/a	NI	SE, SE, SE	n/a	n/a

Table 4.4: Simulated grasps outcomes

4.4.2 Experimental Results

Experiments were carried out on the crane test-bed over two days (August 17th and 24th, 2022). Both days were a mix of sunny and cloudy conditions, providing a wide range of lighting conditions for the camera. Weather and visibility conditions have the potential to influence log segmentation results only since the remainder of the pipeline relies on the replicated logs in Gazebo. The same 12 configurations were employed and the same result reporting procedure from Section 4.3 was followed for the experimental trials.

Table 4.5 presents the number of successful experimental grasp attempts out of five for every configuration and their optimal/sub-optimal categorization. Out of 60 attempts, the log-loader successfully grasped and lifted a log or a bunch of logs in 58 (sum of successful grasps in Table 4.5) tries (96.67% success). As in the simulated trials, multiple log grasping was consistently observed throughout configurations that allowed it: all logs in configurations 2, 4, 5, 6, 7, and multiple logs in configurations 10, 11, and 12. Of the 58 successful attempts, 51 were optimal while the remaining 7 were sub-optimal, the latter consisting of 4 NI, 1 SE, and 2 MA categories. Although some segmentation and localization errors were present in the experimental trials, they were less frequent than in the simulated trials. As previously mentioned, this decrease in segmentation errors from 7 in simulation to 1 experimentally is to be expected when operating on the real test-bed since the TimberSeg 1.0 dataset that was used to train the segmentation algorithm contains images of real wood logs only, with 37 of these images being taken at the FPInnovations log-loader test-bed site. Testing on real

hardware, however, presented new sources of errors, such as inaccuracies in the positioning of the grapple due to grapple positioning errors (estimated to be on the order of 0.1 m); these are likely caused by the flexibility of the crane and the limitations of PD control. The point-cloud generated by the ZED 2i camera was noisy enough to cause the segmentation and log-localization to be consistently slightly in error. Nonetheless, the grasp planner was able to circumvent these errors during most grasping attempts. The two grasp failures were due to segmentation missing a large portion of the logs and generating abnormally large diameters. This caused the reconstruction of log pieces with a width larger than their length in Gazebo, which in turn resulted in predicted grapple angles to be nearly 90° off the correct angles, ultimately leading to failed grasp attempts.

Log Config.	1	2	3	4	5	6
Grasp Success	5	5	5	5	5	5
↳ Optimal	5	5	4	3	3	5
↳ Sub-Optimal	n/a	n/a	NI	NI, NI	MA, SE	n/a
Log Config.	7	8	9	10	11	12
Grasp Success	5	5	5	4	4	5
↳ Optimal	5	5	4	4	4	4
↳ Sub-Optimal	n/a	n/a	NI	n/a	n/a	MA

Table 4.5: Experimental grasps outcomes

4.5 Virtual Camera Placement and Grasp Planning

In this section, we will demonstrate the ability to improve the grasp planning pipeline’s results by altering the position of the virtual camera in Gazebo, further highlighting the benefits of having a virtual camera provide the input to the grasp planner. Two categories of logs arrangement will be presented where the grasp planning CNN provides non-optimal grasps, and these will be improved by simply changing the virtual camera’s pose.

4.5.1 Logs at the Edges of Input Images

Although, this kind of arrangement was not present in the configurations that were used for our experimentation, it was noticed throughout the use of the developed CNN that it tends to predict a failed grasp whenever the input image contains logs that are at the edges of the input image (see Figure 4.8). The highest quality point from the quality distribution in such scenarios tends to be at the center of the image rather than on one of the logs as illustrated in Figure 4.9. This situation tends to occur when multiple almost-parallel logs that belong to the same cluster are present in the image but are well separated from each other, resulting in image appearance such as shown in Figure 4.8. Since the camera is always placed above the center of a bounding box that encompasses the logs, this situation can only occur when the bounding box is wide enough, and therefore, does not occur when a single log is present or multiple logs are condensed in the same area. Note that the center area of the image must also be empty for this problem to occur. The particular choice of grasp of the CNN for

such situations, as illustrated in Figure 4.9, may be due to the fact that the grasping targets in the images that were used to train the model were always the center of attention of the image and were never at the image's boundaries.



Figure 4.8: Input depth image from 3 meters above logs

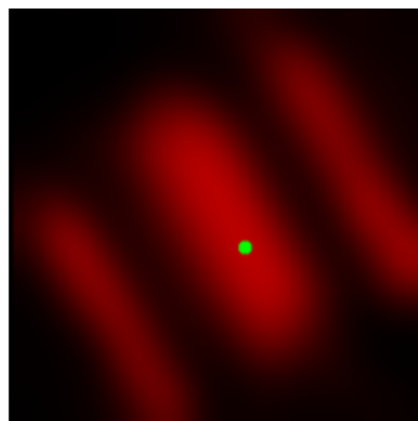


Figure 4.9: Quality Distribution for camera placement of 3 meters above logs



Figure 4.10: Input depth image from 4 meters above logs

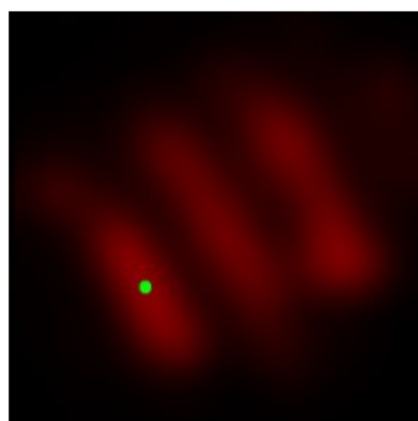


Figure 4.11: Quality Distribution for camera placement of 4 meters above logs

Since this issue stems from the location of the logs in the image, it can be fixed by vertically raising the virtual camera above the logs, widening the area on the ground that it can capture, which causes the logs to move towards the center of the image. Figures 4.8 and 4.9 show the input image and the output quality distribution for a virtual camera placement of 3 meters above the logs (default value used for results in Sections 4.3 - 4.4), while Figures 4.10 and 4.11 show them for a placement of 4 meters above the logs. The grasp planner fails to provide a good grasping location when the camera is 3 meters above the log but succeeds when the camera's height is raised to 4 meters.

To automate the process of raising the camera as necessary, we introduce the following steps:

1. An inner bounding box is defined in the input depth image for the purpose of ensuring that logs are completely contained within it. The box is chosen to exclude a width of 40 pixels from each side of the original image, meaning that logs must be within the center 220×220 pixels of the original 300×300 input image (See Figure 4.12).
2. To determine whether logs are within or outside the bounded area, the Canny algorithm [38] from the `cv2`⁷ python computer vision package is used. The Canny algorithm uses gradients and rapid intensity changes in the image along with edge tracking techniques to robustly detect edges within the image. For our purpose, the Canny algorithm provides a blank image with highlighted contours of the logs, as seen in Figure 4.13.

⁷<https://opencv.org/>

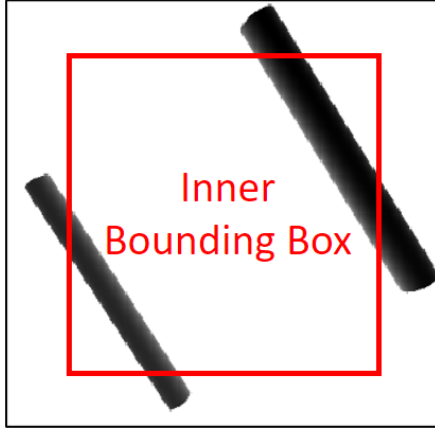


Figure 4.12: Inner bounding box

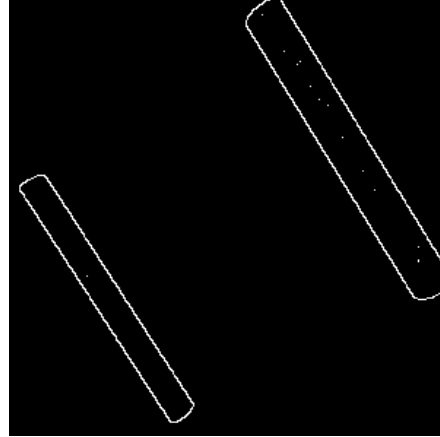


Figure 4.13: Canny algorithm edge detection

3. The resulting image from the Canny algorithm is checked for any pixels of the log contours that fall outside of the defined inner box. If such pixels exist, the virtual Gazebo camera is raised by 10 centimeters.
4. Steps 1 to 3 are repeated until no contours are detected outside of the 220×220 inner box.

This process is summarized with the pseudo-code shown in Algorithm 1.

Algorithm 1 Virtual Camera Position Adjustment using Log Edge Detection

- 1: $box_{inner} \leftarrow (220, 220)$ at center of $image_{input} \leftarrow (300, 300)$
 - 2: **repeat**
 - 3: $contours_{logs} \leftarrow \text{Canny}(image_{input})$
 - 4: **if** any $pixel \in contours_{logs}$ is outside of box_{inner} **then**
 - 5: $camera_{height} \leftarrow camera_{height} + 10$ cm
 - 6: **end if**
 - 7: **until** all $pixels \in contours$ are within box_{inner}
-

4.5.2 Non Intuitive Grasps on Individual Logs

Throughout the experimentation, it was noticed that for individual logs, the network predicted grasps that are sometimes at the center of the log while other times at its edge. Upon further investigation, it became apparent that the choice of the grasping location (center versus edge) depended on the orientation of the targeted log in the input image. Furthermore, for some orientations, there were notable regions of high grasp quality on the CNN’s output quality map that were not on the log itself.

To test this hypothesis, four logs of different length and diameter (see Table 4.6) were created separately in a Gazebo environment with a virtual depth camera placed at 3 meters above each of their centers. For each log, we rotated the camera by 0.05 rad incrementally from $-\pi/2$ rad to $\pi/2$ rad in order to change the orientation of the log in the captured image. At every step, the grasp planning CNN was used to predict a grasping location and orientation, and the following cases were used to assess the grasp prediction:

- **O**: The grasp prediction is optimal. The grasping location is closer to the center of the log than its edges and the predicted grasping angle is within 30° of the log’s orientation.
- **NI-L**: The grasp location is non intuitive: the target grasping location is closer to one of the logs’ edges than its center.
- **NI-A**: The predicted grasping angle is non intuitive: the predicted angle is off by more than 30° from the log’s orientation but less than 60° .

- **F-L:** The predicted grasping location is a grasp planning failure: the target grasp is not on the log.
- **F-A:** The predicted grasping angle is a grasp planning failure: the predicted angle is off by more than 60° from the log's orientation.
- **R:** The grasp quality map had regions of high grasp quality that were not on the log.

Samples of failed, non-intuitive, and optimal grasp plans are shown in Figure 4.14.

Log	Length (m)	Diameter (m)
1	3.4	0.3
2	2.8	0.25
3	2.1	0.2
4	3	0.2

Table 4.6: Log characteristics

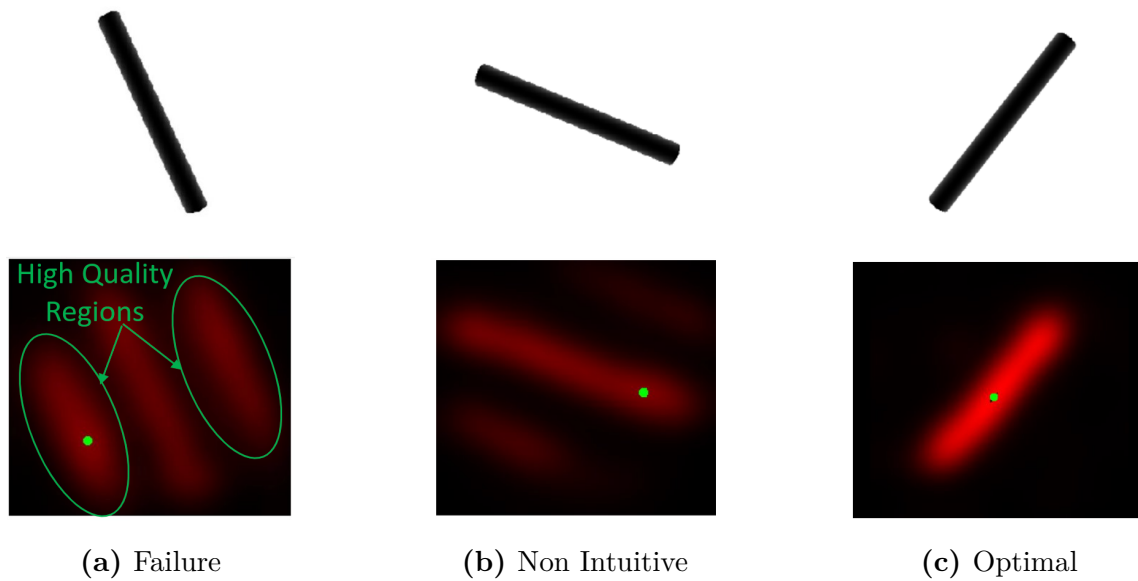


Figure 4.14: Failed, Non Intuitive, and Optimal grasp planning samples

The obtained recordings followed similar trends for all four logs, suggesting that the log's diameter and length had a minor effect on the grasp prediction quality compared to the logs' orientation in the CNN's input depth image, which was the major factor affecting the grasp predictions. The results are reported in Tables 4.7, 4.8, 4.9, and 4.10.

Angle (rad)	Result
$-\pi/2 \rightarrow -1.37$	O, R
-1.32, -1.27	NI-L, R
$-1.22 \rightarrow -1.07$	F-L, R
$-1.02 \rightarrow -0.72$	O, R
$-0.67 \rightarrow -0.42$	NI-L, R
$-0.37 \rightarrow -0.17$	NI-A, R
$-0.12 \rightarrow -0.02$	F-A, R
0.03 \rightarrow 0.58	NI-L
0.63 \rightarrow 0.88	O
0.93 \rightarrow 1.53	NI-L

Table 4.7: Log 1 orientation test results

Angle (rad)	Result
$-\pi/2 \rightarrow -1.27$	O, R
$-1.22 \rightarrow -1.12$	F-L, R
$-1.07 \rightarrow -0.72$	O, R
$-0.67 \rightarrow -0.32$	NI-L, R
$-0.27 \rightarrow -0.02$	O-R
0.03	F-A
0.08 \rightarrow 0.18	NI-L
0.23 \rightarrow 0.38	O
0.43 \rightarrow 0.58	NI-L
0.63 \rightarrow 0.88	O
0.93 \rightarrow 1.33	NI-L
1.37 \rightarrow 1.53	O

Table 4.8: Log 2 orientation test results

Angle (rad)	Result
$-\pi/2 \rightarrow -1.32$	O, R
$-1.27 \rightarrow -1.12$	NI-L, R
$-1.07 \rightarrow -0.37$	O, R
$-0.32 \rightarrow -0.02$	NI-L, R
0.03, 0.08	NI-A
0.13, 0.18	F-A
0.23 \rightarrow 1.13	O
1.18 \rightarrow 1.28	NI-L
1.33 \rightarrow 1.53	O

Table 4.9: Log 3 orientation test results

Angle (rad)	Result
$-\pi/2 \rightarrow -1.32$	NI-L, R
-1.27, -1.17	F-L, R
$-1.12 \rightarrow -0.97$	NI-L, R
-0.92, -0.87	O, R
$-0.82 \rightarrow -0.02$	NI-L, R
0.03 \rightarrow 0.13	F-A
0.18 \rightarrow 0.38	O
0.43, 0.48	NI-L
0.53 \rightarrow 0.98	O
1.03 \rightarrow 1.53	NI-L

Table 4.10: Log 4 orientation test results

The following observations can be made from the tables:

- the high grasp quality regions that are not on the log seem to only exist when the log’s orientation is negative (in $[-\pi/2, 0]$ rad).
- These regions seem to have maximum quality when the log’s orientation is in the range of $[-1.22, -1.07]$ rad and therefore they often get chosen as the grasping target, resulting in a grasp planning failure.
- The grasp planner tends to predict failed grasps due to a significant misalignment in the predicted grasping angle when the log’s orientation is around 0, in the range of $[-0.12, 0.18]$ rad.
- The grasp planning CNN seems to perform better in the positive log orientation range, and specifically in the range of $[0.63, 0.88]$ rad where the grasp predictions were optimal for all four logs.

These observations suggest that the grasp planner can be influenced to favor optimal grasps by changing the orientation of the camera above the logs and ensuring that $\theta_{log} - \theta_{camera} \in [0.63, 0.88]$ rad. Therefore, for further verification, we generated 60 logs of random lengths (between 1.5 and 3.5 meters) and diameters (between 0.15 and 0.3 meters) at random orientations in Gazebo. For each of the logs, the virtual camera was placed 3 meters above them such that $\theta_{camera} = 0$ first (the camera always has the same orientation), and $\theta_{log} - \theta_{camera} = 0.79$ rad after (the camera’s angle is adjusted for every log to maintain the 0.79 rad

difference between their angles). The angle difference of 0.79 rad (45 degrees) was randomly chosen from the [0.63, 0.88] range. The number of optimal, non intuitive, and failed grasp plans for each of the two cases was recorded and the results are provided in Table 4.11.

Camera Orientation	$\theta_{camera} = 0$	$\theta_{log} - \theta_{camera} = 0.79$ rad
Optimal	35	57
Non Intuitive	22	3
Failure	3	0

Table 4.11: Grasp planning results based on camera orientation

As shown in the table, at $\theta_{camera} = 0$ rad, 58.33% of the predicted grasps were optimal, 36.67% were non intuitive, and 5% were failed grasp plans. As for when the camera's orientation was varied with every log to maintain $\theta_{log} - \theta_{camera} = 0.79$ rad, 95% of the predicted grasps were optimal, 5% were non intuitive, and none of them failed. These results clearly indicate that the grasp planning network's predictions are more likely to be optimal at certain camera orientations.

Chapter 5

Log-Counting and Grapple Opening

Achieving full autonomy in log-loading requires developing multiple processes besides grasp planning. The ability to characterize the log-loader's grapple and count its grasped logs is necessary to enhance grasping accuracy and enable autonomous inventory management. In this context, this chapter introduces a multi-branch CNN that predicts a log-loader's grapple opening and the number of logs that it is carrying. Section 5.1 presents the U-Net segmentation CNN which is used to identify the grapple and the carried logs. The U-Net based architecture to make the required predictions is presented in Section 5.2. The network's training process along with the collected training datasets are presented in Section 5.3. To test the developed network, simulated and experimental trials are performed on the FPInnovations log-loading test-bed from Chapter 4; results are reported in Section 5.4.

5.1 U-Net

Motivated by the complexity of biomedical images and the lack of biomedical data that is required to train typical segmentation networks, the U-Net CNN architecture was originally developed as a solution for semantic segmentation of biomedical images that requires few training images [39]. It has won the ISBI challenge for segmentation of neuron structures in 2015 by achieving IoU accuracies that are significantly higher than its competitors on two biomedical datasets that only contained 35 and 20 partially annotated images, respectively. U-Net's success rate and efficiency at segmentation has made it one of the most notable segmentation networks, especially when it comes to datasets with limited data, and therefore, it was adapted and applied to fields beyond bio-medicine, like segmentation in unstructured environments [40], defect and crack detection [41, 42], remote sensing [43, 44], and many others.

The name of the architecture comes from the network being shaped like the letter "U". It is fully convolutional and contains a down-sampling and an up-sampling portions. The down-sampling part consists of a double 3×3 convolution layer followed by ReLU activation and a 2×2 max-pooling operation. This process is repeated four times and the number of channels is doubled with each repetition. As for the expansive part, the features map is up-sampled with 2×2 up-convolution and then concatenated with the features from the down-sampling part of the network through skip connections. A double 3×3 convolution layer with ReLU activation is then applied. This process is repeated four times as well, with

the number of channels being halved with each repetition. The final layer of the network is a 1×1 convolutional layer that maps the number of channels of the features map at the end of the expansive part of the network to a number of channels that is equal to the number of segmentation classes. The U-Net architecture for an RGB input image of size 572×572 and 2 segmentation classes is shown in Figure 5.1.

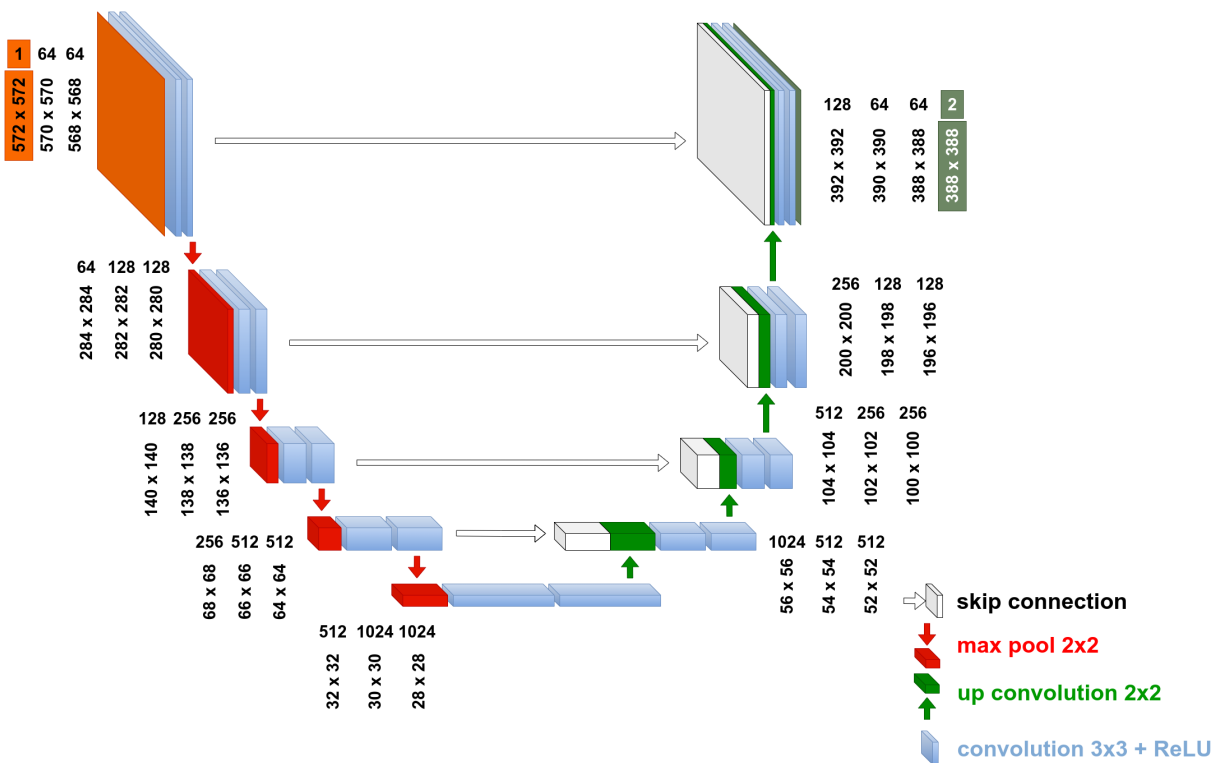


Figure 5.1: Original U-Net architecture

5.2 Grapple Opening and Log Counting Architecture

Given an RGB-D input image of a log-loader's grapple, the developed multi-branch network ultimately predicts the grapple's opening and the number of wood logs that it is carrying.

The process from input to output is summarized in the block diagram of Figure 5.2.

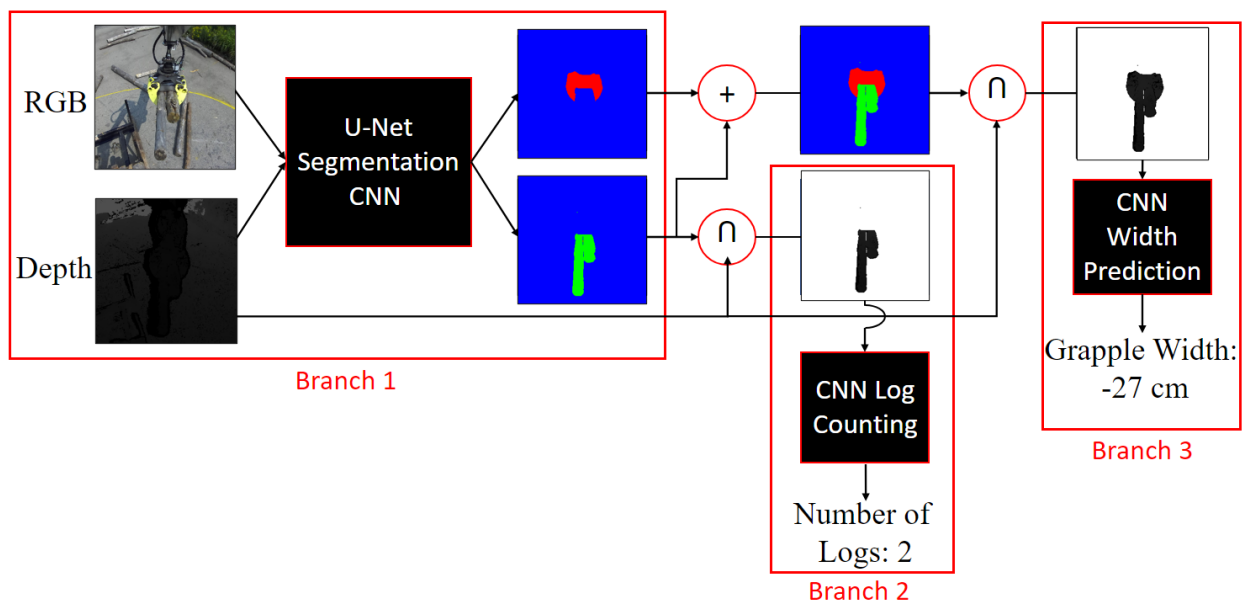


Figure 5.2: Multi-branch network block diagram

A U-Net based CNN is first used to differentiate three classes: the grapple, the carried logs, and the background in the image. The standard U-Net architecture was modified to accept RGB-D data and zero padding was added to the convolutional layers to maintain the original size of the input image at the output of the network. Batch normalization layers were also added after every convolution. The employed U-Net architecture can be seen in Figure 5.3.

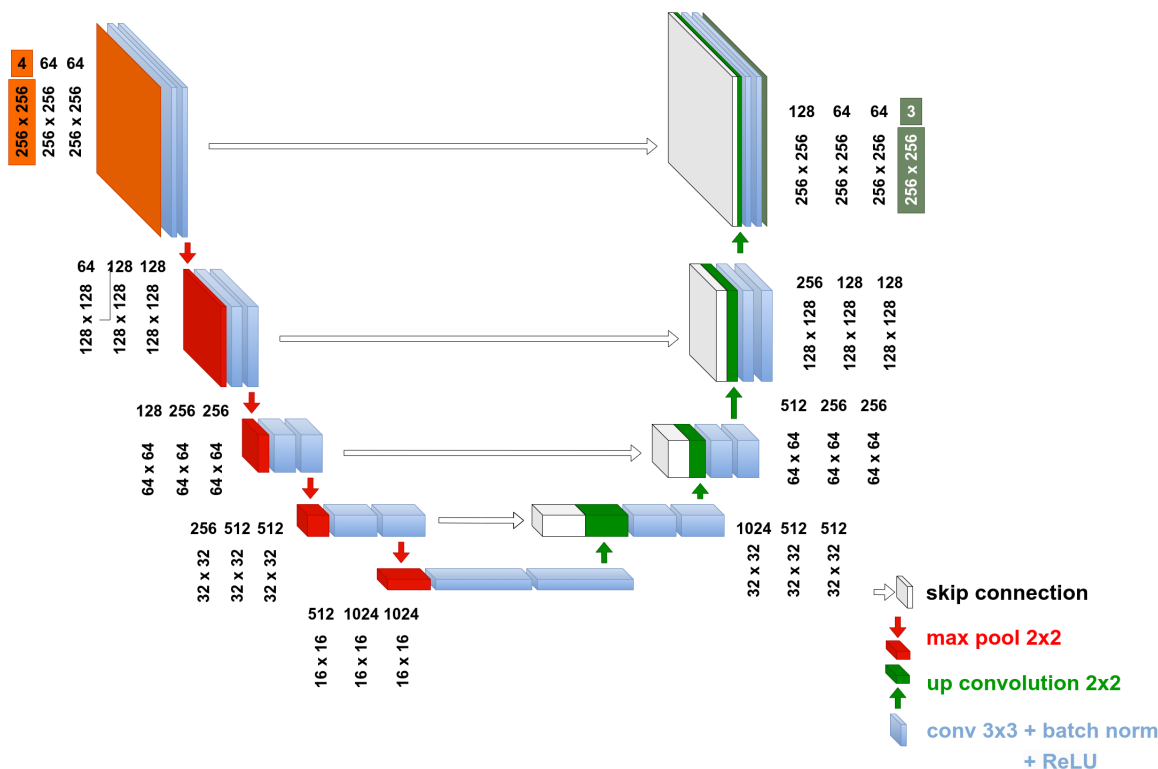


Figure 5.3: Our U-Net architecture

Given the segmentation masks, the mask portraying the carried logs is intersected with the depth information that was input to the U-Net segmentation network to obtain an image that contains depth information about the carried logs only. The logs-specific depth image is then input to the log-counting CNN that will predict the number of grabbed logs (no logs, 1 log, 2 logs, 3 logs). The maximum number of 3 logs was reasonably chosen based on the size of the log-loader and the available logs at the FPInnovations test-bed, and to limit the required amount of training data. The log-counting CNN's architecture is presented in

Figure 5.4.

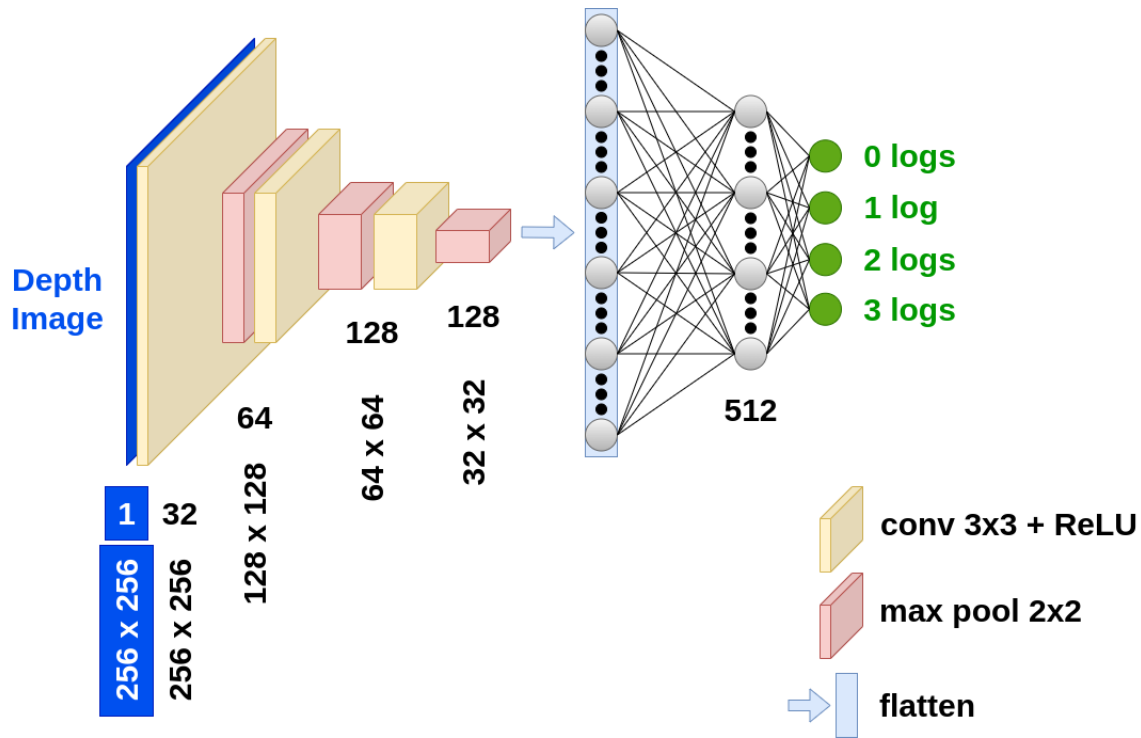


Figure 5.4: Log-counting network branch architecture

For the grapple opening estimation, the segmentation masks portraying both the grapple and the carried logs are intersected with the input depth information to obtain an image with depth information that is specific to the grapple and its grabbed logs only. The logs' information was not excluded for the grapple opening prediction because grabbed logs often occlude parts of the grapple's tongs, and therefore, the network might tend to overestimate the grapple's opening if it has no knowledge about the presence of these logs. The grapple and logs' depth image is then fed to another CNN branch that is tasked to predict the distance between the grapple's tongs. The grapple opening CNN has the same architecture

as the log-counting CNN except for the final layer of connected neurons where it has one neuron that outputs the opening value. The architecture is shown in Figure 5.5.

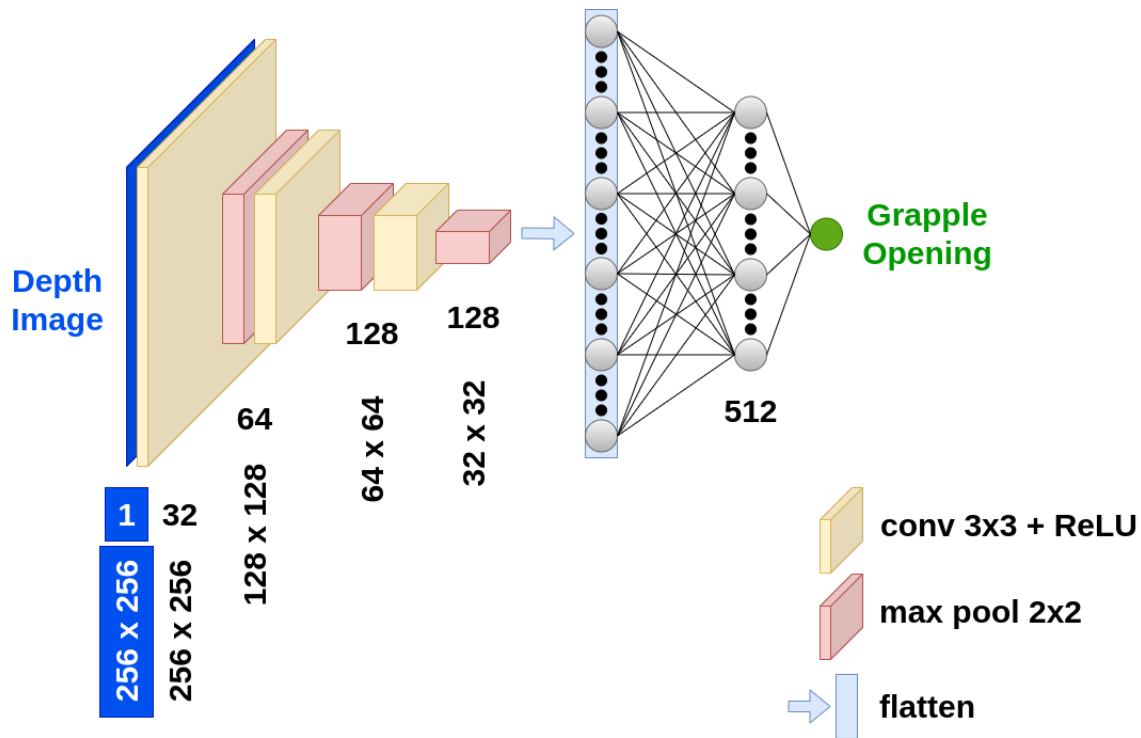


Figure 5.5: Grapple opening network branch architecture

5.3 Training and Evaluation

5.3.1 Simulation Environment and Model

To employ this architecture in simulation and on a real log loader, two versions of U-Net must be trained, one to segment the real grapple and logs, and one for the simulated grapple and logs. Therefore, RGB-D data from the real log-loader's grapple and the simulated grapple

is required. However, since the log-counting and opening-prediction networks rely on depth data only, they can be trained with data from simulation only, and deployed directly on the real log-loader using a sim-to-real approach. In order for sim-to-real to be as effective as possible, the simulated model of the log-loader's grapple must match exactly with the real log loader's grapple. Therefore, a detailed 3D model of the grapple was created based on 2D CAD drawings from the FPIinnovations log-loader's grapple and was used to update the grapple in the Isaac Sim log-loader simulation environment that was presented in Chapter 4. The real and simulated grapples are illustrated in Figures 5.6 and 5.7 respectively.

Furthermore, to make the depth images captured in simulation resemble those from the real log-loader, the parameters of the log-loader mounted camera in the simulation (field of view and focal length) were altered to match those of the ZED 2i camera that is mounted on the FPIinnovations log-loader. Figures 5.6 and 5.7 show the camera view from the ZED 2i and the Isaac Sim camera for the same log-loader configuration.



Figure 5.6: ZED 2i grapple camera view



Figure 5.7: Isaac Sim grapple camera view

5.3.2 Training Datasets

To train the three network branches, 289 RGB-D grapple images were collected from the FPInnovations log-loading test-bed and 7045 images were collected from the Isaac Sim simulation. The images contain the log-loader’s grapple facing the camera at various openings with zero to three grasped logs between its tongs. These images were used to form three training datasets as follows:

- **Dataset 1:** Contains the 289 real grapple RGB-D images, and was used to train the U-Net model that is to be employed on the real log-loader.
- **Dataset 2:** Contains 512 images that were randomly selected from the 7045 simulation RGB-D images. This dataset was used to train the U-Net model that is used to segment the simulated grapple and grasped logs.
- **Dataset 3:** Contains the 7045 RGB-D images from simulation, and was used to train both the grapple-opening and log-counting network branches. The dataset is divided as follows: 3895 images for 0 logs, and 1250, 950, and 950 images for 1, 2, and 3 logs, respectively.

The images from Dataset 1 and Dataset 2 were annotated by hand with masks indicating the pixels that correspond to the grapple, the logs that are grasped by the grapple (any non-grasped logs in the images were not considered), and the background. Figures 5.8 and 5.9 show two samples from the datasets with their hand-labeled masks.

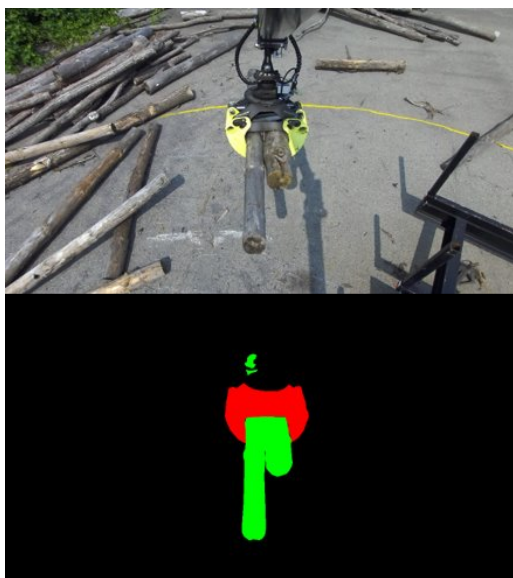


Figure 5.8: Dataset 1 sample

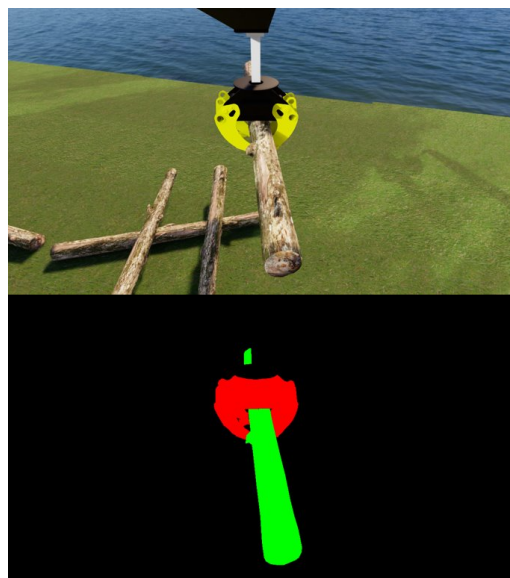


Figure 5.9: Dataset 2 sample

For Dataset 3, the grapple opening labels were automatically generated during data collection in Isaac Sim, and the log-counting labels were manually added to the dataset. Samples from this dataset are illustrated in Figure 5.10.



(a) Grapple opening of 43 cm
and logs count of 0



(b) Grapple opening of -42 cm
and logs count of 1

Figure 5.10: Isaac Sim grapple images dataset samples

5.3.3 Network Training

Each branch of our network was trained separately. This approach provides us with the flexibility of training each branch with different data and avoids the scaling and weighing issues that would have to be addressed if the losses for all three branches were summed and minimized for joint training of the branches.

Starting with our U-Net branch, the same procedure was followed to train the model for grapple segmentation in simulation and the model for the real log-loader: the corresponding grapple segmentation dataset (Dataset 1 for segmenting the real grapple, and Dataset 2 for the simulated one) was augmented 11 times with random rotations, zooms and crops (3179 images for the Dataset 1 and 5632 images for Dataset 2), before being split into a training set that contains 90% of the data and a validation set that contains the rest. The Adam optimizer was then used with a learning rate of 0.001 for both models of U-Net to minimize the pixel-wise cross-entropy loss:

$$CE_{segmentation} = -\frac{1}{NN_p} \sum_{n=1}^N \sum_{i=1}^{N_p} \sum_{c=1}^C y_{i,c} \log P(\hat{y}_{i,c}) \quad (5.1)$$

where N represents the number of training images, Np the total number of pixels in the input image, C the number of classes (grapple, grasped logs, and background), $y_{i,c}$ the true probability of pixel i belonging to class c (0 if it does or 1 if it doesn't), and $P(\hat{y}_{i,c})$ the predicted softmax probability of class c for pixel i .

Figures 5.11 and 5.12 show the training and validation losses for the U-Net model that was trained on the real grapple data, and that on the simulated grapple data respectively.

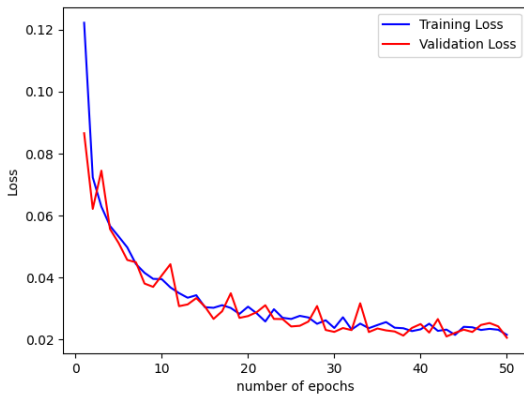


Figure 5.11: Real grapple U-Net training and validation losses

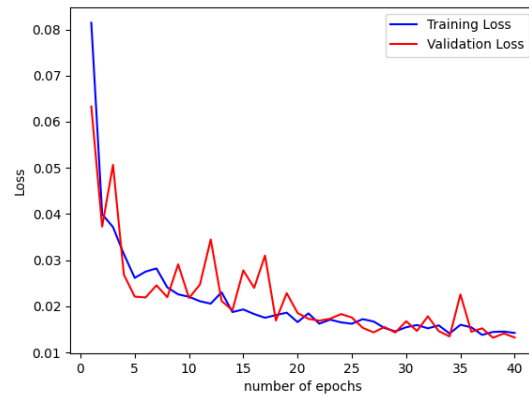


Figure 5.12: Simulated grapple U-Net training and validation losses

The final models were chosen from the epochs that had the minimum validation losses. These were epoch 43 for the real-grapple U-Net version and epoch 38 for the simulated grapple version. Figures 5.13 and 5.14 show sample outputs by the selected models.

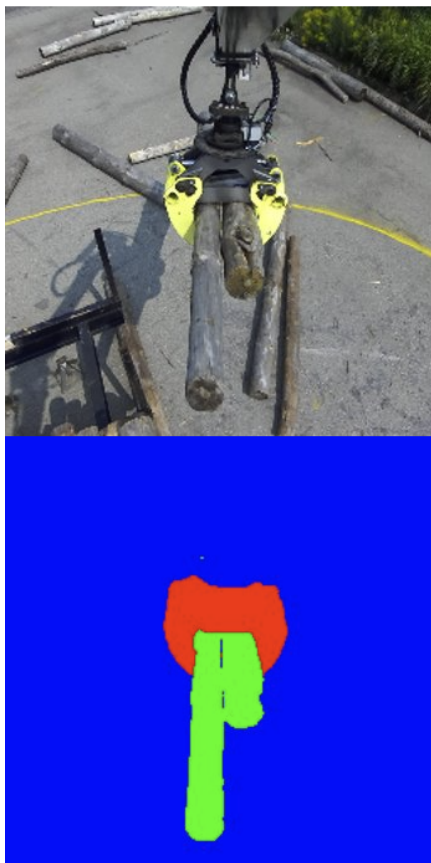


Figure 5.13: Real grapple segmentation sample

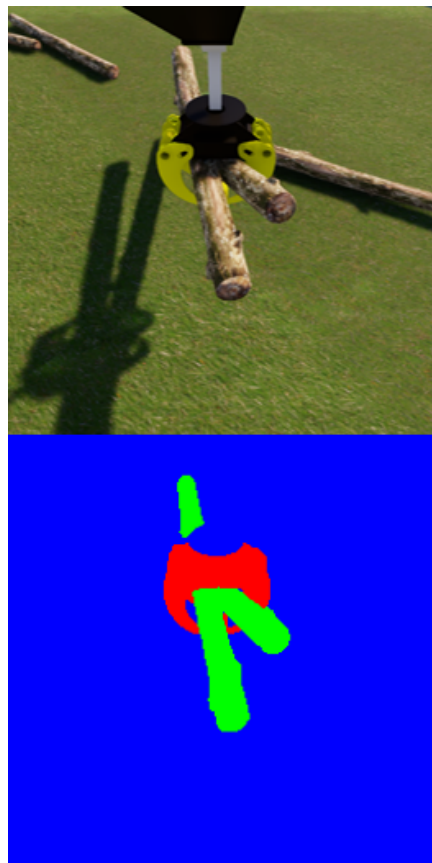


Figure 5.14: Simulated grapple segmentation sample

For the log counting branch, Dataset 3 (7045 simulated RGB-D images) was employed without any augmentation. Since data collection was done in simulation, we opted for collecting a large number of new images rather than augmenting a relatively small batch. The RGB-D images from this dataset were first passed to the final chosen U-Net segmentation model to generate segmentation masks. The log masks were intersected with the depth information from the RGB-D image to generate a set of depth images containing information

about the grasped logs only. These resulting images were then split such that 90% of them formed the training set and the remaining 10% formed the validation set. Since this branch's output is one of four classes (no logs, 1 log, 2 logs, 3 logs), training was done with the Adam optimizer and a learning rate of 0.001 to minimize the cross-entropy loss:

$$CE_{log-counting} = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C y_c \log P(\hat{y}_c) \quad (5.2)$$

where N represents the number of training images, C the number of classes ($C = 4$), y_c the true probability of the image belonging to class c (0 if it does or 1 if it does not), and $P(\hat{y}_c)$ the predicted softmax probability of belonging to class c .

Figure 5.15 presents the training and validation losses while Figure 5.16 shows the training and validation accuracies.

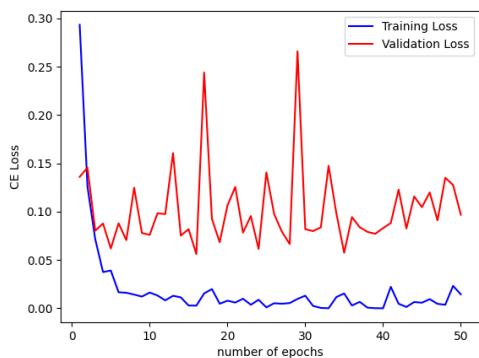


Figure 5.15: Log-counting branch training and validation losses

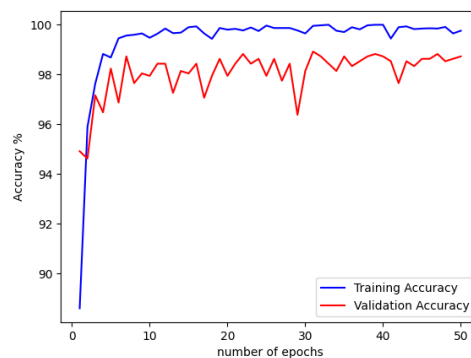


Figure 5.16: Log-counting branch training and validation accuracies

The final log-counting model was chosen based on a trade-off between validation loss and accuracy. The model from epoch 35 with a validation accuracy of 98.7% was selected.

Similarly for the grapple opening prediction branch, Dataset 3 was passed to our trained U-Net model to generate the segmentation masks. The grapple and the logs' masks were intersected with the depth information from the original images to receive a set of depth images that exclude the background. 90% of these images were used as a training set and the remaining 10% as a validation set. As for the grapple opening labels, their minimum value of -72.8 cm and maximum value of 146.2 cm were mapped to the range [0, 1]. Since the output of this branch is a continuous value, the Adam optimizer with a learning rate of 0.0005 was used to minimize the mean squared error loss:

$$MSE_{grapple-opening} = \frac{1}{N} \sum_{n=1}^N (y - \hat{y})^2 \quad (5.3)$$

where N represents the number of training images, y the ground truth grapple opening, and \hat{y} the predicted grapple opening.

Figure 5.17 presents the training and validation losses for this branch. The model from epoch 42 was chosen as our final grapple-opening model since it had the lowest validation loss, with a Root Mean Square Error (RMSE) of 0.033, which corresponds to 7.3 centimeters.

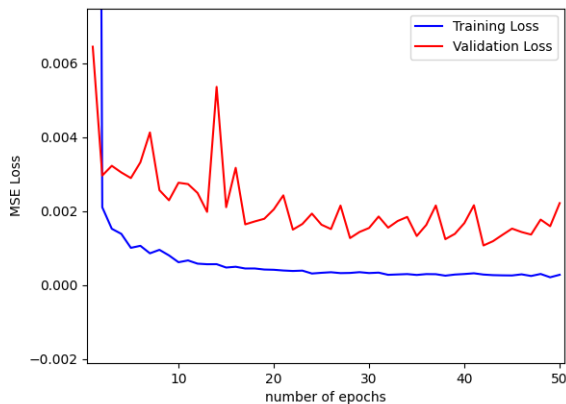


Figure 5.17: Grapple opening training and validation losses

5.4 Experimental Validation and Results

5.4.1 Simulated Results

To validate the developed architecture in our Isaac Sim log-loading simulation, the log-loader’s grapple was commanded to move to 600 random positions within the log-loader’s reach while maintaining the grapple at an angle facing the camera: 300 positions with 0 grasped logs and a random grapple opening, and 100 positions with each of 1, 2, and 3 randomly selected grasped logs. At every position, the predicted and true opening, and the predicted and true number of grasped logs were recorded. The RMSE and the Mean Absolute Error (MAE) for the opening predictions, along with the prediction accuracy rates for the log counting are provided in Table 5.1.

Number of Grasped Logs	Grapple Opening RMSE (cm)	Grapple Opening MAE (cm)	Log-counting Accuracy %
0	4.8	3.1	99
1	7.7	6.1	89
2	8.4	6.4	80
3	9.3	7.5	86

Table 5.1: Grapple opening and log-counting simulated results

As the table shows, the network is able to accurately predict the grapple’s opening when it is not carrying any logs, with an RMSE of 4.8 cm and an MAE of 3.1 cm. As expected, the opening prediction’s precision decreases as the number of grasped logs increases due to larger areas of the grapple being occluded (see Figure 5.13); the highest RMSE and MAE of 9.3 and 7.5 cm, respectively, were obtained when the grapple carried three logs. As for the log-counting, the results prove the ability of the network to count the number of grasped logs, with counting accuracies greater than 80%. Note that the performance of the network was adversely affected by the poor contact modelling in our Isaac Sim simulation which made the grasped logs wobble heavily, take awkward positions, and go through parts of the grapple on some occasions.

Since log-counting is an important factor in log inventory management, we further tested the log-counting network branch by employing it to count the number of collected logs during 5 autonomous log-loading cycles: for each cycle, multiple logs were placed in the working area of the log-loader and the grasp planning pipeline from Chapter 3 was used to perform

autonomous grasp attempts. However, for each attempt, right before unloading the logs in the trailer at the drop-off location, the number of collected logs was predicted by the log-counting network branch. These predictions were then summed up to obtain the predicted total number of collected logs for every cycle. The network was able to reasonably estimate the number of grasped logs, with a 4% error, and the obtained results for every cycle are shown in Table 5.2.

Cycle	True No. Logs	Counted No. Logs
1	9	10
2	12	13
3	10	10
4	9	9
5	10	10
Total	50	52

Table 5.2: Simulated log inventory counting results

5.4.2 Experimental Results

The final step of validating our developed architecture was to test its sim-to-real capabilities by deploying it on the FPInnovations log-loading test-bed. The U-Net model that was trained with Dataset 1 was used for grapple segmentation, and the grapple opening and log-counting networks that were trained in simulation were employed without any modifications. To provide ground-truth measurements for the grapple opening network branch, the grapple was instrumented with a draw-wire sensor. The sensor determines the opening of the grapple based on the displacement of a cable that extends and retracts with the hydraulic cylinder

that controls the grapple’s opening. The sensor was connected to an Nvidia Jetson Nano and the measured opening values were streamed through ROS over Wifi to our main computer.

Similarly to the simulated results’ procedure, the grapple was moved to 300 positions: 150 with no logs, and 50 for each of 1, 2, and 3 logs. The predicted and true opening, along with the predicted number of grasped logs were recorded at every position. The RMSE and MAE for the opening predictions, and the log-counting accuracy are provided in Table 5.3.

Number of Grasped Logs	Grapple Opening RMSE (cm)	Grapple Opening MAE (cm)	Log-counting Accuracy %
0	15.1	10.3	100
1	14.0	10.3	98
2	11.1	9.5	82
3	11.0	9.5	64

Table 5.3: Sim-to-real grapple opening and log-counting experimental results

As the table shows, the grapple opening network experienced an expected decline in performance when the trained model was transferred from simulation to reality. However, given the grapple opening’s range of ~ 220 cm, the obtained results remain reasonable, with the maximum RMSE and MAE being 15.1 and 10.3 cm, respectively, for the case of the grapple with no grasped logs. Some of the contributors to this drop in performance may be the swinging of the grapple that was not accounted for in the simulated training data, the noise in the ZED 2i camera’s images, minor errors in the CAD modelling of the grapple, and minor differences between the simulated and ZED 2i cameras’ perspectives. As for the log-counting, the network proved to be more successful in reality than in simulation for the

zero to two logs cases, even with our sim-to-real approach. This is most likely caused by the absence of the contact issues that are present in simulation, as were described in Section 5.4.1. As for the case of three grasped logs, the network provided successful predictions in only 64% of the cases due to one log being hidden under the other grasped two logs in many of the trials, and therefore not seen by the camera (see Figure 5.18). This problem did not occur at the same frequency in simulation because of the contact modelling issues: the logs could not maintain contact with each other and stabilized in positions where they were clearly distinguishable.



Figure 5.18: Grapple with three grasped logs, one of which is hidden

The sim-to-real performance of the log-counting branch was further evaluated on inventory management by counting the number of collected logs during 5 autonomous log-loading cycles, as was done in simulation. The results are reported in Table 5.4.

Cycle	True No. Logs	Counted No. Logs
1	12	13
2	10	9
3	11	10
4	8	8
5	9	7
Total	50	47

Table 5.4: Experimental log inventory counting results

The network was able to count the grasped logs with an error of 6%, with most errors being three logs detected as two because one of them was hidden from the camera's view. These results prove the validity of our architecture and its ability to provide predictions that are transferable from simulation to reality.

Chapter 6

Conclusion

To conclude the thesis, this chapter summarizes, in Section 6.1, the developed machine learning methods throughout this work, along with our most notable findings. Potential future directions to build up on our work towards log-loading autonomy are finally presented in Section 6.2.

6.1 Research Summary

In this thesis, we explored and evaluated the use of convolutional neural networks in forestry log-loading applications, to automate the grasping of cut-to-length logs. We started by adopting the grasp definition and architecture of the GG-CNNs from [19] for general grasp planning, and systematically modified the latter through multiple layer operations to obtain a new CNN architecture with enhanced performance. The performance of the three networks

was analytically evaluated through standard machine learning and grasp planning evaluation metrics, and through simulated and experimental robotic grasping trials on general object sets that we assembled. Our network outperformed its two counterparts throughout all the evaluation steps by achieving a higher accuracy rate on the Cornell general grasping dataset, and higher efficiency and grasping success rates during our experimentation.

Having our improved CNN architecture, we built a log-loading specific grasp planning pipeline around it. The computer vision log segmentation solution from [35] was used to detect logs in a log-loader’s scene, segment them and characterize them. The logs’ information was then passed to a clustering algorithm that selected a target logs cluster to grasp from. The chosen cluster was replicated in a virtual Gazebo environment, which allowed us to leverage the power of instantaneously placing a virtual depth camera at any desired location above the logs to capture the required depth images for grasp planning with our CNN.

The grasp planning pipeline was implemented on a simulated log-loader in Isaac Sim and on a real log-loading test-bed at FPInnovations. Grasping attempts were performed on log configurations ranging from a single log to a small pile of logs, and that are commonly seen in a forest log-loading scenario. Our pipeline proved to be successful at picking up logs, achieving a high grasping success rate and demonstrating the ability to plan grasps that target multiple logs at the same time. Furthermore, we examined the effect of the placement of the virtual camera on the grasp planning outcome, and showed that the quality of the

planned grasps can be significantly improved even with naive camera placement strategies like raising the camera upwards and changing its orientation.

Finally, we explored enhancing log-loading autonomy by enabling grapple opening width estimation and inventory management with computer vision and a multi-branch convolutional neural network. The grapple and the carried logs were segmented in the first branch using the U-Net semantic segmentation architecture. The segmentation masks were used to isolate the grapple and logs specific information in the input depth data, and the resulting depth images were fed to two convolutional branches that predicted the grapple's opening and counted the number of logs that it grasped. The network was validated in an Isaac Sim simulation that showed promising results when it comes to opening estimation and inventory management. For experimental validation, the grapple opening and log-counting branches were trained in simulation only and transferred directly on the FPinnovations log-loading test-bed. As expected with sim-to-real approaches, the network experienced a decline in performance, however, it still provided reasonable results to validate the possibility of determining the grapple's opening and detecting grasped logs by training a CNN in simulation, and without instrumenting the grapple.

6.2 Future Work

The work of this thesis is a stepping stone towards the automation of forestry log-loading with computer vision and convolutional neural networks. While our presented preliminary results prove the validity of our developed methods, there are several avenues for further exploration and improvement:

- Although this work demonstrates the possibility of training a grasp planning network on a general grasping dataset and using it for log-loading purposes, collecting a log-loading specific training dataset with the assistance of log-loader operators is a natural direction to explore. A log-loading specific dataset would contain embedded information in its labels on the strategies that operators use to decide on optimal grasping spots and potentially prevent the network from providing non intuitive grasps.
- Despite having experimentally validated Chapter 3's grasp planning pipeline on a log-loading test-bed, the performed experiments were not extensive or completely realistic: only 60 grasp attempts were performed on 12 configurations, the logs did not significantly vary in color, shape or length, the terrain where tests were conducted was relatively flat and did not contain any obstacles. Therefore, a larger experiment can be conducted in an environment that is more representative of the conditions that are encountered in the forest to accurately quantify the performance of the grasp planning pipeline.

-
- Since the input grasp planning data is captured by a virtual camera, optimal camera placement strategies can be developed to take advantage of the ability to instantaneously manipulate its placement with little resources. For instance, the camera can be intelligently made to collect multiple depth shots from several angles above the logs to provide more informed grasp predictions.
 - At their current stage, the training datasets from Chapter 5 are exclusive to the FPInnovations log-loading test-bed. The datasets can be expanded by adding images from multiple grapples to enable the network’s predictions to generalize to other log loaders. Furthermore, the grapple is always facing the camera in the current datasets’ images, and therefore, the grapple has to rotate to face the camera for the network to provide accurate predictions. Expanding the datasets with images of grapples at different orientations may allow the network to provide grapple opening and log-count predictions irrespective of the grapple’s orientation.
 - Since our current grapple opening and log-counting architecture requires RGB-D data for the segmentation branch, we had to collect RGB-D data of both the simulated and the real grapple. One can investigate methods to perform segmentation based solely on depth data. This would allow for the sim-to-real transfer of the full network from simulation to the real log-loader without any data collection from the real grapple.

Bibliography

- [1] E. Ayoub, P. Levesque, and I. Sharf, “Grasp planning with cnn for log-loading forestry machine,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11802–11808, IEEE, 2023.
- [2] F. Huq and I. Branch, “Skills shortages in canada’s forest sector,” *Canadian Forest Service*, 2007.
- [3] S. Kollarova, *Innovation and Advanced Technology Use in the Canadian Forest Sector*. PhD thesis, Université d’Ottawa/University of Ottawa, 2014.
- [4] E. Yousefi, D. P. Losey, and I. Sharf, “Assisting operators of articulated machinery with optimal planning and goal inference,” in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 2832–2838, IEEE, 2022.
- [5] S. Westerberg and A. Shiriaev, “Virtual environment-based teleoperation of forestry machines: Designing future interaction methods,” *Journal of Human-Robot Interaction*, vol. 2, no. 3, pp. 84–110, 2013.

-
- [6] D. Ortiz Morales, S. Westerberg, P. X. La Hera, U. Mettin, L. Freidovich, and A. S. Shiriaev, “Increasing the level of automation in the forestry logging process with crane trajectory planning and control,” *Journal of Field Robotics*, vol. 31, no. 3, pp. 343–363, 2014.
- [7] J. Andersson, K. Bodin, D. Lindmark, M. Servin, and E. Wallin, “Reinforcement learning control of a forestry crane manipulator,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2121–2126, IEEE, 2021.
- [8] H. Gietler, C. Böhm, S. Ainetter, C. Schöffmann, F. Fraundorfer, S. Weiss, and H. Zangl, “Forestry crane automation using learning-based visual grasping point prediction,” in *2022 IEEE Sensors Applications Symposium (SAS)*, pp. 1–6, IEEE, 2022.
- [9] S. Weiss, S. Ainetter, F. Arneitz, D. A. Perez, R. Dhakate, F. Fraundorfer, H. Gietler, W. Gubensäk, M. M. D. R. Ferreira, C. Stetco, *et al.*, “Automated log ordering through robotic grasper,”
- [10] S. Caldera, A. Rassau, and D. Chai, “Review of deep learning methods in robotic grasp detection,” *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.
- [11] H. Duan, P. Wang, Y. Huang, G. Xu, W. Wei, and X. Shen, “Robotics dexterous grasping: The methods based on point cloud and deep learning,” *Frontiers in Neurorobotics*, vol. 15, p. 73, 2021.

-
- [12] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
- [13] S. Ainetter and F. Fraundorfer, “End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13452–13458, IEEE, 2021.
- [14] W. Hu, C. Wang, F. Liu, X. Peng, P. Sun, and J. Tan, “A grasps-generation-and-selection convolutional neural network for a digital twin of intelligent robotic grasping,” *Robotics and Computer-Integrated Manufacturing*, vol. 77, p. 102371, 2022.
- [15] L. Chen, P. Huang, Y. Li, and Z. Meng, “Edge-dependent efficient grasp rectangle search in robotic grasp detection,” *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 2922–2931, 2020.
- [16] S. Kumra, S. Joshi, and F. Sahin, “Antipodal robotic grasping using generative residual convolutional neural network,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9626–9633, IEEE, 2020.
- [17] Y. Yu, Z. Cao, Z. Liu, W. Geng, J. Yu, and W. Zhang, “A two-stream cnn with simultaneous detection and segmentation for robotic grasping,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

-
- [18] A. Depierre, E. Dellandréa, and L. Chen, “Jacquard: A large scale dataset for robotic grasp detection. in 2018 ieee,” in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3511–3516.
- [19] D. Morrison, P. Corke, and J. Leitner, “Learning robust, real-time, reactive robotic grasping,” *The International journal of robotics research*, vol. 39, no. 2-3, pp. 183–201, 2020.
- [20] S. Ainetter, C. Böhm, R. Dhakate, S. Weiss, and F. Fraundorfer, “Depth-aware object segmentation and grasp detection for robotic picking tasks,” *arXiv preprint arXiv:2111.11114*, 2021.
- [21] F.-J. Chu, R. Xu, and P. A. Vela, “Real-world multiobject, multigrasp detection,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [23] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, “Scalable deep reinforcement learning for vision-based robotic manipulation,” in *Conference on Robot Learning*, pp. 651–673, PMLR, 2018.

-
- [24] S. Joshi, S. Kumra, and F. Sahin, “Robotic grasping using deep reinforcement learning. arxiv,” 2020.
- [25] P. Egli and M. Hutter, “Towards rl-based hydraulic excavator automation,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2692–2697, IEEE, 2020.
- [26] P. Egli and M. Hutter, “A general approach for the automation of hydraulic excavator arms using reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5679–5686, 2022.
- [27] P. Egli, D. Gaschen, S. Kerscher, D. Jud, and M. Hutter, “Soil-adaptive excavation using reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9778–9785, 2022.
- [28] Q. Lu, Y. Zhu, and L. Zhang, “Excavation reinforcement learning using geometric representation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4472–4479, 2022.
- [29] R. Dhakate, C. Brommer, C. Bohm, H. Gietler, S. Weiss, and J. Steinbrener, “Autonomous control of redundant hydraulic manipulator using reinforcement learning with action feedback,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7036–7043, IEEE, 2022.

-
- [30] J. Andersson, “Predicting gripability heatmaps using conditional gans,” 2022.
- [31] L.-M. Faller, C. Stetco, and H. Zangl, “Design of a novel gripper system with 3d- and inkjet-printed multimodal sensors for automated grasping of a forestry robot,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5620–5627, IEEE, 2019.
- [32] S. Fodor, C. Vázquez, and L. Freidovich, “Automation of slewing motions for forestry cranes,” in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, pp. 796–801, IEEE, 2015.
- [33] S. Fodor, L. Freidovich, and C. Vazquez, “Practical trajectory designs for semi-automation of forestry cranes,” in *Proceedings of ISR 2016: 47th International Symposium on Robotics*, pp. 1–8, VDE, 2016.
- [34] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [35] J.-M. Fortin, O. Gamache, V. Grondin, F. Pomerleau, and P. Giguère, “Instance segmentation for autonomous log grasping in forestry operations,” *arXiv preprint arXiv:2203.01902*, 2022.

- [36] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, “Masked-attention mask transformer for universal image segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1290–1299, 2022.
- [37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
- [38] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
- [39] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.
- [40] B. Baheti, S. Innani, S. Gajre, and S. Talbar, “Eff-unet: A novel architecture for semantic segmentation in unstructured environment,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 358–359, 2020.
- [41] J. Jing, Z. Wang, M. Rättsch, and H. Zhang, “Mobile-unet: An efficient convolutional neural network for fabric defect detection,” *Textile Research Journal*, vol. 92, no. 1-2, pp. 30–42, 2022.

-
- [42] F. Liu and L. Wang, “Unet-based model for crack detection integrating visual explanations,” *Construction and Building Materials*, vol. 322, p. 126265, 2022.
- [43] L. Wang, R. Li, C. Zhang, S. Fang, C. Duan, X. Meng, and P. M. Atkinson, “Unetformer: A unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 190, pp. 196–214, 2022.
- [44] X. He, Y. Zhou, J. Zhao, D. Zhang, R. Yao, and Y. Xue, “Swin transformer embedding unet for remote sensing image semantic segmentation,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.