Fusion of Estimated Depth and RGB Features for Improved Grasp-Type Selection of Novel Objects

by

Ali Asghar

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

Abstract

Prostheses can alleviate some of the challenges faced by upper limb amputees in performing activities of daily living. However, electric-powered prosthetic hands have not seen much improvement over the past decade. Unintuitive interfaces for selecting grasp types have resulted in low user satisfaction and high abandonment rates. Recently, efforts have been made to automate the grasp type selection process by collecting visual data, such as Red, Green Blue (RGB) images or depth data of the object to be grasped and classifying the object into the desired grasp type. This effort has been greatly aided by the advent of Deep Convolutional Neural Networks (DCNNs), which have been trained on examples of objects and their desired grasp types. However, the biggest challenge is to improve the generalization capabilities of DCNN models, so that they can efficiently classify novel objects, i.e., objects that the model was not trained on.

Combining RGB and depth data has been shown to improve model generalization; however, common methods of acquiring depth data require bulky hardware, that cannot be installed on a prosthetic hand for practical applications. Therefore, this research focused on estimating depth maps through pre-trained models developed using RGB input images acquired from a single compact RGB camera, instead of depth maps acquired from bulky dedicated hardware such as an RGB-D camera or similar hardware.

An object detector based DCNN architecture was used to detect grasp types of objects along with their bounding boxes in cluttered scenes. To combine the RGB and estimated depth data, this research used a novel method to fuse RGB feature maps and estimated depth feature maps. In order to train the DCNN, a dataset was created with images of objects in a cluttered scene from the viewpoint of a camera mounted on a prosthesis. Every graspable object in each image was annotated with a bounding box and assigned one of two grasp types: neutral wrist palmar or pronated wrist precision. Different methods of encoding single-channel depth maps as three-channel depth data were evaluated, including duplication, surface normal encoding, jet colormap encoding, and HHA encoding. Moreover, different strategies to fuse RGB feature maps with estimated depth feature maps were also evaluated.

Experiments determined that the developed model was capable of operating in real-world scenarios, such as in cluttered scenes with multiple graspable objects. Compared to training the model on only RGB data, there was an increase of up to 12.7% in common metrics used to evaluate the model's generalization capabilities when the RGB data was fused with estimated depth data. This is the first work that demonstrated improvement in DCNN model performance for the task of detecting grasp types of novel objects by using fusion of RGB and estimated depth-map features. The improvement in performance of the model that was trained using estimated depth data exceeds that of methods that require dedicated hardware to acquire depth data by up to 2.8%. The proposed model can be incorporated in the control schemes of upper limb prostheses, without the need for dedicated hardware.

Acknowledgements

Special thanks to Dr. Jonathan Kofman for the continued support, guidance, and feedback he has provided as my thesis supervisor. His kindness and care made it possible to navigate through tough times brought upon by COVID-19 and personal health challenges.

I would like to thank my family, without whose support I would not be here.

I would also like to extend my gratitude to my lab mates Dr. Scott Pardoel, Gaurav Shalin, Lukas Stracovsky and Nathan Ng for sharing their views and brainstorming ideas with me.

Dedication

To my grandmother, Rashida, whose prayers set me up for success.

To my mom and dad, Fatima and Hussain, who sacrificed their dreams countless times so I could realize mine.

To my beautiful and loving wife, Alefiya, who has cared for me boundlessly.

I could not have made it without any of you.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

Upper limb amputations, either congenital or acquired, limit the functionality of amputees and their ability to interact with their environment. Most manmade objects that are encountered in everyday life have been designed to be used by a functional hand, which means that individuals with a lack of this functionality face many challenges in their activities of daily living and work. Restoration of the lost functionality is, therefore, important to alleviate some of the challenges faced by amputees.

Prosthetic devices available to amputees can be divided into three major categories: passive, body-powered, and electric-powered. Passive prostheses mainly aim to reproduce the aesthetics of the lost hand, focusing primarily on the static cosmetic appeal of the device. However, task-specific attachments, for example, a screwdriver attachment at the end of the prosthesis, are also available [1], [2] and help in providing task-specific functionality for a passive prosthesis. Body-powered prostheses use a body harness and cables that the amputee can pull on using their upper-body muscles, to allow them to open and close an end effector. While durable and reliable, body-powered prostheses that use a body harness and cables are not very cosmetically appealing and provide limited functionality.

Electric-powered anthropomorphic hands, i.e., those that mimic the human hand, are both cosmetically appealing as well as functional. They include electronic sensors and electric actuators and controllers to control the motion of the prosthetic hand. However, even though there have been rapid advancements in the fields of robotics and soft computing, the task of controlling electric-powered prosthetic hands remains challenging. Prosthetic hands must not only meet weight and size constraints but also provide an intuitive interface for the amputee to control them. Recent surveys [3] continue to show that most amputees prefer body-powered devices over electric-powered devices, mainly because electric-powered prosthetic devices are complex to operate.

To simplify the control strategy of electric-powered prosthetic devices and make them more intuitive for the amputee to operate, it is convenient to control the motion of the prosthetic device as a whole rather than controlling the individual digits. Studies have shown that up to 80% of all activities of daily living can be performed by as few as six grasp types [4]. Based on the GRASP taxonomy [37], the six most frequent grasp types used in activities of daily living can be identified as: medium wrap, index finger extension, power sphere, lateral, precision disk, and tip pinch. A grasp type is a unique combination of finger joint angles that allows a person to immobilize an object and perform the desired task with the given object. This implies that for a prosthetic hand to be functional, the need to control individual degrees of freedom can be abstracted away and replaced by the ability to select the most appropriate grasp type, given the object to be manipulated [5]. Such abstraction allows the hand to be underactuated and still be fully capable of performing the intended function, while making the control strategy much simpler [6]. As an example of underactuation, consider the three degrees of freedom (DoF) offered by the metacarpophalangeal (MCP), proximal interphalangeal (PIP) and distal interphalangeal (DIP) joints that each flex and extend a phalanx in the finger (Figure 1.1). These joints can be controlled in a prosthetic device using a single actuator that couples the motion of the individual phalanges, using a tendon or four-bar mechanism that flexes and extends all three DoF of the finger simultaneously (Figure 1.2).

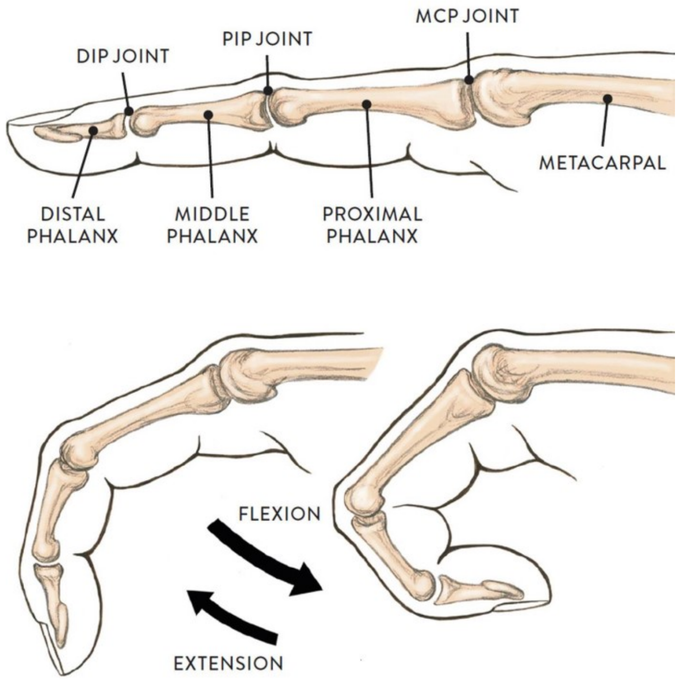


Figure 1.1: The distal interphalangeal (DIP), proximal interphalangeal (PIP) and the metacarpophalangeal (MCP) joints each have 1 degree of freedom (flexion and extension) [7]

Current commercially available prosthetic devices do indeed simplify the control strategy by allowing the user to select a grasp type, instead of trying to individually control the digits [8]–[11].

However, the process of selecting the appropriate grasp type is nontrivial and often results in the user having to use their functional hand to select the grasp type, for example, by requiring the pressing of a button or using a smartphone application [12]. This increases the cognitive load on the user, as they now must decide which of the available grasp types would be the most appropriate, given the object they would like to grasp [13]. Moreover, the use of a functional hand to control the prosthetic hand is paradoxical. The functional hand used to control the prosthetic hand could instead be used to accomplish the intended task much more efficiently and quickly. The increased cognitive load and the paradox just described lead to the abandonment of most electric-powered prosthetic hands, as the devices fail to meet the user's expectations with regards to functionality [3], [14], reducing the purpose of such devices to a decorative showpiece rather than a functional hand that can be relied upon for efficient and quick prehension (i.e., to form a stable grasp with the desired object).

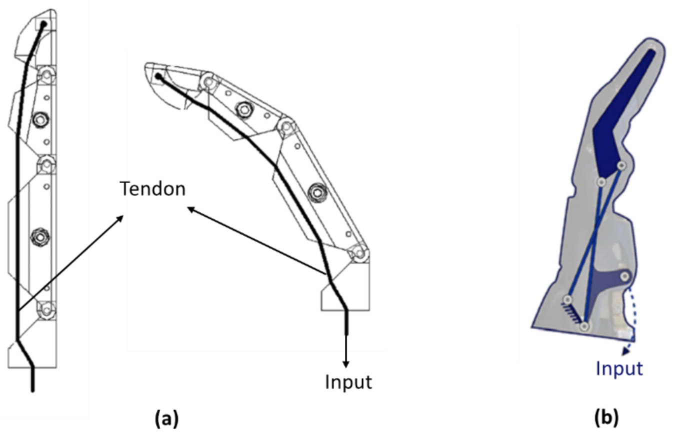


Figure 1.2: Underactuated finger mechanisms that require only a single input to flex all three finger joints using: (a) tendons (adapted from [15]) and (b) a four bar mechanism (adapted from [16]).

To improve the adoption rate of electric-powered prosthetic devices, the control strategy needs to not only incorporate grasp type selection, but also decrease the cognitive load on the user and ensure that the functional hand is not required during the grasp-selection process. In other words, the grasp selection process must be automated. Therefore, the task of intuitively controlling an electric-powered anthropomorphic prosthetic hand can thus be restated as automating the process of selecting the desired grasp type.

To solve the problem of automating grasp type selection, two main approaches have been used. One approach aims to connect the electric-powered prosthetic hand to the user's central nervous system by measuring the depolarizations and hyperpolarizations (increases and decreases in voltage, respectively) that occur on the sarcolemma (muscle fiber membrane) of the muscles on the user's residual limb by using a technique called surface electromyography (sEMG) [17], [18]. The depolarizations and hyperpolarizations are necessary, and precede, the contraction of a muscle, resulting in a small electrical signal. These electrical signals, also known as myoelectric signals, can be measured using specialized electrodes, known as sEMG sensors, which can be placed on a user's skin on top of a residual limb muscle. When the user wishes to select a grasp type, they simply contract their residual limb muscles in a specific pattern. By using multiple sEMG sensors around the user's residual limb and applying pattern recognition techniques, the myoelectric signals can be classified into a grasp type that the amputee intends to select [19].

The other main approach to automatic grasp type selection is to enable the prosthetic device to make intelligent decisions on its own. This has been achieved by augmenting the prosthesis with Red-Green-Blue (RGB) or grayscale intensity cameras and/or depth sensors, which can either be mounted on the prosthesis [20] or the amputee's body, such as on their head [21]. These sensors are used to collect visual data of the target object (the object to be grasped), which is then used to select the most appropriate grasp type for the given object [20]–[33]. Studies have shown that fusing depth data with RGB or grayscale data improves the prosthesis' ability to select the correct grasp type [25], [27]. The approach described above is often referred to as computer-vision based methods for automatically selecting grasp types due to the use of visual data of the object.

Once the grasp type has been selected, the rest of the control strategy is trivial. For example, after the grasp type has been selected, the device can automatically pre-shape the hand to the selected grasp type, indicating to the user that the device is ready to grasp the object [8]. The user then needs to simply position their hand appropriately around the object, and instruct the hand to close, for example, by contracting a residual-limb muscle that has an sEMG sensor on top of it [8]. It is important to distinguish between the use of sEMG sensors to instruct the hand to close, versus classifying the acquired myoelectric signals into a grasp type using pattern recognition techniques. In the former case, the sEMG sensor acts as an on/off switch, while in the latter case, the sEMG sensors are used to acquire data that is then processed for grasp-type selection.

Due to several practical limitations which have been summarized in Section 2.4, neither sEMG-based pattern recognition nor computer-vision based solutions have been successfully commercialized. In this thesis, arguments to support computer-vision based methods as the superior solution to the automatic grasp-type selection problem are presented along with a novel method to improve its practical applicability. In particular, noting that depth data improves computer-vision based techniques, this thesis proposes a Deep Convolutional Neural Network (DCNN) model with novel strategies for fusing Red-Green Blue (RGB) image intensity data with depth data of the target object to automatically select the grasp type. However, it is unfeasible to mount bulky depth sensors on prosthetic devices to acquire depth data, due to size and weight constraints. Therefore, this thesis proposes that the depth data be estimated through RGB data using a pre-trained neural network rather than using bulky depth sensors. This thesis also evaluates several methods for encoding single-channel estimated depth maps as three-channel depth data, which have been claimed in the literature to improve performance of neural networks that have been pre-trained on RGB data [34]–[36]. Furthermore, this thesis evaluates the hypothesis that incorporating estimated depth data can improve the generalizing abilities of DCNN models, and thus improve their performance on novel objects (i.e., objects that were not shown to the model during model training). The ability to accurately classify novel objects is critical, as the model cannot be trained on every single object the user might encounter. Therefore, this work evaluates model performance on novel objects with and without the integration of the estimated depth data. To the best of the author's knowledge, this is the first study that evaluates the impact of using estimated depth data on the performance and generalization abilities of DCNN models for the task of automatically selecting grasp types.

1.2 Objectives

The primary aim of this thesis is to determine whether the fusion of RGB data with estimated depth data can improve the generalization capabilities of vision-based grasp-type detection models. The primary aim can be realized through the following objectives:

1. Develop a new dataset that can be used to train and evaluate DCNN models for the task of grasp-type detection on novel objects in real world scenarios.
2. Develop a grasp-type detection DCNN model that can fuse RGB data with estimated depth data using different strategies.
3. Evaluate different strategies to fuse RGB data with estimated depth data. Determine which strategy is optimal for grasp-type detection model performance on novel objects.

4. Determine which method of encoding estimated depth maps gives optimal grasp-type detection model performance on novel objects.
5. Determine whether fusing estimated depth data with RGB data improves grasp-type detection model performance on novel objects. If so, determine the most optimal combination of fusion strategy and encoding method.

1.3 Thesis Outline

This thesis is organized as follows: Chapter 2 provides an overview of the current trends in grasp-type selection and discusses their limitations. Chapter 3 explains the methodology for developing and training grasp type selection DCNNs including different strategies to fuse RGB and estimated depth data as well as different methods of encoding depth data. Chapter 4 presents and analyses the results of the experiments that were conducted. Chapter 5 concludes the thesis and provides remarks on future work.

Chapter 2
Background and Literature Review

2.1 Control Strategies based on Manual Grasp Type Selection

The market for electric-powered prosthetic hand devices has seen very little innovation over the past decade, with the underlying technology for controlling the devices mostly remaining unchanged. Current commercially available devices require the manual selection of grasp types using various methods. The most popular method for manual grasp type selection is referred to as Direct Electromyographic (direct-EMG) Control, wherein sEMG sensors on the user's residual limb are used as switches to allow the user to manually select between different grasp types that the device is preconfigured for [8], [9], [11]. Other methods for manual selection of grasp type include pushbuttons on the device that can be used to toggle between different grasp types [9] or smartphone applications that can be used to select the desired grasp type [12]. Manual grasp type selection makes operating devices unintuitive and, at times, nearly impossible without a fully functional second hand. In this section, some of the most popular commercially available prosthetic devices that utilize manual grasp type selection are reviewed to examine their control strategies and discuss their drawbacks.

The functionality offered by commercially available devices can be quantified by the number of grasp types they offer. It is noted here that most companies tend to market their devices as being capable of a large number of grasp types; however, that number is inflated by introducing slight variations to the same grasp type. For a fair comparison, the GRASP Taxonomy [37] is used to determine the number of grasp types offered by these devices. Therefore, only grasp types that are uniquely different based on the GRASP taxonomy are considered in this comparison. Table 2.1 outlines the grasp types available with each device with the respective grasp types displayed in Figure 2.1. The grasp types offered by commercially available devices may include, but are not limited to, the six most frequent grasp types that occur in activities of daily living [4]. The devices reviewed in this section can be seen in Figure 2.2.

2.1.1 Hero Arm

A common control strategy among commercially available electric prosthetic devices is direct-EMG, which allows the user to manually select the grasp type and choose when to execute the grasp using sEMG sensors. The affordable Hero Arm [8] available from Open Bionics employs direct-EMG and claims to be capable of up to six different grasp types grouped into pairs, with only four unique ones based on the GRASP taxonomy.

Table 2.1: Grasp types offered by commercially available prosthetic hands based on the GRASP Taxonomy [37].

Grasp Type / Device Name	Large diameter	Fixed hook	Tripod	Tip pinch	Lateral	Index Finger extension	Precision disk	Adduction grip	Medium Wrap	Power Sphere	Extension type
Hero Hand	✓	✓	✓	✓							
Bebionic Hand	✓	✓	✓	✓	✓	✓	✓	✓			
Luke Arm	✓			✓	✓				✓	✓	
iLimb Hand			✓	✓		✓					✓

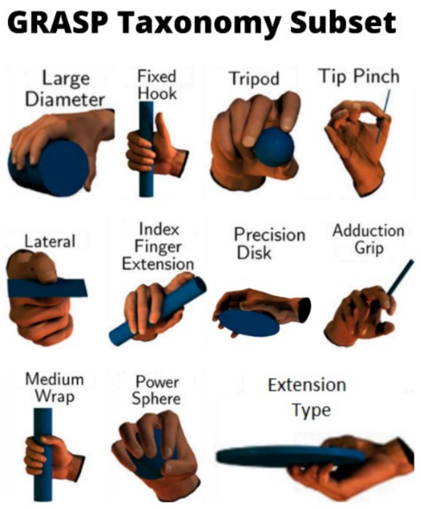


Figure 2.1: A subset of the GRASP Taxonomy with the common grasp types offered by commercially available electric-powered prosthetic hands. (Adapted from [37]).

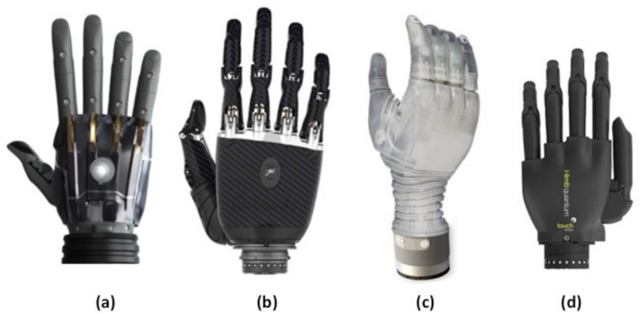


Figure 2.2: Commercially available prosthetic hands with manual grasp selection control strategies. (a) Hero Arm (b) BeBionic (c) Luke Arm (d) iLimb Quantum.

The Hero arm uses an sEMG sensor positioned on top of the wrist extensor muscle along with a push button on the back of the hand to manually select grasp types. When the sEMG sensor detects myoelectric signals from the wrist extensor muscle (which are produced when the user contracts their wrist extensor muscle) for longer than one second, it switches to the other grasp type in the same group (each group has two grasp types). By pressing and holding the push button on the back of the hand, the user is able to toggle between three grasp type groups, each offering two grasp types that the user can toggle between by contracting their wrist extensor muscle.

The Hero arm also provides manual actuation of the wrist and thumb. The interphalangeal joint of the thumb can be manually adjusted into either the open, middle or closed positions while the wrist can be pronated and supinated over a range of 180 degrees using a locking mechanism (Figure 2.3). Once the appropriate grasp type is selected, the user can close the hand by contracting their wrist flexor muscle. The contraction of the wrist flexor muscle is detected by a second sEMG sensor positioned on top of the muscle.

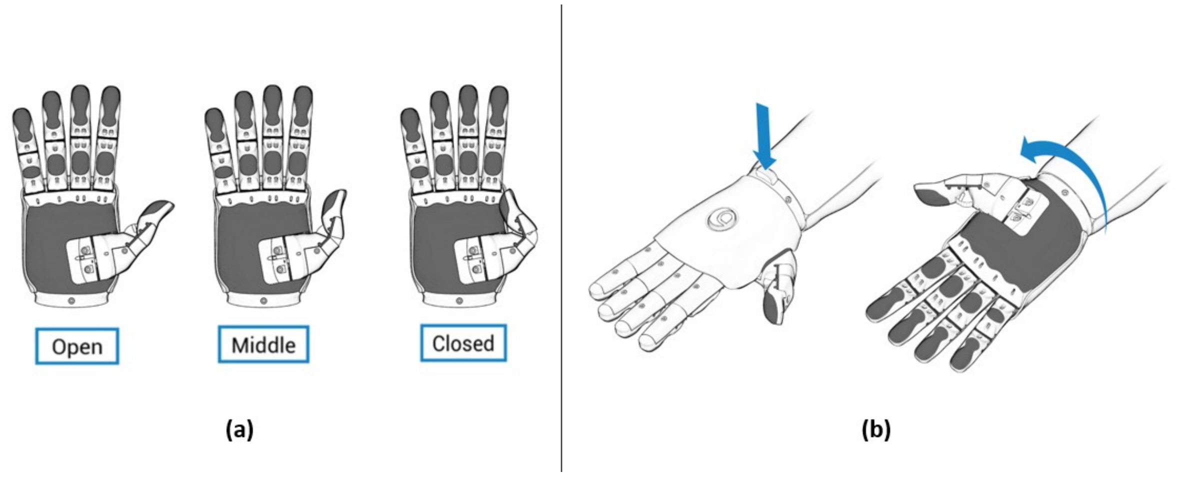


Figure 2.3: Hero Arm manual actuation of: (a) the thumb into open, middle and clossed positions, and (b) the wrist at any desired angle using the locking mechanism [8].

2.1.2 Bebionic Hand

The Bebionic prosthetic hand [9] from Ottobock claims to provide 14 different grip patterns with 8 of these being unique based on the GRASP taxonomy. The Bebionic also employs a direct-EMG based control strategy using a combination of sEMG sensors and a physical button on the back of the hand to switch between grasp types. However, only a maximum of 8 out of the 14 claimed grasp types are available at any given time. Unlike the Hero Arm, the flexion and extension of the thumb are actively actuated, while the thumb opposition is passive and can be configured in either the opposed or non-opposed positions. The active thumb actuation provides some level of autonomy; however, the manual grasp selection based control strategy still adds a significant cognitive load on the user.

2.1.3 Luke Arm

The Luke Arm [10] from Mobius Bionics also comes with sEMG sensors and buttons for manual grasp selection using direct-EMG based control strategies. Additionally, users can choose to connect several other input devices on their Luke arm, that can be preconfigured to control the prosthetic device. The input devices offered by Mobius Bionics include Inertial Measurement Units (IMU), pressure transducers, pressure switches, rocker switches, and linear transducers. The wide range of input devices gives the user the ability to select the one that is the most comfortable and intuitive based on their personal preferences and level of amputation. Depending on the mode of input chosen by the user, the Luke Arm can be preconfigured to manually select up to 6 different grasp types, 5 of which are identified as being unique based on the GRASP taxonomy.

2.1.4 i-Limb Hand

The i-Limb [11] hands from Ossur claim to offer up to 24 different gripping patterns but these can be grouped into only 4 different grasp types based on the GRASP taxonomy since most of them are only minor variations of one another. The gripping patterns available with the i-Limb family of hands can be configured to be selected in a wide variety of ways, including muscle control, smartphone app, gesture control, and Bluetooth-based proximity control. Muscle control is what Ossur calls its direct-EMG control strategy; the smartphone app has one-tap icons that can be programmed to select one of the grasps. In gesture control mode, each one of the hand’s forward, backward, left, or right gestures can be programmed to select a specific grasp type. While simple to implement, the above manual grasp selection methods imposes a significant cognitive load on the user.

The i-Limb hands also offer a control strategy that mimics automatic grasp type selection. In *proximity mode*, the hand can be programmed to select a specific grasp type when it is fully opened near a preprogrammed Bluetooth chip. Objects can be pre-tagged with these Bluetooth chips, allowing the hand to automatically select a preprogrammed grasp type when the hand gets close to one of the tagged objects. This strategy imposes a relatively low cognitive load on the user; however, the Bluetooth chips can be counterproductive in cluttered scenes where different objects may require different grasp types. This is because the hand will automatically select the preprogrammed grasp type when near the chip, regardless of the object the user desires to grasp. Moreover, multiple Bluetooth chips cannot be positioned near each other, as they may cause interference.

2.1.5 Limitations of Manual Grasp-Type Selection Based Strategies

The need to use a fully functional hand, for example, to push a button or operate a smartphone application, in order to toggle to the appropriate grasp type and the time delay between needing to grasp an object and achieving the desired grasp type makes the process tedious and frustrating. Although devices like the Hero Arm and BeBionic hand reduce the need to use a functional hand by allowing the user to select some of the grasp types using sEMG sensors on their residual limb, the user still needs to figure out which of the available grasp types will be the most suitable for a given object. In most cases, several hours of training and practice are required for users to get to a skill level where they can successfully incorporate manual grasp-type-selection based devices into their daily lives. Moreover, it has been reported that even experienced users face difficulty in utilizing the full range of device capabilities [38]. While devices such as the Luke Arm provide a wide range of input devices to select from, ultimately, these input devices are only different ways to toggle between grasp types. Therefore, intuitive control strategies that can automate the grasp selection process would significantly reduce the complexity required to operate electric-powered prosthetic devices and help reduce their high rate of abandonment.

2.2 EMG for Automatic Grasp Type Selection

2.2.1 EMG based Pattern Recognition

One approach aimed at reducing the user's cognitive burden is to automatically select the grasp type by analyzing the patterns in the myoelectric signals generated by the muscles of the residual limb when the user contracts the muscles so that the user need not manually select the grasp type [39]. Most muscles responsible for flexing and extending the wrist and fingers are present in the forearm. For user's with transradial amputation or wrist disarticulation, the activation pattern of their residual forearm muscle contains encoded information about the type of grasp the user wants to execute. When the user intends to select a particular grasp type, their brain sends signals to the residual forearm muscles resulting in contractions of a specific combination of muscles. These contractions result in the generation of myoelectric signals, that can be measured by sEMG sensors, which are positioned all around the residual limb [40]. Analysing the patterns in the myoelectric signals from the residual forearm muscles allows for the decoding of the information about the intended grasp type and provides an intuitive way to form an interface between the prosthetic hand and the user's central nervous system.

The myoelectric signals are generated as a result of asynchronous activation of hundreds of groups of muscle fibres with the number of groups and frequency of firing related directly to the force produced by the muscle [37]. However, the instantaneous raw values obtained are not very useful and resemble random noise, with signal to noise ratios typically less than 10, even in the best case [37]. Therefore, meaningful features must first be extracted from the raw measured myoelectric signals that preserve class separability between the different grasp types. In general, time-domain features are considered to be robust and generate feature spaces with relatively high inter-class separations, while requiring little computational resources [38], [39]. These features can then be used to train a classifier to learn correspondences between the features and the grasp types. Comparative studies agree that as long as the extracted features are descriptive enough, most classifiers achieve relatively similar performance [40]. The outputs of the classifiers can then be used to determine which actuators to activate in the prosthetic device [41], [42] to achieve the desired grasp type.

2.2.2 Limitations of EMG based Pattern Recognition Systems

Several limitations have prevented automatic grasp selection using pattern recognition techniques on myoelectric signals from being successful in clinical trials. Most of the EMG data collection for feature extraction and classification is done in a static environment, with the amputees' arms resting on a surface, allowing them to have greater control over targeted muscle activations. However, in real-life scenarios such as while walking or doing chores, the arm is non-static. This mobility introduces errors in the pattern recognition pipeline [41]. Moreover, users may use varying muscle forces throughout the day to accomplish the same task. A different force corresponds to a different myoelectric signal pattern, causing an increase in the classification error to a point where the classifier is unusable (up to 20% error) [19]. Furthermore, during normal operation, it is common for the sEMG sensors to get displaced. This displacement is also common while donning the prosthetic device, resulting in the sensors being in a slightly different position over the residual limb every time the user dons their device. Classifiers trained based on sensors placed at a certain position on the residual limb have been shown to generate significantly higher error rates when the position of these sensors was altered [42]. Build up of sweat between the sensor and the skin also introduces noise in the sensor readings, resulting in increased classification errors [41]. The problem of sEMG sensor displacement and sweat buildup can be addressed by using intramuscular EMG (iEMG) sensors that are surgically inserted directly into the target muscles. However, based on surveys, users are less likely to undergo invasive procedures for inserting iEMG sensors [43]. Furthermore, the problem of non-static muscle activity and variations in muscle forces still prevents the pattern recognition system from being accurate even with iEMG sensors.

EMG-based pattern recognition approaches have further limitations related to the need for customization, based on the amputation level of every user. Ideal sites for EMG signal acquisition must be determined for each individual by a professional prosthetist. Then, training signals must be acquired from these sites, followed by feature extraction and classifier training. More data may be required based on classifier performance, and the entire process can be frustrating for the amputee. Furthermore, EMG-based systems lose their purpose of being intuitive for transhumeral and above levels of amputations, since these amputees have lost the muscles involved in controlling the hand. This requirement for EMG-based systems to be highly customized and their dependence on the user's level of amputation results in poor generalizability of the pattern recognition system and has hindered its adoption in commercial devices.

2.3 Computer Vision for Automatic Grasp Type Selection

Another way that researchers have approached the task of automating grasp-type selection is to collect visual data of the object that will be grasped, instead of collecting the user's myoelectric signals [20], [21], [29], [23]–[25], [27], [30], [31], [44]–[46]. For example, an RGB camera mounted on the prosthetic device can collect Red-Green-Blue per-pixel color intensity data of the target object, i.e., the object that the user desires to grasp. The data can then be processed by a computer to classify the object into the most appropriate grasp type. The classifier can be a traditional computer-vision algorithm that uses handcrafted geometrical features of the target object and rule based reasoning [20], [21], [29], or, it can be a machine learning model such as a DCNN that learns abstract features [23]–[25], [27], [30], [31], [44]–[46]. However, both proposed approaches share the same aim: to extract meaningful features from the target object that can help to automatically identify the most appropriate grasp type from a given set of grasp types. Furthermore, replacing the EMG sensors with a camera for automating the grasp selection process eliminates the problems that are intrinsic to the EMG-based pattern recognition strategies, discussed in Section 2.2. Moreover, the classifier does not need to adapt to every user, since it extracts features from the object, not the user. Using a classifier that is independent of the user may reduce the time needed by users to get skillful in operating their devices, which may lead to greater user satisfaction and potentially, lower abandonment rates.

Several methods of incorporating visual data of the object in the control strategy to automate grasp selection have been developed and are reviewed below. RGB cameras are not the only way to acquire visual data of the object. Gray-scale cameras that acquire light intensity data or time-of-flight sensors that acquire depth data have also been used [25], [28]. Devices reviewed in this section that incorporate visual sensors to classify objects into the desired grasp types may also use other sensors to execute different phases of the control strategy. The control strategies reviewed in this section are divided into five distinct phases: 1) detection of the user's intention to grasp an object, 2) selection of the target object (from the various objects in the scene), 3) selection of the appropriate grasp type based on features extracted from visual data of the target object, 4) pre-shaping the device for the selected grasp, and 5) execution of the grasp. Not every phase of the control strategy requires visual data of the object, nor does every device explicitly implement each phase. However, each phase is presented separately to evaluate its role in the overall control strategy. At the end of the section, some limitations of the current vision-based automatic grasp-selection methods are discussed.

2.3.1 Detection of user intention to grasp

Knowledge of when the user intends to grasp an object using their prosthetic device has been used as an event-based trigger for state transitions within devices. State transitions primarily result in the device entering the next phase in their control strategy, such as to acquire visual data of the object [33]. To detect that the user intends to perform a grasp, different strategies have been employed by researchers. For example, the user can inform the device of their intent to grasp an object by triggering an EMG sensor [20], [30], [31], [33], [44], [47]. In this case, the EMG sensor is not used to automatically select the grasp type, but to aid in state transitions within the system. No pattern recognition is applied to the EMG signals and the sensor simply behaves as a switch with a pre-set threshold on the intensity of the signal. Using EMG sensors as a switch, therefore, does not have the drawbacks of the EMG-based pattern recognition systems. However, requiring the user to manually inform the device of the user's intention to grasp makes the process semi-autonomous, as the user is required to make a conscious effort. To fully automate the process, intracranial EEG (iEEG) sensors have been explored to automatically detect the user's intention to grasp by studying the activity of the brain [21]. However, iEEG sensors must be surgically inserted, and most users are hesitant in undergoing such an invasive procedure [43].

2.3.2 Target object selection

In real world scenarios, the object that the user intends to grasp may be situated in a cluttered scene. Therefore, the vision based automatic grasp type selection system must first identify which one of the objects in the scene a grasp type needs to be selected for. In other words, the prosthetic device must first select the target object, which is the object the user intends to grasp, before the object's visual data can be processed for grasp-type selection.

Some researchers have oversimplified the problem of target-object selection by assuming that there will be only one object in the scene with one grasp type, the target object [30], [46]–[48]. However, this assumption does not generalize well in real-world scenarios, as there may be multiple graspable objects with different grasp types present in the scene. Therefore, to enable a vision-based system to perform in cluttered scenes with multiple objects, several methods have been proposed to select the target object. If the vision sensor is mounted on the prosthetic device, then it is reasonable to assume that the object closest to the center of the image is the target object [20], [24], [33]. The duration the camera views an object, or the device-object distance has also been incorporated in selecting the target object more accurately [22]. However, determining the distance of an object from the device using just a monocular camera is not trivial. One method of estimating the object distances from the device in a cluttered scene using a monocular camera is to use the area of the bounding box around the objects [22]. However, such a method would require area-to-depth correspondences for the objects to be known *apriori* [22]. A different approach to determine device-object distances is to use a dedicated sensor, such as an ultrasonic sensor, that measures the distance between the device and the target object [20], [28]. Some methods propose the use of a separate system to determine the target object. For example, eye-tracking systems have been used by researchers to determine the target object based on the user's gaze [21], [27].

2.3.3 Grasp selection

In the grasp selection phase of the vision-based control strategy, researchers process visual data of the target object to classify it into the most suitable grasp type from a set of grasp types that the prosthetic device can execute. The visual data is typically collected by a sensor (e.g., an RGB camera) mounted on the prosthetic device [20] or on some part of the user's body (such as their head) [21]. Several approaches to select grasp type using visual data of the target object have been proposed. One approach is to make rule-based decisions by extracting handcrafted geometrical features from visual data of the target object [20], [21], [29]. For example, by representing the object as an ellipse, features such as the ellipse's area [29] or its major and minor axis [20] were calculated. Other features have also been used, such as the radius of a sphere or cylinder that encapsulates the target object, which was calculated by incorporating depth information [21]. The calculated features were then compared to pre-set thresholds and were classified into a grasp type using if-then rules.

With the recent influence that deep learning has had on the field of computer vision, Convolutional Neural Networks (CNNs) have gained popularity as a powerful method for extracting learned features rather than using handcrafting geometric properties from images of the target object [48]. The basic architecture of a CNN is depicted in Figure 2.4. The first layer consists of several filters that are convolved over the input, which is the target object's visual data (e.g., RGB or depth data), to generate feature maps. In later layers, more filters are convolved over the feature maps and the process is repeated as many times as necessary. In some CNN architectures, the outputs of the convolutional layers are passed through an activation function such as a ReLU (Rectified Linear Unit) [49] to introduce non-linearities. Other CNN architectures may choose to pass the feature maps through a dimensionality reduction layer, such as a MaxPool layer [50], whereby only the maximum value over the area of the filter is considered. In the final layers, the feature maps are passed to a classifier, which is most often a multi-layered perceptron (MLP) [51]. The classifier takes the feature maps as an input and outputs the predicted grasp types. The filters used to generate feature maps are not known *apriori*. Instead, the values that define the filter (known as its parameters) are initialized as random values [50] and then learned through an optimization policy such as Stochastic Gradient Descent (SGD) [52]. The optimization policy is used to minimize the error between the predicted and ground truth grasp types by inputting the target object's visual data and iteratively updating the parameters of the filters. The error is calculated using a loss function, which is designed to measure the dissimilarity between the predicted and expected output of the model. The process of updating the filter parameters by showing examples of the target object's visual data and ground truth grasp type to the CNN model is known as model training. The resulting filter parameters after training are known as learned parameters (as the model "learns" the parameters by "seeing" examples) and the feature maps generated by learned parameters are colloquially referred to as learned features. Figure 2.4 depicts only a basic architecture of a CNN, and several modifications have been proposed by the deep learning community over the years, such as ResNet [53].

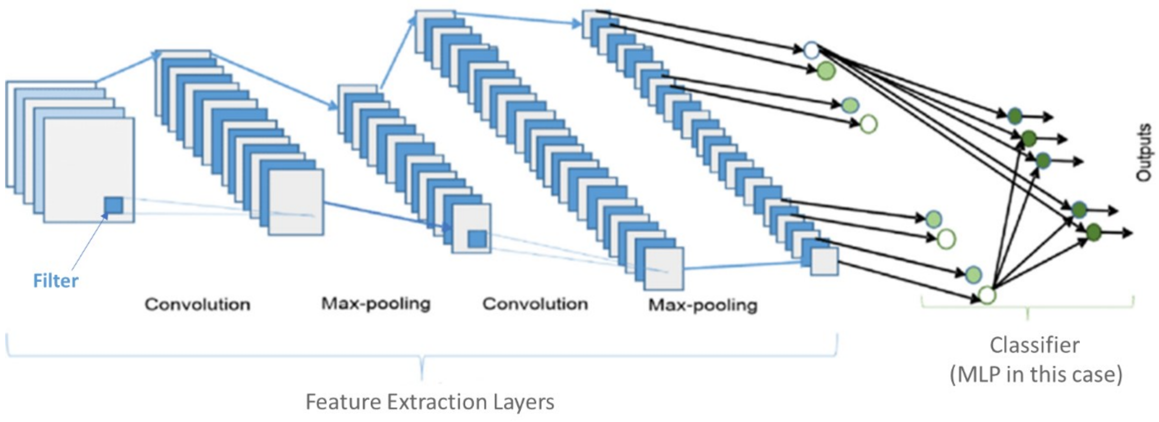


Figure 2.4: A simple CNN architecture with convolutional layers for feature extraction, MaxPool layers for dimensionality reduction and a Multi-Layered Perceptron (MLP) as a classifier. (Adapted from [50]).

For the task of automatic grasp type selection, several CNN models have been trained on examples of visual data and the corresponding desired grasp type of objects. Considerable work has been done on collecting visual data of objects labelled with the desired grasp type and designing CNN architectures that can accurately and efficiently classify objects with the correct grasp type [23], [24], [27], [30], [44], [45], [54]. Instead of training custom models initialized with random parameters, pre-trained models (such as Inception v3 [55]) have also been used to extract features, which can then be used to classify the grasp type of the target object [31]. Moreover, instead of simply using RGB data as in conventional CNNs, integrating depth data into the CNN has been shown to improve the accuracy of classifying the grasp type [25], [47]. However, collecting depth data traditionally requires dedicated hardware (i.e. depth sensor), which in practice, is difficult to mount on a prosthetic hand due to its strict size and weight constraints. In one work, instead of analyzing individual images of the scene independently, a time series analysis was performed on a stream of images using deep learning techniques [47]. To achieve this, a CNN was used to extract features from the image stream, which were then used as an input to a Long Short-Term Memory (LSTM) network [56] for classification. As opposed to MLPs being used for classifying features, LSTM models enable a neural network to learn to classify the grasp types of objects by incorporating features from the current as well as previous images from a continuous stream of images [47].

2.3.4 Pre-shaping of Prosthetic Hand

After the correct grasp type is identified, the prosthetic device is actuated to assume the pre-grasp shape either autonomously [20], [21] or through user input, for example, by triggering an EMG sensor [30], [33]. If it is known that the user intends to grasp an object (by detecting user intention to grasp), then pre-shaping of the device can be done autonomously, which allows for lower latency of the overall control strategy. However, some researchers found it useful to let the user override the autonomous system in case the intent detection system gives a false detection, or the automatically selected grasp is not appropriate for the task [33].

2.3.5 Grasp Execution

The final phase of a vision-based control strategy is to form a stable grasp with the target object. After the device is pre-shaped with the selected grasp type, it is positioned around the target object and is closed until it comes in contact with the object and forms a stable grasp. This is either done autonomously [21] by the prosthetic device, or the user takes control and decides when to start and stop the actuation of the device [22], [29], [33], [54].

In cases where the grasp is executed autonomously, the control system determines when the device is at the correct grasping pose (at the right distance from the object and in the correct orientation) to begin grasp execution by actuating the device [21]. The control system then determines when a stable grasp has been established with the object in order to stop actuating the device [21]. To start and stop device actuation, the device is equipped with the capability to determine when the device is in the correct pose and when a stable grasp has been achieved, respectively [21]. Determining device pose and grasp stability requires additional sensors and feedback loops, increasing the complexity of the overall control strategy.

Instead of autonomously actuating the device, some researchers allowed the user to decide when to start and stop device actuation [22], [29], [33], [54], which is easier to implement and does not increase complexity of the control strategy. Moreover, allowing the user to control device actuation provides for a more pleasant user experience. Users conveyed their intent to start or stop the actuation of the prosthetic device by means of sensors, such as sEMG sensors [22], [33], [54] or mechanomyography (MMG) sensors [29], which were positioned on their residual limb. As opposed to sEMG sensors that measure myoelectric signals, MMG sensors measure mechanical vibrations in muscles when they are contracted. In either case, the sensors were used as on-off switches with a pre-set threshold. After the users positioned the device around the object, they contracted their muscle that was designated for cueing device actuation. If the contraction generated a signal higher than the pre-set threshold, then the device started actuating. The user then visually inspected if a stable grasp had been made with the object, upon which they contracted the designated muscle again for stopping device actuation. As before, if the signal recorded by the sensor was greater than some pre-set threshold, then the device stopped actuating, which marked the end of the grasp execution phase.

2.3.6 Limitations of Computer Vision-based systems

The major drawback of current computer vision-based grasp-type selection techniques, discussed in Section 2.3, is their lack of applicability in real-world scenarios. Traditional methods based on handcrafting geometrical features of the target object have only been shown to work under highly constrained environments [20], [21], [29]. Methods that train CNNs for learning target object features used training examples positioned in structured environments, such as a plain background [25], [30], [45], [46]. CNN models that have been trained on data from structured environments have a high bias in unstructured real-world scenarios [47]. Moreover, the CNN models were tested on data similar to what the models were trained on [30], [51], such as using different view points of the same object for training and testing [30], [46]. Using similar training and testing data does not permit determination of the generalization capabilities of the proposed technique. A generalized model is able to provide comparable performance on objects it has seen during training, as well as novel objects (objects that were not shown during training). Generalization is an important property for CNN based methods, as it is impossible to train a model on every single object the prosthetic device user might encounter. Furthermore, many CNN based methods treated the grasp-type selection task as an image-classification task, grossly oversimplifying the problem [25], [30], [44], [46]. The image classification problem takes an RGB image as an input, and outputs one value: the class that the object in the image belongs to. Therefore, it can only identify one object per image [48]. However, it is not uncommon for the target object to be in a cluttered scene with multiple other graspable objects.

Methods that incorporate depth data to improve model performance do so by acquiring depth data from dedicated hardware (depth sensor) that cannot be practically mounted on a prosthetic device intended for everyday use [25], [47]. The practical limitations are mainly due to the size, weight and power consumption constraints of a prosthetic hand device, which must all be satisfied to lower user abandonment rates.

2.4 Summary

In Section 2.1, prosthetic devices that implement manual grasp selection-based control strategies were reviewed. Manual grasp selection refers to the need for the user to first decide which grasp type is required and then inform the device about their decision using an input device such as a push-button. Such control strategies induce significant cognitive loads on the user and have resulted in high abandonment rates [3].

In Section 2.2, control strategies that automatically select the grasp type by analyzing the myoelectric signals on the user's residual limb were discussed. Note that the user still needs to decide the most suitable grasp type, while the device automatically detects the user's intentions. These methods face several practical challenges, such as the sEMG sensor being displaced [42], sweat interfering with the signal [41], poor classifier performance in unconstrained environments [19], [41], and the need to adapt to the requirements of individual users.

As an alternative to acquiring data from the user, Section 2.3 explored methods of automating grasp-type selection that acquire visual data of the target object. Researchers have used computer-vision techniques to extract either handcrafted [20], [21], [29] or learned features (such as using CNNs) from visual data of the target object. Visual data included RGB data (colorized images), image intensity data (grayscale images) or depth data. The extracted features were used to classify the target object into its desired grasp type. Researchers that employed computer vision techniques for automatic grasp type selection also augmented their control strategies with other sensors (such as sEMG [25] and ultrasonic sensors [20]) to determine the user's intention to grasp, select the target object, and execute the selected grasp type.

CNNs were found to be the preferred way of extracting learned features from visual data of the target object [23], [24], [27], [30], [44], [45], [54]. While CNNs are a powerful tool, previous work has limited their generalization capability by not properly formulating the problem and training models under constrained environments [25], [30], [45], [46]. In other words, previous CNN-based grasp type selection methods are not capable of delivering acceptable performance on novel objects and in environments that were not shown to the model during training. Moreover, by treating the selection of grasp type as an image classification problem [25], [30], [44], [46], the CNN models would not perform well if the object were in a cluttered scene. Incorporating depth data has been shown to improve the generalization capabilities of CNN models for the task of automatic grasp type selection [25], [47]. However, in previous research, the depth data were acquired by dedicated hardware that would not be practical to install on a prosthetic hand due to size, weight, and power consumption constraints.

In light of the limitations of existing computer-vision based strategies for automatic grasp-type selection, a detailed rationale for the methods of this thesis is provided in Section 3.1.

Chapter 3
Methods for Fusing RGB and Estimated Depth data for Automatic Grasp-Type Selection

3.1 Rationale

To address the current limitations of computer-vision based grasp-type selection methods, this thesis treats grasp-type selection as an object-detection problem. In particular, a CNN is used as an object detector, identifying every object in a cluttered scene using a rectangular bounding box, and furthermore, simultaneously classifying the object into a grasp type, from a set of possible grasp types that the prosthetic hand is capable of executing. For this work, it is assumed that the input images are acquired from a RGB camera mounted on a prosthetic hand, and the prosthetic hand is capable of executing two grasp types: the neutral wrist palmer and the pronated wrist precision grasps. The bounding boxes of objects in the scene can be used to simultaneously detect the user's intention to grasp and determine the target object. For example, in the case where the camera is mounted on the prosthetic hand, the object whose bounding box is closest to the center can be selected as the target object and the area of the target object's bounding box can be used to determine the target object's distance from the prosthetic hand. Intent to grasp can be determined if the prosthetic hand is closer to the target object than some predetermined threshold. Therefore, treating the automatic grasp type selection problem as an object detection problem eliminates the need for separate sensors and strategies to implement the user intention detection, target object selection, and grasp type selection phases. Moreover, in this thesis, depth data is estimated from RGB data using a pre-trained monocular depth estimation CNN model, to eliminate the need for dedicated depth sensors. Furthermore, improvements in the CNN model's ability to detect the grasp types of novel objects in unconstrained environments by fusing depth data with RGB data are evaluated, to determine whether estimated depth data provides similar gains in the model's generalization capabilities as depth data acquired through dedicated hardware (depth sensor). The methods used and decisions made toward achieving these objectives are discussed in the remaining sections of this chapter.

3.2 Grasp Type Dataset

In order to train a CNN model to be able to detect the grasp type of novel objects, the model must be provided with examples of RGB and estimated depth data of objects in cluttered scenes with a rectangular bounding box around each object. Each bounding box must also be labelled with a grasp type most suited for the object it represents. The examples must be diverse in terms of object geometric properties, surface texture, colour, size, and reflective properties, so that the trained model has strong discriminative abilities while maintaining a high degree of generalizability. The model should be able to perform well on the training examples without overfitting to them, such that when a novel object is presented during testing, the model is able to make accurate predictions of the object's bounding box and grasp type.

Deep architectures can improve the performance of CNNs [48]. However, a deep architecture requires more layers, which increase the number of parameters that are needed to be trained, and thus the need for more training examples. To meet the high demand of data for Deep CNNs (DCNNs), ImageNet [57], MS COCO [58] and other datasets have been created. The former is a collection of over 1.2 million labelled RGB images belonging to 1,000 different object classes, while the latter has over 200 thousand labelled RGB images belonging to 80 object classes.

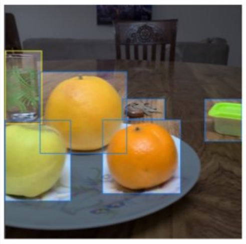
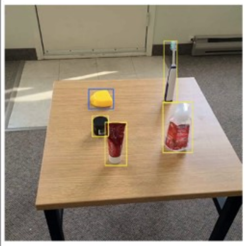
For this study, the training examples must be labelled with the preferred grasp type, not the object class. This means that the existing image datasets present a limitation if used in this study. While it is possible to filter the available datasets for objects that can be classified into one of the selected grasp types on which the DCNN is to be trained, these filtered images may be cluttered with other objects that may affect the discriminative abilities of the newly developed models in this thesis. If not all the objects in an image are annotated during training, the DCNN might learn to ignore them as background. Therefore, if images of the objects that were not annotated at train time are presented to the DCNN model at test time, it is highly likely that the model will ignore those objects as being part of the background, resulting in a high number of false negatives.

Therefore, to minimize the sources of bias in the training pipeline and to precisely evaluate the role of depth information in detecting grasp type, a new dataset of images was created for this research, where all objects in every image are labelled as belonging to one of two grasp types. The remainder of Section 3.1 elaborates on how the dataset was created and the reasoning behind the decisions made.

3.2.1 Image Collection

While training a DCNN, it is important to ensure that the examples seen by the model during training are similar to those expected during model deployment. Therefore, RGB data was collected with the assumption that the new model will have to perform inference on RGB data acquired from a camera mounted on a prosthetic arm or hand. Moreover, the objects were placed in cluttered scenes, where a single RGB image may contain multiple graspable objects, as expected in real-world scenarios.

Table 3.1: Individual scenes from which images were acquired to create the dataset used for training, validating, and testing the DCNNs. Example frames show the bounding boxes around objects.

Scene	Scene 1	Scene 2	Scene 3	Scene 4
Description	Home dining table	Home study table	Home kitchen counter	Home 2 dining table
Number of objects	7	6	8	8
Number of images	2318	1340	1046	1077
Example frame				
Scene	Scene 5	Scene 6	Scene 7	Scene 8
Description	Home 2 study table	Common household items	Bathroom items	Lab table
Number of objects	7	8	5	4
Number of images	315	298	204	131
Example frame				

A smartphone camera was used to record RGB videos in a variety of scenes that represent real-life situations, such as working at a study table, eating at a dining table, meal preparation on a kitchen counter. An able-bodied person held a smartphone camera in their hand and moved the camera to approach the objects and create a view similar to the expected view of a camera mounted on a prosthetic hand preceding a grasping task. No depth data was collected as the depth data was estimated from the RGB data.

RGB data was sampled from the acquired videos at varying sampling rates. The optimal sampling rate depended on individual scenes as special care was given to minimize the correlation between samples, although it is acknowledged that some correlation would exist due to the nature of the collection method. A higher sampling rate would result in a larger dataset and hence more training examples. However, the training examples would be highly correlated, and not provide extra information to the model. Training the model on correlated examples is like training on the same example multiple times, which can be achieved by increasing the number of epochs (number of passes of the entire training dataset) the model is trained for. A sampling rate of approximately 6-10 frames per second (fps) was used.

In total, 6,729 samples of RGB data were collected from the videos, which included instances of 53 unique objects. Each sample had a resolution of 1440x1440 pixels. Table 3.1 summarizes the different scenes that were recorded.

3.2.2 Image Annotation

The primary objective of this research is to evaluate the effectiveness of incorporating estimated depth information in the prediction of the grasp type of novel objects. Therefore, the dataset is not created to have the maximum possible number of grasp types, but instead, only has two different grasp types: the neutral wrist palmar grasp and the pronated wrist precision grasp. These two grasp types were chosen as they correspond to the two most frequently used grasp types for activities around the house [4], which is where the data was collected.

For each of the grasp types used this thesis, the objects were chosen to incorporate a wide variety of geometric and surface properties (e.g., size, shape, color, texture, reflectiveness), that would challenge the model to learn generic instead of object-specific features. This should enable the DCNN model to accurately predict the grasp type of even those objects that it was not shown during training, as long as the novel object has those generic features that the DCNN has learnt to associate with the respective grasp type. The ability to predict the grasp type for unseen objects is extremely important for real-world applications, as it is impossible to train a model for every single object that the prosthetic device user will encounter. Testing to measure the ability of CNN models to make grasp type predictions on novel objects is also absent in most studies found in the literature [22], [24], [25], [27], [30], [46] and would help in determining the true advantage of incorporating depth data for model training and inference. Figure 3.1 shows the two grasp types used and example objects from the dataset that was created.

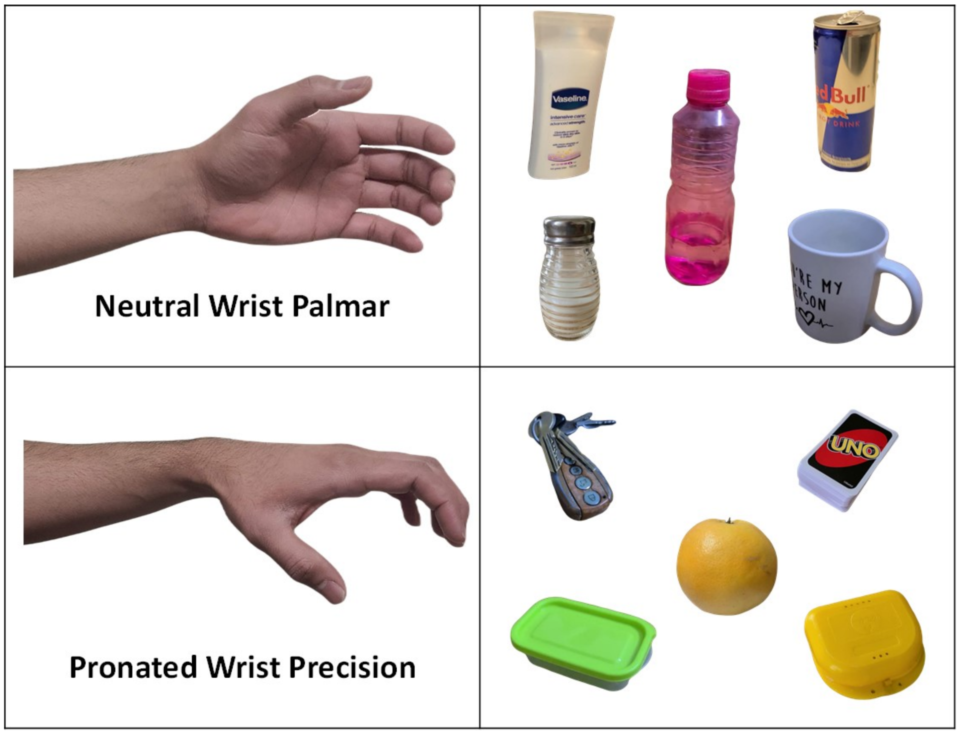


Figure 3.1: Neutral Wrist Palmar and Pronated Wrist Precision grasp types with example objects from the dataset corresponding to the respective grasp types.

For each image sampled from a scene, every object in the image was assigned a bounding box. First, the x and y pixel coordinates of the center of the object relative to the top left corner of the image were recorded. Next, the width and height of a rectangle encapsulating the object were measured in pixels. Finally, the values were normalized between 0 and 1 using the following equations:

$$x_i = \frac{X_i}{image\ width} \qquad (Eq.\ 3.1)$$

$$y_i = \frac{Y_i}{image\ height} \qquad (Eq.\ 3.2)$$

$$w_i = \frac{bounding\ box\ width}{image\ width} \qquad (Eq.\ 3.3)$$

$$h_i = \frac{bounding\ box\ height}{image\ height} \qquad (Eq.\ 3.4)$$

where X_i and Y_i are the x and y pixel coordinates of the center of the object i relative to the top left corner of the image, x_i, and y_i are the normalized bounding box coordinates and w_i and h_i are the normalized bounding box width and height, respectively, for object i. The image width and image height were both 1440 pixels for all the images in the dataset.

Each bounding box was also assigned a class label, either the neutral wrist palmar grasp or the pronated wrist precision grasp. The annotations were performed by two male engineering graduates (ages 25 and 28). The annotators were provided with a list of all the objects that could be seen in the videos along with their respective grasp types. The annotations were cross-checked by the author of this thesis and only a few errors needed rectification, such as loose bounding boxes i.e., instances where unnecessary background pixels were included inside the bounding boxes. Examples of the rectangular bounding boxes were drawn on a few example frames and are displayed in Table 3.1. The dataset consists of over 18,000 bounding boxes, with class labels approximately equally divided between the two grasp types.

3.2.3 Train-Validation-Test Split

To evaluate the DCNN models on novel objects, it was ensured that the samples from scenes used for testing were not shown to the model during training. For example, if samples from Scene 1 were used for training, then no samples from this scene were used for testing. This forces the model to make detections on objects it has not seen during training, which in turn are placed in unique backgrounds dissimilar from the ones used for training, further challenging the model's generalization abilities.

The samples used during training were split into training and validation sets. The training set samples are used for loss calculation and backpropagation for parameter updating, while the validation set is used to evaluate the model's performance at the end of each epoch. Although the training and validation sets contained samples from the same scenes, there was no duplication of samples in the two sets. Validating the model on inputs similar to the training data and then testing the model on novel data provides a quantitative measure of the model's generalization abilities.

The experiments performed for this study use approximately 4.6 K images in the training set, 1.2 K images in the validation set, and 1K images in the test set.

3.2.4 Data Augmentation

It can be challenging for a DCNN to learn generic high-level features that would be necessary to make predictions on novel objects without overfitting on the training data from just 4.6 K images, especially with a high correlation between the examples shown to the model during training. To overcome the shortfall in the number of training images and their high correlation, several data augmentation techniques were employed that have been known to reduce training time and avoid overfitting [59]. The steps are shown in Figure 3.2 and described below.

Firstly, the mosaic data augmentation algorithm [60] is used to synthesize training examples by mixing four different samples together. This is done by forming a 2x2 grid of randomly sized cells and populating them with slices of four different training images. For each RGB image in the training dataset, slices of the original image and three others randomly chosen images are placed inside the randomly sized grid cells. Additionally, the mosaics are translated and scaled. Furthermore, the Hue-Saturation-Value (HSV) of the mosaics were modified to introduce variation in colour. Finally, the mosaics were flipped horizontally with a probability of 0.5.

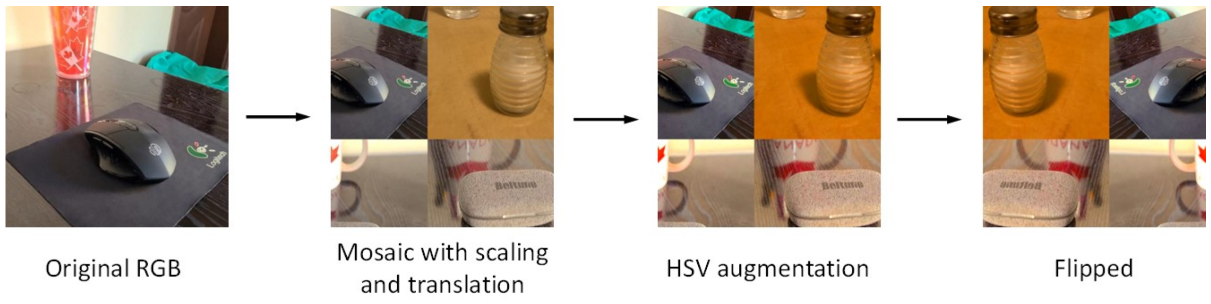


Figure 3.2: Augmentation process used for training: (from left to right) the original RGB image is turned into a 2×2 mosaic using three other random images from the training dataset with randomly sized slicing, scaling, and translation. Next, the HSV values are randomly modified. Finally, the image is flipped horizontally with a probability of 0.5.

The described augmentation techniques were employed with the aim of allowing the model to learn a wider variety of surface properties by ensuring that no two training samples are the same, even across different epochs. This is because the sizes of the slices are random and the other three slices are from random images chosen during each iteration at train time. Moreover, slicing the images results in the objects being randomly cropped, which allows the model to perform well even when the full image of an object is not available.

During validation and testing, there is no need to update the model parameters or address overfitting and loss convergence. Instead, the learned parameters need to be evaluated on real-world examples. Therefore, no data augmentation algorithms are applied to the validation and test datasets.

3.3 Grasp-Type Detection Model

Since the dataset described in Section 3.2 consists of two grasp types, the chosen model architecture must be able to differentiate between three classes (two grasp types and the background). Moreover, the model must also be able to predict the location of the bounding boxes in the image. It is possible to perform these tasks by extracting engineered features from the image and using multiple binary classifiers and regression models, such as SVMs. Note that 3 binary classifiers would be needed for the three-class classification and 4 regression models to predict the bounding boxes. However, DCNNs were chosen in this thesis for three main reasons: (1) DCNNs are able to learn task specific abstract features that can outperform engineered features, (2) a single DCNN can be used to predict both the grasp type and bounding boxes simultaneously, and (3) it is anticipated that DCNNs will be better at extending to a larger number of grasp types.

DCNNs have been shown to be good at being able to transfer their knowledge across domains [61]. The knowledge gained by a DCNN that is trained to solve one problem, e.g. image classification, can be applied to a different but related problem, e.g. object localization. Therefore, instead of designing a new DCNN architecture and training it from scratch, a pre-trained DCNN architecture is used in this thesis. The decisions made in choosing a suitable DCNN architecture and transferring knowledge across domains for the task of detecting grasp types are explained in this section.

3.3.1 DCNN Architecture

The choice of the appropriate architecture for the DCNN largely depends upon the task of interest. For this study, the task of interest is to detect bounding boxes around objects in a cluttered scene and classifying the object into the correct grasp type. Therefore, the task is similar to object detection, which is a vastly studied problem in the field of computer vision. However, a key difference between the standard object detection task and grasp type detection is that instead of classifying individual objects by their names, multiple objects are classified into a single grasp type.

Current state-of-the-art DCNN architectures for object detection can be divided into two categories. First are the region proposal-based architectures, where the proposed regions are classified into different object categories such as cup, bottle, jar etc. These were popularized by the R-CNN architecture [62] and its successors. The other family of object detection DCNNs regard the object detection problem as a regression and classification problem simultaneously in a single stage. Regression is used to determine the parameters for the bounding box (such as width, height, and center coordinates). A most notable one is YOLO [63] and its successors.

Since the chosen model would ultimately be required to run in real-time on a prosthetic hand that has size, weight and power constraints, the YOLO algorithm was chosen due to its relatively smaller memory footprint and lower latency. Specifically, YOLOv5 v6.0 [64] was used in this study. Its architecture is illustrated in Figure 3.3. However, it must be noted that the specific version of the YOLO algorithm has little significance in this study, as long as its performance is comparable to the other state-of-the-art object detection algorithms. This is because the difference in the performance of the model with and without the inclusion of depth information is the primary focus of this study. The relative performance between models that do and do not incorporate depth information is more important than the absolute performance of these models compared to other object detection algorithms.

Note from Figure 3.3 that it is often useful to divide a single-stage object detection DCNN such as YOLO into three parts, namely backbone, neck, and head. The backbone can be thought of as a feature extractor; it transforms the raw RGB data it is provided as input, into feature maps. The neck can be considered as a feature aggregator since it combines and processes the feature maps from previous layers to prepare them for the head. Finally, the head acts as a classifier and regressor, taking the output of the neck and making predictions on the class and bounding box of the objects.

YOLOv5 makes predictions at three different scales, corresponding to strides of 8, 16 and 32. The strides represent the ratio of the input RGB dimensions to the feature map dimensions. For example, at Stride 8, the head of the object detector will make predictions on feature maps of dimensions 80×80 if the input RGB image had dimensions 640×640 pixels. This is important for the fusion of RGB feature maps with depth feature maps, discussed in Section 3.3.3. The multiscale prediction

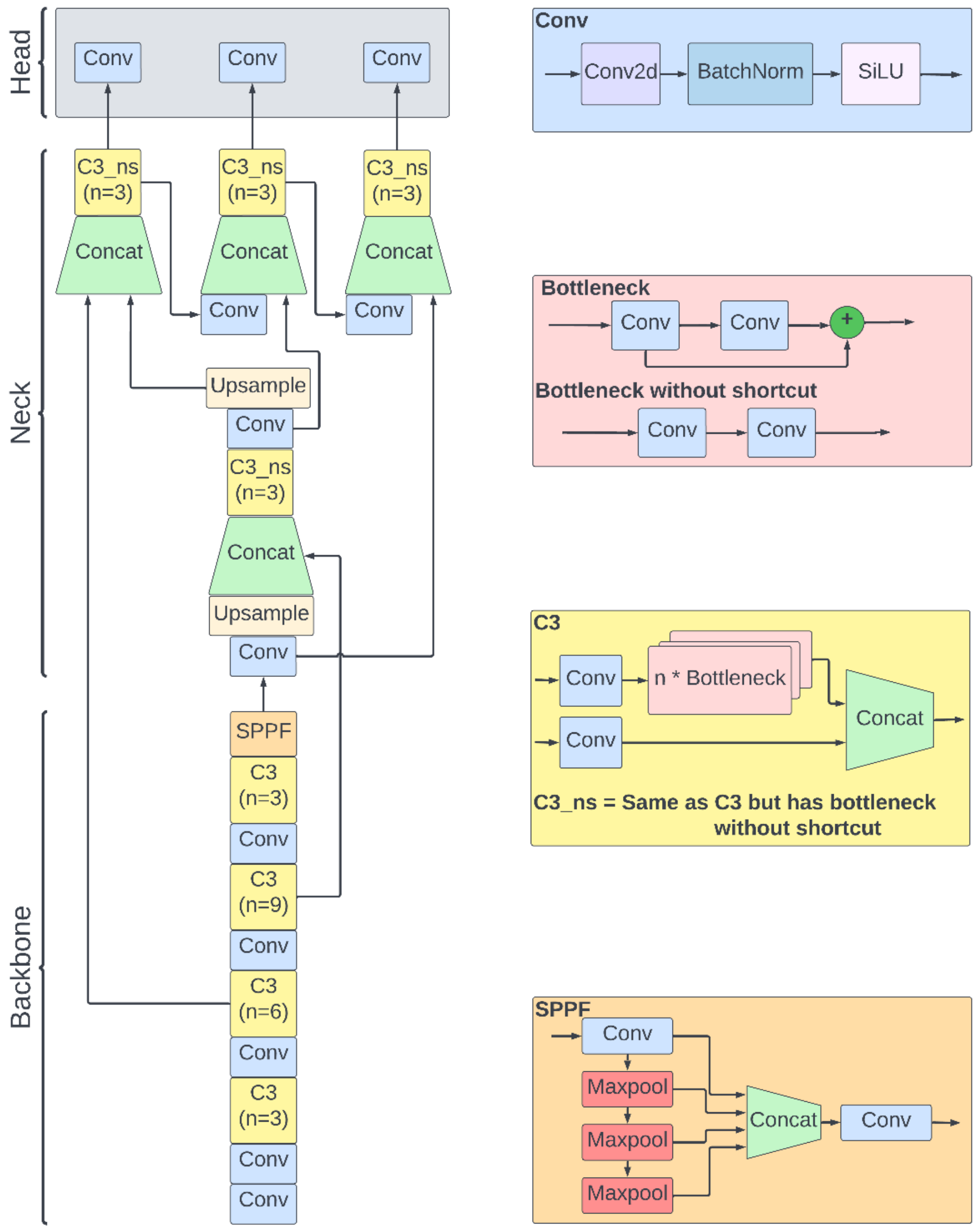


Figure 3.3: YOLOv5 v6.0 model architecture. The simplest block is a Conv block which consists of a convolutional layer, batch normalization layer, and a sigmoid linear unit (SiLU) activation layer. The Concat block concatenates its inputs channel-wise. The Upsample layer is used to resize the feature maps so they can be concatenated. The backbone, neck and head have also been labelled [64].

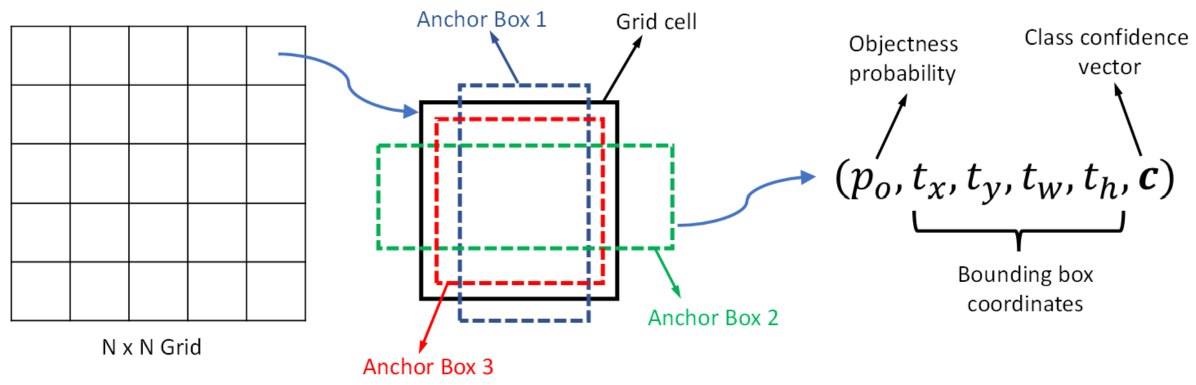


Figure 3.4: Bounding-box prediction. The input image is divided into an N×N grid, where $\boldsymbol{N\epsilon\{20, 40, 80\}}$, corresponding to the three scales at which detection occurs. For each grid cell, the grasp detection DCNN makes predictions for three anchor boxes. For each anchor box, the objectness probability, bounding box coordinates, and class confidence vector are estimated.

paradigm described here and used by YOLOv5 was first used in YOLOv3 and was shown to improve the detection of smaller objects [65].

At each scale, three anchor boxes that were calculated using k-means clustering on the MS COCO dataset [65] were used. Figure 3.4 shows how the bounding boxes are predicted using the anchor boxes. The output tensor at each scale can be thought of as a N x N grid. For each grid cell and for each anchor box, the network predicts a vector that has three parts: objectness probability, bounding box coordinates, and the class confidence vector. The objectness probability is used to determine whether an object is present in the grid cell. t_x, t_y, t_w and t_h are used to predict the bounding box parameters using the following equations:

$$b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x \qquad (Eq. 3.5)$$

$$b_y = \left(2 \cdot \sigma\left(t_y\right) - 0.5\right) + c_y \qquad (Eq. 3.6)$$

$$b_w = p_w \cdot (2 \cdot \sigma(t_w))^2 \qquad (Eq. 3.7)$$

$$b_h = p_h \cdot (2 \cdot \sigma(t_h))^2 \qquad (Eq. 3.8)$$

where b_x, b_y are the center coordinates of the bounding box, b_w, b_h are the width and height of the bounding box, c_x, c_y are the offsets of the grid cell from the top-left corner of the image and p_w

and p_h are the width and height of the anchor box, respectively. $\sigma()$ is the sigmoid activation function which ensures that the output of the model is constrained between 0 and 1. The class confidence vector has length equal to the number of classes for which the network is trained (two in the case of this research); where each element in the vector corresponds to the network's confidence in the class being true for a given grid cell. Therefore, the bounding box predictions have dimensions of N×N×[3*(1+4+2)], where N represents the feature map width and height at a given scale.

The predictions are used to calculate the *binary cross-entropy losses* for the objectness and class predictions, and the Complete Intersection over Union (*CioU*) [66] loss for the bounding box offset predictions. The *objectness loss* is used to optimize the model to differentiate between an object and the background, while the *class loss* teaches the model which grasp type an object belongs to. On the other hand, the *CioU loss* helps to ensure that the predicted bounding box accurately represents the ground truth bounding box. The losses are then summed over all three scales. Finally, the loss is backpropagated to update the model parameters using the SGD optimization algorithm with a learning rate of 0.01.

3.3.2 Transfer Learning

As mentioned in Section 3.2.4, the number of images collected for the grasp-type dataset may be insufficient for the model to learn generic high-level features that would be necessary to make predictions on novel objects without overfitting on the training data. While data augmentation helps reduce the correlation between the examples shown to the DCNN during training, training time can be further reduced, and the model can be further generalized, by transferring knowledge from other datasets. For this study, the YOLOv5 v6.0 model was initialized with parameters pretrained on the MS COCO dataset for the object detection task. This effectively increases the number of training examples shown to the model without the need for a larger task-specific labelled dataset. To adapt the pre-trained model for the grasp-type detection task, the head of the model is modified to classify objects into one of the two grasp types instead of one of the 80 object categories in the MS COCO dataset.

3.4 Depth Data Representation

To evaluate the impact of including depth data on the detection of grasp types for novel objects, the depth data must first be acquired and then fused with the DCNN model, which is only capable of accepting RGB images as input.

One way to acquire depth data is to use dedicated hardware such as a depth sensor or a stereo camera setup [33], [47], [67], [68]. However, this option becomes unfeasible when the size, weight, and power consumption constraints of a prosthetic device are considered. Another way of acquiring depth data is to estimate it from images acquired through the RGB sensor using a pretrained monocular depth estimation DCNN. Estimating depth maps does not require dedicated hardware, since the RGB sensor is needed regardless of whether depth information will be used in making grasp-type detections. Therefore, this study focuses on the impacts of using estimated depth information acquired through a pre-trained DCNN rather than by depth data measured through dedicated hardware, as the former is more feasible in real-world application. To the best of the author's knowledge, this is the first study that evaluates the impact of using estimated depth data on the task of detecting grasp types for novel objects.

The remainder of this section provides details on how the depth maps were estimated, different methods of encoding depth maps, and strategies for fusing RGB data with depth data.

3.4.1 Depth Map Estimation

Recently, the capabilities of DCNNs have been explored to estimate depth from monocular RGB images [69]–[73]. However, the problem is ill-posed since any depth estimation from monocular images will be prone to scale ambiguity. One way of minimizing the effects of scale ambiguity and ensuring global consistency in scale is to treat the task of depth estimation from monocular images as a regression problem [74]. Given an RGB image and its ground truth depth map, a DCNN model can be trained to directly regress the depth values. Such a DCNN has the capability of operating on a wide range of scenes, only limited by the examples it was shown during training.

For this work, it is important to choose a pre-trained monocular depth estimation model that generalizes well to datasets that the model has not seen during training, since it is not possible to fine tune the model to the dataset described in Section 3.2 due to a lack of ground truth depth data. Moreover, in real world applications, it should not be assumed that the model will operate in familiar environments.

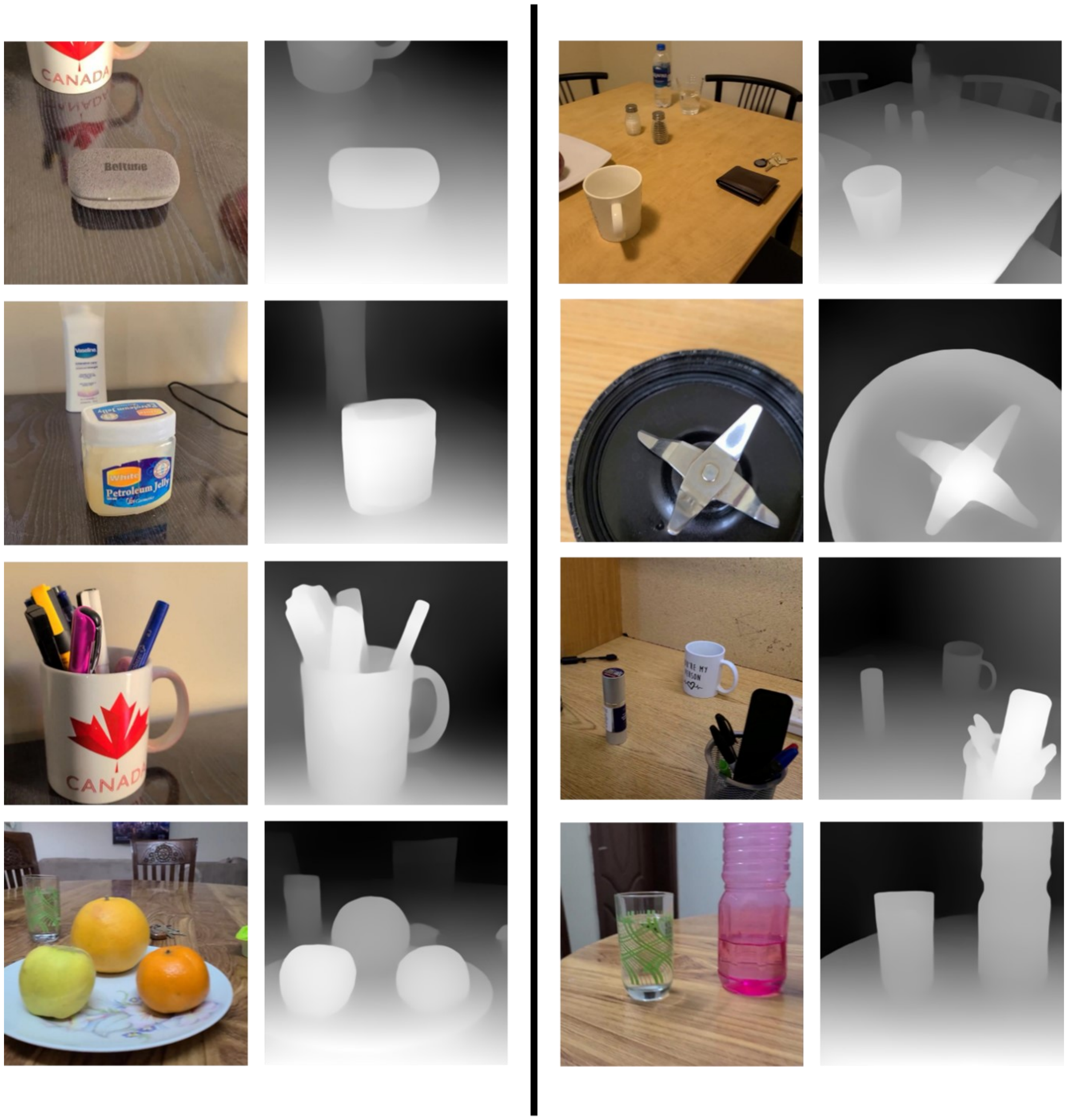


Figure 3.5 Qualitative results of MiDaS [75] on the dataset described in Section 3.2

For this thesis, the MiDaS [75] pre-trained monocular depth estimation model was used. To the best of the author’s knowledge, and at the time of writing this thesis, MiDaS has the best performance on unseen datasets compared to other state-of-the-art approaches by a large margin [75]. MiDaS utilizes the DCNN architecture proposed in [76] and trains it on a large collection of diverse datasets consisting of 75,000 examples. This allows MiDaS to provide state-of-the-art performance in a variety of unconstrained environments and for inputs that it has not seen during training. This makes the model ideal for this thesis research, since any system designed for a prosthetic device should be capable of performing in a wide variety of unexpected scenarios. Since it is not possible to quantitatively validate the output of the model for the dataset described in Section 3.2 due to a lack of ground truth data, a qualitative analysis was performed and gave satisfactory results. From Figure 3.5, it can be observed that MiDaS performs well in a variety of challenging conditions, such as, occlusions, cluttered scenes, shiny surfaces, transparent objects, and reflections even without being fine tuned on the dataset described in Section 3.2.

MiDaS outputs an inverse depth map normalized between 0 and 255. A depth map can be thought of as an image where pixel intensity values represent the depth of the surfaces in the scene from a given viewpoint. In an inverse depth map, the greater the pixel intensity value, the closer it is to the point from which the scene is being viewed. An example output of the MiDaS model can be seen in Figure 3.6. Figure 3.6(a) shows the original RGB image that was provided as an input to MiDaS and Figure 3.6(b) shows a gray-scale visualization of the output depth map. The brighter a pixel, the closer it is to the camera. Although the estimated depth maps are not very sensitive to small changes in depth and cannot be used to accurately reconstruct the 3D geometry of the objects, they can provide valuable information. It can be observed that the model can identify the edges of the objects with high accuracy, and provide information of the object shape, which is important in determining the object grasp type. Moreover, the model can distinguish the objects in the foreground from objects in the background, thus identifying the object of interest. The question of whether or not these depth cues are sufficient in providing descriptive features that will allow a DCNN to better discriminate between different grasp types can be considered as the primary aim of this study.

3.4.2 Depth Encoding

Feature maps must first be extracted from the estimated depth map before it can be fused with the grasp type detection DCNN model. The feature maps should have a strong correlation with the outputs of the grasp type detection DCNN. As discussed in Section 3.3.2, transfer learning can help prevent overfitting and improve the generalizability of DCNNs. Therefore, using a feature extractor

DCNN that has been pre-trained on a large dataset of depth maps for a task similar to grasp-type detection (such as object detection) would be the ideal choice. However, such pre-trained depth map feature extractors are not publicly available to the best of the author's knowledge. One solution is to train a depth map feature extractor from scratch, i.e., initialize its parameters randomly and update them by backpropagating the loss. However, this will cause the training time to increase and may result in overfitting. Another approach that has been shown to be beneficial is to initialize the DCNN model used for the depth-map feature extraction with parameters trained on ImageNet data [34]–[36]. However, ImageNet only has RGB images, which have three channels. Therefore, any model trained on ImageNet will expect to receive three channels as input. However, depth maps are single channel. Hence, the single-channel depth map needs to be converted into three channels before a DCNN pretrained on an RGB dataset can be used as a depth-map feature extractor.

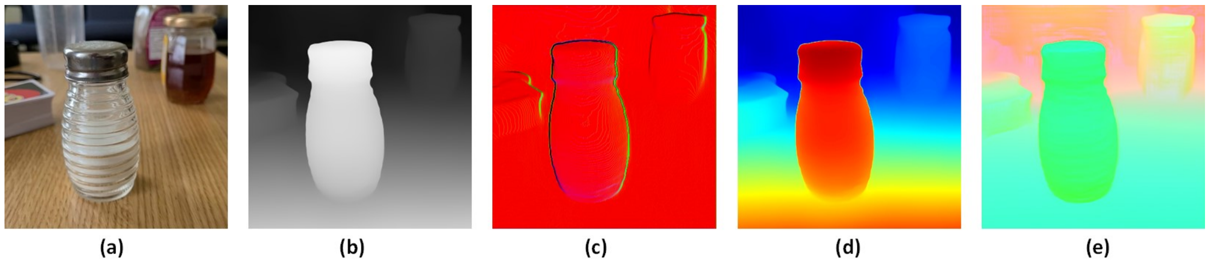


Figure 3.6: Example inputs to the DCNN pipeline: (a) RGB input to the backbone and neck of the object detector, (b) grayscale rendering of the depth map generated by MiDaS [75], (c) surface normal encoding, (d) jet colormap encoding and (e) HHA encoding.

A naïve approach would be to simply duplicate the single-channel depth map into three channels. A more involved approach is to encode the depth data using handcrafted features that have three channels. For example, one approach is to represent the x, y, and z components of the surface-normal vector at each pixel as separate channels [36] (Figure 3.6c). Moreover, it has also been shown that colourizing the depth map using the jet colormap (Figure 3.6d) is a computationally inexpensive way of encoding depth maps, where the distance value of each pixel is mapped to colour values ranging from red (near) over green to blue (far) [34]. Another approach which has been shown to be useful is HHA encoding [35], which calculates the horizontal disparity, height above ground, and the angle the surface normal makes with gravity at every pixel of the depth map, and represents them as separate channels (Figure 3.6e). [36]

This thesis aims to investigate the impact different methods of encoding depth data has on the performance of the grasp-type detection DCNN. Therefore, the following methods for encoding depth data are evaluated and compared: 1) duplicated depth channels, 2) surface normal encoding, 3) jet colormap encoding, and 4) HHA encoding. Results are presented and compared in Chapter 4.

3.5 Fusion of Depth and RGB Feature Maps

As discussed in Section 3.4.2, a DCNN pretrained on RGB images is used to extract feature maps from encoded three-channel depth data. This DCNN will be referred to as the *depth feature extractor* and its output as *depth feature maps* from here on. The depth data provided as an input to the depth feature extractor is either encoded using handcrafted features or is a simple duplication of the single channel estimated depth map. The single-channel depth map is obtained using a pre-trained monocular depth estimator DCNN (MiDaS) as described in Section 3.4.1. After evaluating several candidates (see Appendix A), the 18-layer version of the Residual Network (ResNet18) [53] was selected as the depth feature extractor for this thesis.

In this thesis, the channel-wise concatenation operation is used to fuse depth feature maps with RGB feature maps. However, the point along the neck of YOLOv5 v6.0 that this operation is performed is treated as a hyperparameter, in order to find the optimal method of fusing depth feature maps with RGB feature maps. Three novel methods of fusion were developed and evaluated: (1) Early Fusion, (2) Intermediate Fusion and, (3) Late Fusion. The output of the concatenation operation of the grasp detection pipeline with the depth feature maps will be referred to as the *fused feature maps*.

3.5.1 Early Fusion

The process of early fusion is depicted in Figure 3.7. The backbone of the grasp detection DCNN has been collapsed into a single block for visualization purposes. RGB data is provided as an input to both the backbone of the grasp detection DCNN as well as the depth estimator. The depth estimator outputs a single channel depth map, which is converted into three channel depth data via the depth encoder. The three channel depth data is passed to the depth feature extractor, which outputs depth feature maps. The depth feature maps are concatenated with the RGB feature maps near the bottom of

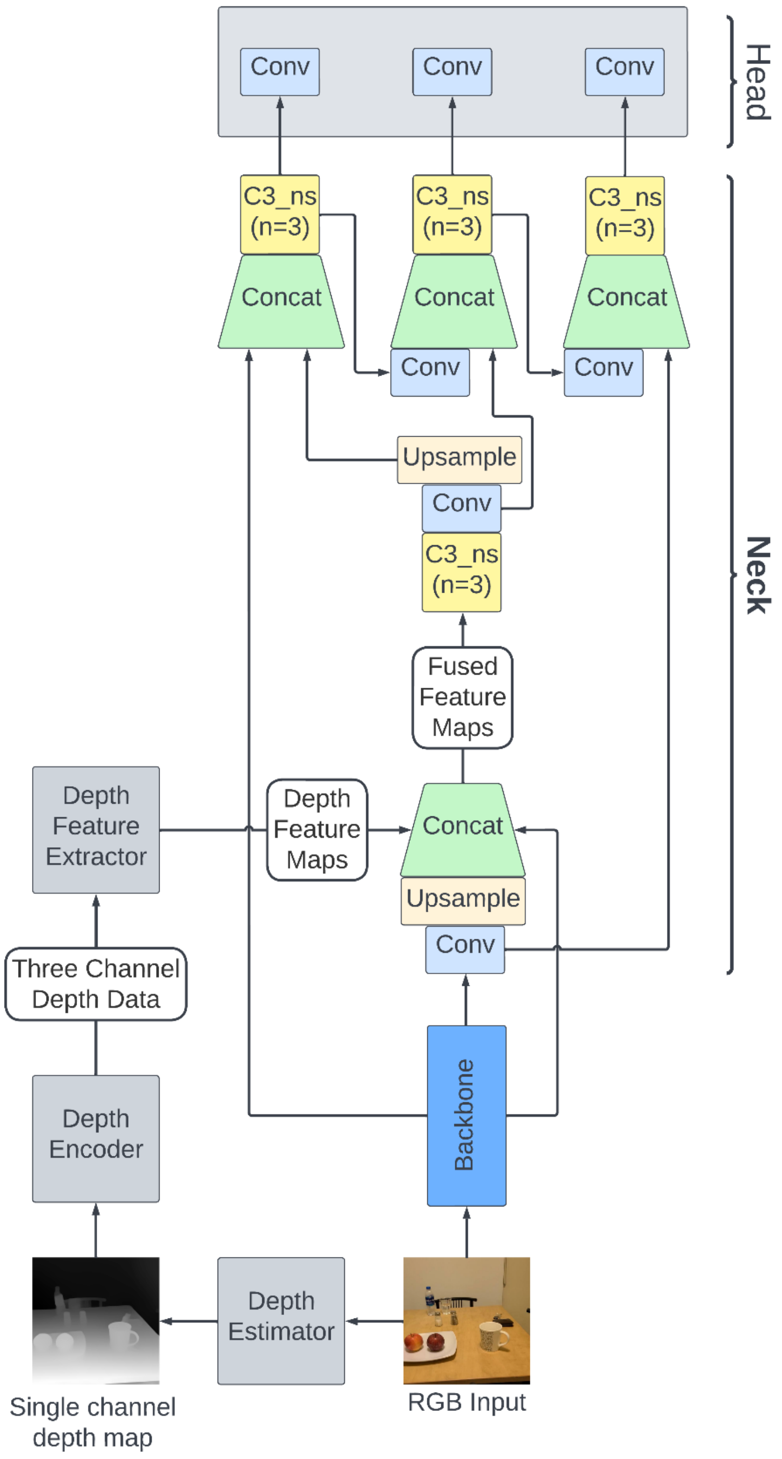


Figure 3.7: Early fusion of depth features with the grasp detection DCNN.

the neck of the grasp detection DCNN. The channel-wise concatenated feature maps are referred to as fused feature maps and are passed to the next layer in the grasp detection DCNN.

In order to successfully channel-wise concatenate the depth feature maps with the RGB feature maps, the dimensions of both feature maps must match. Therefore, the depth feature maps are resized to match the RGB feature maps using nearest neighbor interpolation. Note that the tensor that represents the fused feature maps has a different length along the channel dimension compared with the RGB feature maps. Therefore, the filters in the next convolutional layer will have a different dimension compared to the pre-trained filters, and hence, will need to be trained from scratch. However, subsequent layers will be able to use pre-trained parameters as all other output tensors will still have the same dimensions.

3.5.2 Intermediate Fusion

Intermediate fusion is depicted in Figure 3.8. The depth feature map generation process is the same as for early fusion. However, the difference is in the point at which the depth feature maps are concatenated with the RGB feature maps. In intermediate fusion, the depth feature maps are channel-wise concatenated with the RGB feature maps near the top of the neck of the grasp detection DCNN. Observe in Figure 3.8 that there are three concatenation operations near the top of the neck, each concatenating feature maps at a different scale, providing the model a better scale invariance. Therefore, the depth feature maps are concatenated with the RGB feature maps at each scale.

Concatenation layers allow low level features from earlier layers to be passed to layers deep in the DCNN. Therefore, to allow the grasp detection DCNN to learn from low level as well as high level depth features, depth feature maps are extracted from different layers in the depth feature extractor. In particular, depth feature maps are extracted at three different strides, corresponding to the three different scales at which grasps are detected. As with early fusion, the depth feature maps are resized using nearest neighbor interpolation to match the corresponding RGB feature map's dimensions. Each output tensor of the concatenation layer is referred to as fused feature maps and are separately sent to the next layer in the grasp detection DCNN.

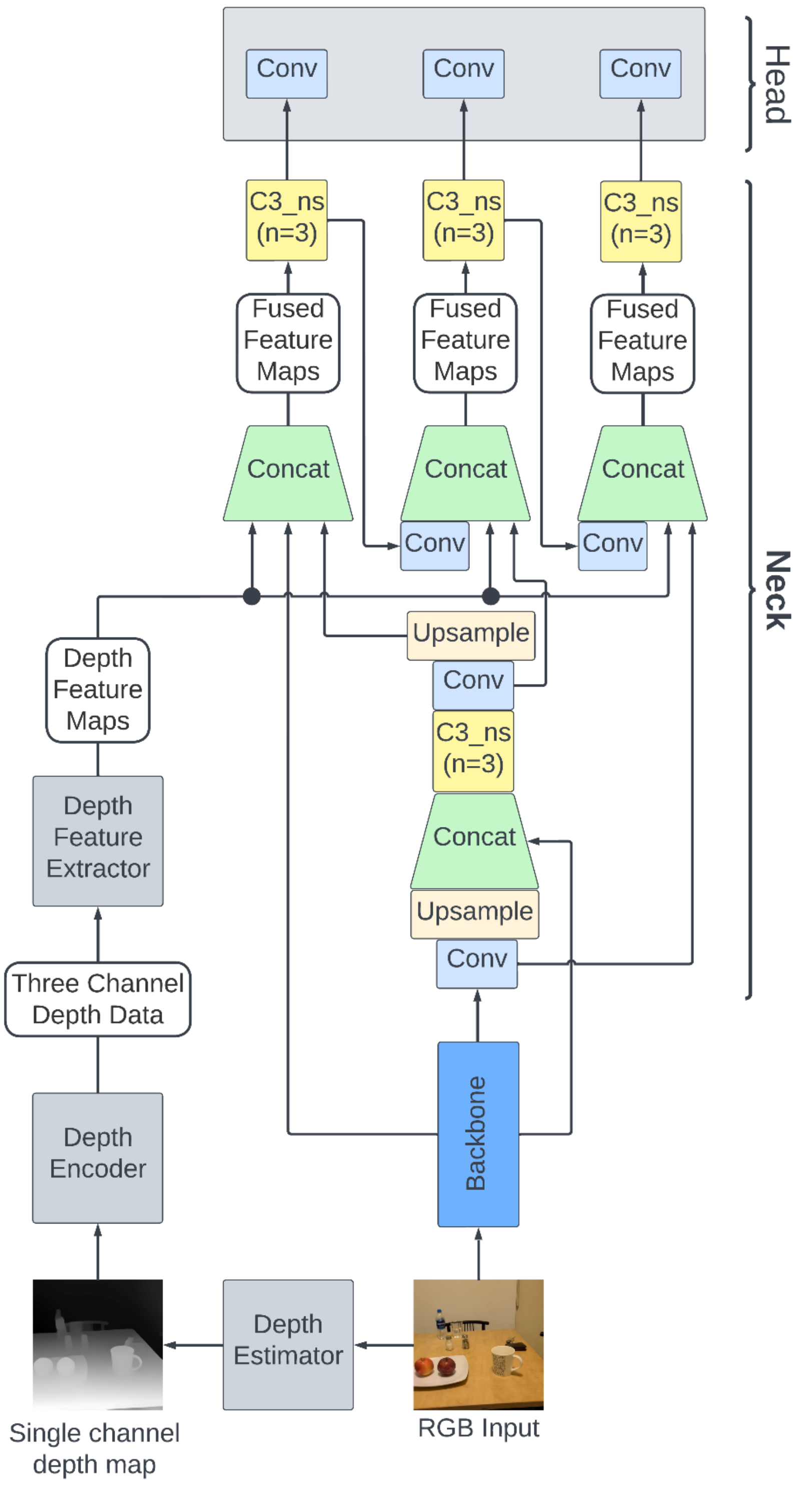


Figure 3.8: Intermediate fusion of depth features with the grasp detection DCNN.

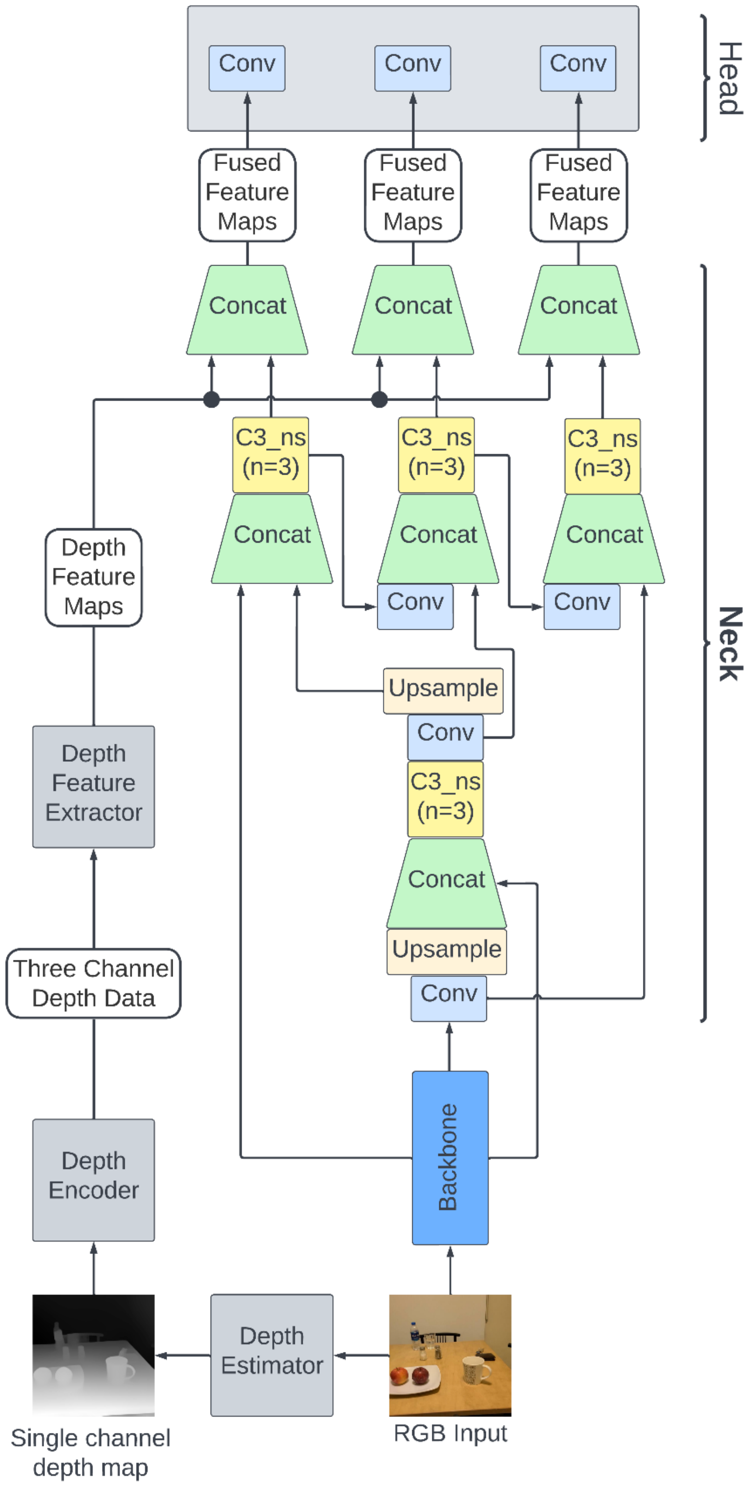


Figure 3.9: Late fusion of depth features with the grasp detection DCNN.

3.5.3 Late Fusion

Late fusion, depicted in Figure 3.9, also fuses depth feature maps at three different strides of the depth feature extractor with the grasp detection DCNN, similar to intermediate fusion. However, the concatenation occurs just before the RGB feature maps are sent to the classification head. To achieve this, three new concatenation layers are introduced, which take RGB feature maps that are sent as input to the convolutional layers in the classification head and concatenates them with depth feature maps that are taken from the depth feature extractor. Each concatenation layer uses depth feature maps from a different layer in the depth feature extractor. Again, similar to early and intermediate fusion, the depth feature maps are resized using nearest neighbor interpolation to match the corresponding RGB feature map's dimensions. The fused feature maps are then used directly as an input to the corresponding convolutional layer in the classification head.

3.6 Experimental Procedure

All experiments used YOLOv5-large v6.0 [64] pretrained on the MS COCO dataset [58] as the grasp type detector, MiDaS v3.0 DPT-large pretrained on 10 datasets (ReDWeb [77], DIML [78], Movies [75], MegaDepth [79], WSVD [80], TartanAir [81], HRWSI [82], ApolloScape [83], BlendedMVS [84], IRS [85]) as the depth estimator, and the 18-layer version of the Residual Network (ResNet18 [53]) pre-trained on ImageNet [57] data as the depth feature extractor. For each level of fusion (early, intermediate, or late), the experiments were repeated for the different depth encodings (duplicated depth maps, surface normals, jet colormap and HHA). As a control, an experiment with no depth fusion was also performed.

The grasp detector backbone was provided with an RGB input of dimensions 640×640×3, which was processed into sets of RGB feature maps of dimensions 80×80, 40×40, and 20×20, at strides of 8, 16, and 32, respectively. The third dimension of the feature maps (number of feature maps) depended on the level of fusion.

The depth estimator was provided an RGB input of dimensions 384×384×3 and outputted a depth map of the same dimensions. The depth feature extractor was provided with encoded depth data, which could be in the form of duplicated depth maps, surface normal, jet colormap or HHA. Since ResNet18, which was used as the depth feature extractor, was pretrained for input dimensions of 224×224×3, the depth data were also resized to these dimensions.

For early fusion, depth feature maps were extracted at a stride of 8 from the depth feature extractor, resulting in feature maps of dimension 28×28×128. The depth feature maps for early fusion were resized for concatenation along the third dimension with stride 16 RGB feature maps that had dimensions 40×40×1024, resulting in fused feature maps of size 40×40×1152.

For intermediate and late fusion of depth data, since fusion was repeated at three scales, the depth feature maps were extracted at strides of 4, 8, and 16 from ResNet18, resulting in three sets of feature maps of dimensions 56×56×64, 28×28×128, and 14×14×256, respectively. The first two dimensions were resized to match the dimensions of the RGB feature maps at strides 8,16 and 32, respectively. The RGB and depth feature maps were then concatenated along the third dimension. Therefore, the fused feature maps had dimensions 80×80×576, 40×40×640, and 20×20×1280 at the three scales, respectively, for intermediate fusion. Similarly, for late fusion, the fused feature maps had dimensions 80×80×320, 40×40×640, and 20×20×1280 at the three scales, respectively.

The fused feature maps were then sent to the next layer in the network, and eventually to the classification head where predictions were made for the objectness probability, bounding box, and class confidence vector (as described in Section 3.3.1). The grid sizes were 80×80, 40×40, and 20×20 at each scale, respectively. For each grid cell in each grid, predictions for three bounding boxes were made (corresponding to the three anchor boxes). This resulted in a large number of bounding box predictions (19,200 just for the 80×80 grid). During training, the predictions were used for loss calculation. However, for performance metric calculations, the predictions were used to perform detections. Ideally, each object should be associated with one bounding box (and its corresponding grasp type detection) only. Therefore, the predicted bounding boxes were filtered using non-maximum suppression (NMS) [86]. NMS first removes all bounding boxes with a confidence score lower than 0.5. The confidence score is a product of the objectness probability and the class confidence:

$$confidence = objectness\ probability \times class\ confidence \qquad (Eq. 3.9)$$

The remaining bounding boxes were compared with each other. If two bounding boxes had an Intersection-over-union (IoU) greater than 0.5 and a high class confidence for the same grasp type, then it was assumed that they were detecting the same object. Therefore, only the bounding box with the greater confidence was retained, while the other was rejected. IoU is a measure of proximity between two bounding boxes, and is given by:

$$IoU = \frac{Area\ of\ overlap\ between\ the\ bounding\ boxes}{Area\ of\ union\ between\ the\ bounding\ boxes} \qquad (Eq. 3.10)$$

NMS was repeated for all the bounding box predictions at each scale, and then across scales, with the aim to retain only one bounding box per object.

For training, the RGB data was augmented as described in Section 3.1.4 before being passed to the objector detector backbone and neck. The output of the depth encoder was also augmented to match the RGB input during training; however, no HSV augmentation was applied to the depth data. As an example, an augmented RGB input and its corresponding augmented HHA encoding can be seen in Figure 3.10. In practice, the encodings were precomputed for the training data so that the mosaics could be created. No augmentation was used during validation or test time for calculating the performance metrics.

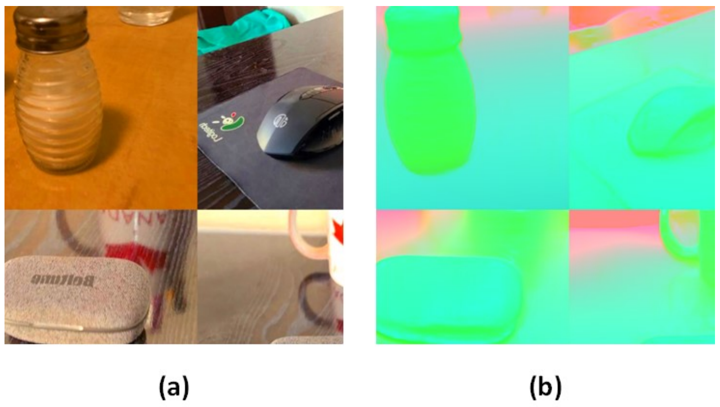


Figure 3.10: Augmented-depth-map encoding. Encodings of estimated depth maps undergo the same mosaic, translation, scaling, and flipping augmentation as the RGB images. No HSV augmentation is applied on the encodings. (a) the augmented RGB input (b) the corresponding augmented depth map encoding.

Since training was initiated from pre-trained weights, there was no need for extensive training. It was found that freezing the object detector backbone, i.e., not updating its weights, and training for 10 epochs resulted in the best performance and prevented overfitting. All experiments were performed on a GeForce RTX 3080 Graphics Processing Unit (GPU) with 10 GB of VRAM and a batch size of

16. Note that the depth estimator was not trained; only the grasp type detection DCNN and the depth feature extractor DCNN were trained.

3.7 Metrics for Evaluating Grasp-Type Detection

Since the task of grasp-type selection described in this thesis is quite similar to the object detection task, the same metrics were used for comparing models as those that have become standard for comparing object detection models. This includes the average precision (AP) and mean average precision (mAP) [87]. AP is calculated by the area under the precision-recall curve for each grasp type and averaging it over the two grasp types. Precision or positive predictive value is a measure of the number of positive predictions that are true positives:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \qquad (Eq.\ 3.11)$$

Recall or sensitivity measures the ability of the model to classify all instances of the class and is given by:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \qquad (Eq.\ 3.12)$$

A prediction is considered to be a true positive only when the confidence in the predicted class is greater than 0.5 and the predicted bounding box has an intersection-over-union (IoU) with the ground truth bounding box of at least 0.5. IoU was calculated using Equation 3.10; however, instead of calculating IoU between two predicted bounding boxes as in NMS, the IoU was calculated between the output of NMS and the ground-truth bounding box.

For mAP, the area under the precision-recall curve was calculated for 10 IoU values, between 0.5 and 0.95, at equal intervals. All the values were then averaged for the 10 IoU values and the two grasp type categories. AP provides a good measure of the ability of the model to correctly classify the grasp type of the object in the scene, while mAP is a measure of the proximity of the predicted bounding box to the ground-truth bounding box.

Chapter 4
Results and Discussion

4.1 Introduction

In this chapter, results are presented for the performance of the DCNN grasp-detection models during training and testing. During training, at the end of each epoch, model predictions were used to calculate the *binary cross-entropy losses* for the objectness and class predictions, and the CIoU [66] loss for the bounding-box offset predictions, as described in Section 3.3.1. The CIoU loss for bounding box prediction is referred to as *box loss* for simplicity. The losses were calculated for the training dataset as well as the validation dataset.

For the validation dataset, at the end of each epoch, the performance metrics (Section 3.7) were calculated to determine the performance of the models. After training was completed, the models with the best performance metrics on the validation dataset were evaluated on the test dataset. The test set contains RGB data from objects that the models had not seen during training; therefore, it provides a better understanding of the model's expected performance in the real world.

To act as a control, the model trained with no depth data was also included in all the results and is referred to as the *no depth* or *base* model, as it is the model against which the impact of fusing depth data was compared. The base model used no depth feature extractor, depth encoding, or feature fusion, and only used RGB data to detect grasp types.

The results in this chapter are organized into separate sections for each of the three levels of fusion (early, intermediate, and late), where for each level of fusion, the models were separately trained using the different methods of encoding estimated depth maps (duplicated depth maps, surface normals, jet colomaps and HHA). The performances of the models are compared and discussed at the end of the chapter in Section 4.5.

4.2 Early Fusion of Depth Data

The plots of the three losses calculated at the end of each epoch on the training set for early fusion as described in Section 3.5.1 are displayed in Figure 4.1. Figure 4.1(a) shows the objectness loss smoothly decreased for all depth encodings. Similar observations can be made for the box and class losses in Figures 4.1(b) and Figure 4.1(c), respectively.

Figure 4.2 displays the plots for the three losses on the validation dataset at the end of each epoch. Figure 4.2(a) shows a downward trend for the objectness loss for all the depth encodings, though the trend is not as smooth as that observed in 4.1(a). This is expected as the loss functions were optimized for the training dataset, not the validation dataset. A similar comparison can be made between the box loss on the training and validation dataset in Figure 4.1(b) and Figure 4.2(b), respectively. On the other hand, for the class loss, many fluctuations can be observed for the validation dataset in Figure 4.2(c). The fluctuations in class loss are greatest for the model trained without depth data and can be observed to be less severe with models that were fused with depth data. A smoother class loss for models with fused depth data indicates that depth data helps the model to better differentiate between the two grasp type classes.

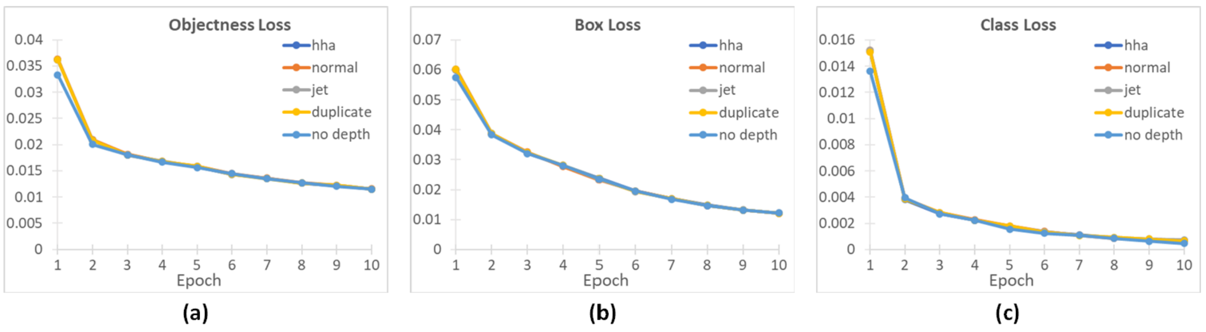


Figure 4.1: Loss functions at the end of each epoch calculated on the training dataset for models with early fusion of depth data. The base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

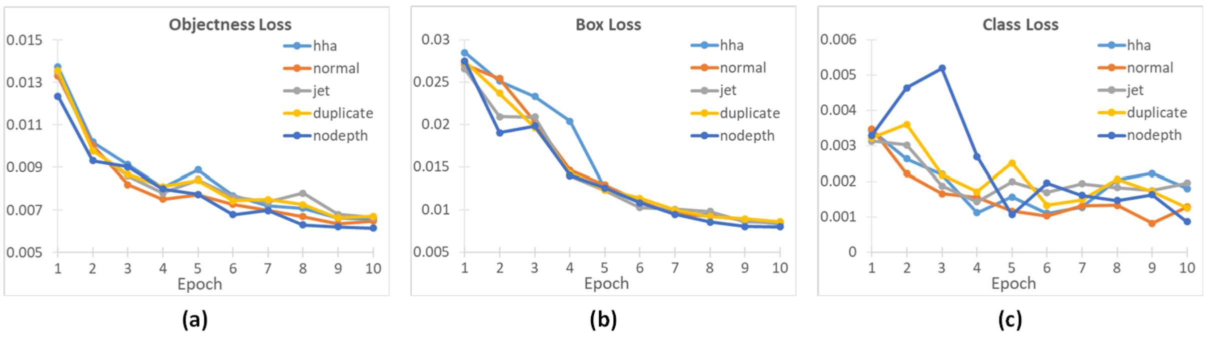
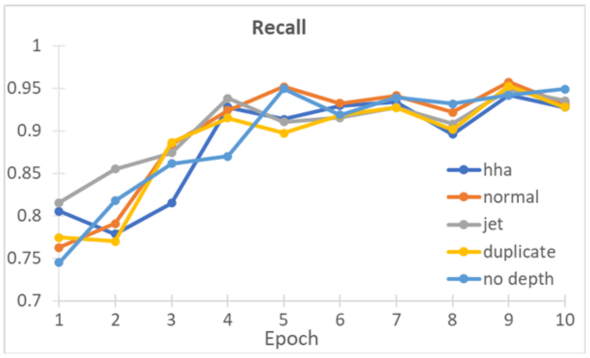
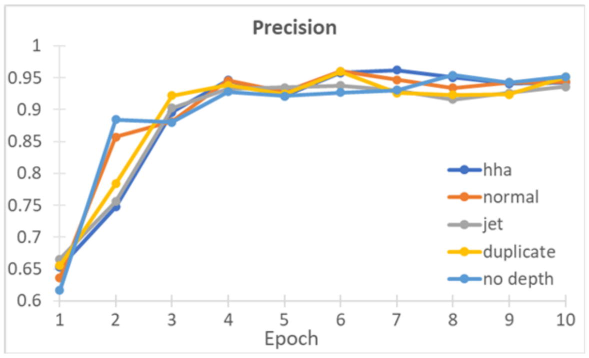


Figure 4. 2: Loss functions at the end of each epoch calculated on the validation dataset for models with early fusion of depth data. The base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

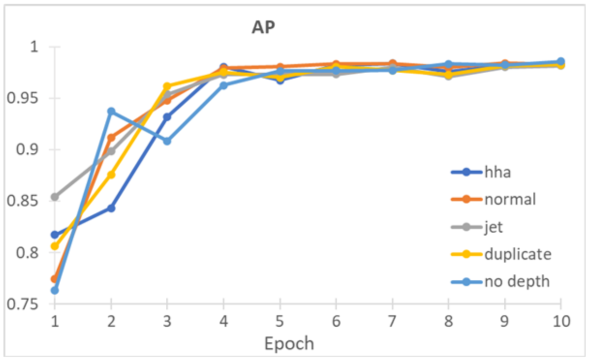
Figure 4.3 displays the performance metrics for the models with early fusion on the validation dataset at the end of each epoch. Fluctuations can be observed for recall in Figure 4.3(a). However, in general, the trend for recall was upwards as training proceeded for all the types of depth encodings. For precision (Figure 4.3(b)), AP (Figure 4.3(c)) and mAP (Figure 4.3(d)), smooth upward trends can be observed. Figures 4.1, 4.2 and 4.3 provide evidence for successful training, as all loss functions had a downward trend while all performance metrics had an upward trend, for all types of depth encoding.



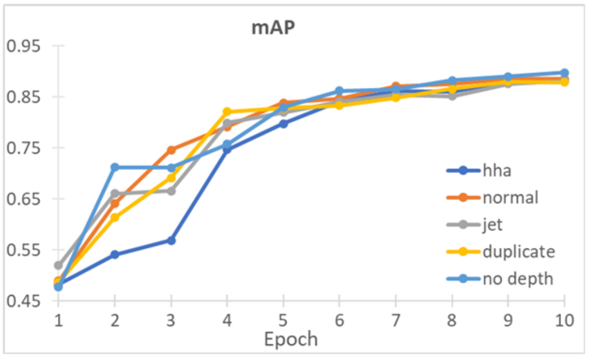
(a)



(b)



(c)



(d)

Figure 4.3: Performance metrics calculated on the validation dataset at the end of each epoch for models with early fusion of depth data**.** The base model with no depth is included for comparing the performance metrics, namely: (a) recall, (b) precision, (c) average precision (AP), and (d) mean average precision (mAP).

The performance metrics on the validation and test datasets of the models with early fusion that had the best metrics on the validation dataset throughout training are shown in Table 4.1. The base model that was not fused with depth information performed better on the validation dataset for all metrics, except for the recall metric, where the model that used surface normal encoding had a recall of 0.957 and the base model had a recall of 0.949. For recall and mAP, the base model outperformed all models that included fused depth data. However, for precision and AP, the model with duplicated depth maps achieved better performance with values of 0.886 and 0.717, respectively, against the base model's 0.827 and 0.704, respectively.

Table 4.1: Performance metrics of the best performing models with early fusion of depth data on the validation and test datasets**.** Best values for each metric are in bold.

Depth Encoding	Validation Set				Test Set			
	Precision	Recall	AP	mAP	Precision	Recall	AP	mAP
No Depth	**0.952**	0.949	**0.986**	**0.898**	0.827	**0.760**	0.704	**0.527**
Duplicate	0.924	0.953	0.982	0.881	**0.886**	0.732	**0.717**	0.510
Surface Normals	0.943	**0.957**	0.984	0.887	0.854	0.728	0.696	0.492
Jet Colormap	0.936	0.934	0.982	0.882	0.839	0.741	0.696	0.501
HHA	0.943	0.928	0.983	0.884	0.837	0.737	0.687	0.498

4.3 Intermediate Fusion of Depth Data

The plots of the three losses calculated at the end of each epoch on the training set for intermediate fusion as described in Section 3.5.2 are displayed in Figure 4.4. An interesting observation for the objectness loss in Figure 4.4(a) and box loss in Figure 4.4(b) can be made for the base model with no depth data fusion. The objectness loss and the box loss for the base model were consistently lower than for the models with depth data fusion at the end of each epoch. However, the same is not true for the class loss (Figure 4.4(c)).

Figure 4.5 displays the plots for the three losses on the validation dataset at the end of each epoch. For the objectness loss in Figure 4.5(a) and box loss in Figure 4.5(b), a similar trend occurred for the base model with no depth data fusion as was observed for the losses on the training dataset. The objectness and box losses were consistently lower for the base model than for the models that were fused with depth data. For the class loss (Figure 4.5(c)), the model that was provided depth data in the form of duplicated depth map started poorly, but quickly recovered to have a class loss comparable with the other models.

Figure 4.6 shows the performance metrics for the models with intermediate fusion on the validation dataset at the end of each epoch. For recall (Figure 4.6(a)), precision (Figure 4.6(b)), and AP (Figure 4.6(c)), smooth upward trends can be observed with comparable performance for all the models. However, for mAP (Figure 4.6(c)), the base model was consistently better than the models that had intermediate fusion of depth data, regardless of the type of depth encoding.

The performance metrics on the validation and test datasets of the models with intermediate fusion that had the best metrics on the validation dataset throughout training can be seen in Table 4.2. The model with no depth fusion had the best metrics on the validation dataset, while on the test dataset, the model that was provided depth data in the form of surface normal encoding performed better when considering the precision, recall and AP metrics. However, on the mAP metric, the base line model's performance was better than that of any of the models that had intermediate fusion of depth data. Furthermore, it can be observed that with a value of 0.760, the baseline model's recall is better than the models that had intermediate fusion using duplicate depth maps, jet colormaps and HHA with values of 0.731, 0.744 and 0.741, respectively.

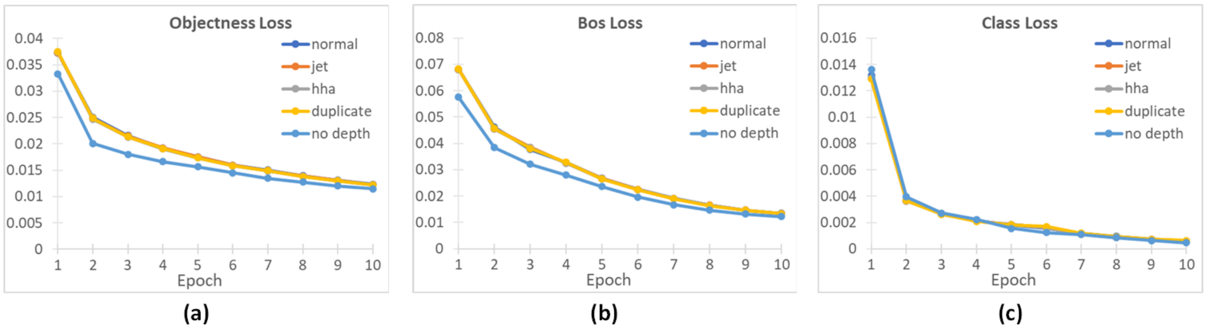


Figure 4.4: Loss functions at the end of each epoch calculated on the training dataset for models with intermediate fusion of depth data. The base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

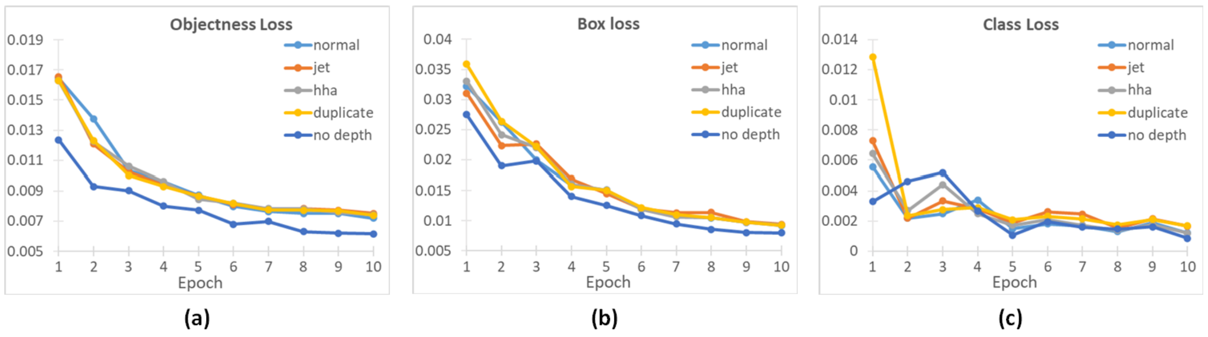


Figure 4.5: Loss functions at the end of each epoch calculated on the validation dataset for models with intermediate fusion of depth data. Base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

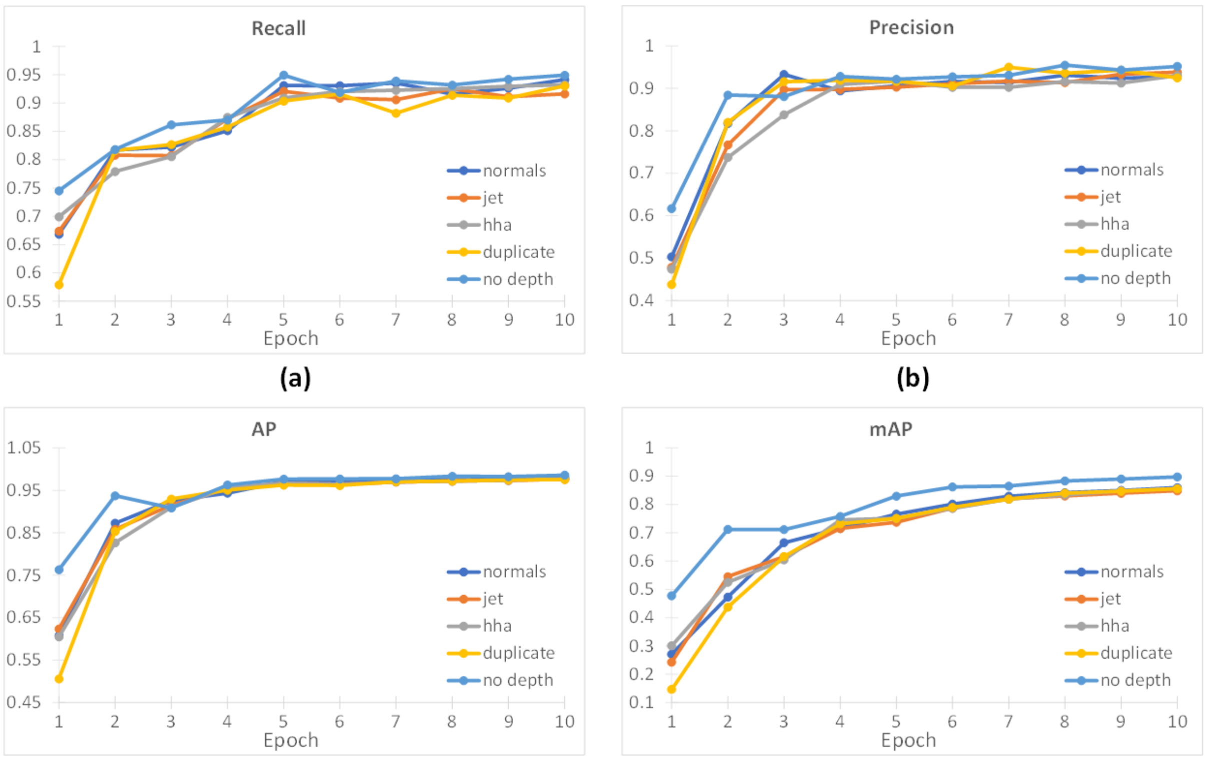


Figure 4.6: Performance metrics calculated on the validation dataset at the end of each epoch for models with intermediate fusion of depth data. The base model with no depth is included for comparing the performance metrics, namely: (a) recall, (b) precision, (c) average precision (AP), and (d) mean average precision (mAP).

Table 4.2: Performance metrics of the best performing models with intermediate fusion of depth data on the validation and test datasets**.** Best values for each metric are in bold.

Depth Encoding	Validation Set				Test Set			
	Precision	Recall	AP	mAP	Precision	Recall	AP	mAP
No Depth*	**0.952**	**0.949**	**0.986**	**0.898**	0.827	0.760	0.704	**0.527**
Duplicate	0.923	0.930	0.976	0.856	0.872	0.731	0.708	0.507
Surface Normals	0.928	0.942	0.980	0.860	**0.884**	**0.763**	**0.744**	0.526
Jet Colormap	0.938	0.916	0.976	0.849	0.852	0.744	0.711	0.510
HHA	0.930	0.933	0.978	0.856	0.880	0.741	0.718	0.517

4.4 Late Fusion of Depth Data

The plots of the three losses calculated at the end of each epoch on the training set for late fusion as described in Section 3.5.3 are displayed in Figure 4.7. The objectness loss (Figure 4.7(a)), box loss (Figure 4.7(b)) and class loss (Figure 4.7(c)) plots all show a smooth downward trend for all models, indicating that the training went well.

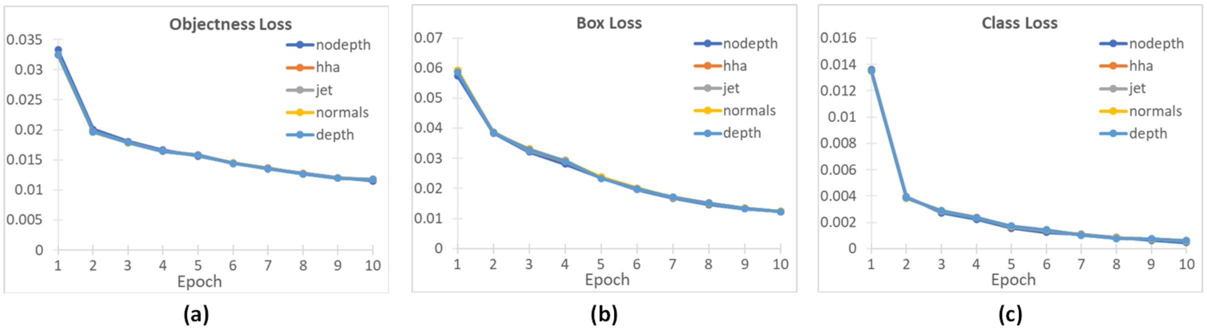


Figure 4.7: Loss functions at the end of each epoch calculated on the training dataset for models with late fusion of depth data. The base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

Figure 4.8 displays the plots for the three losses on the validation dataset at the end of each epoch. The objectness loss (Figure 4.8(a)) had a few fluctuations for the model with no depth fusion; however, the trend for all the models was generally downwards. For the box loss (Figure 4.8(b)), all models started with fluctuations, but eventually had a downward trend as training progressed. On the other hand, for the class loss (Figure 4.8(c)), all models had high fluctuations, but an overall downward trend as training progressed. The fluctuations in the class loss compared with the steady downward trend in the objectness and box loss on the validation dataset indicates that the models successfully found the objects and their bounding boxes but struggled in differentiating the grasp-type of the objects.

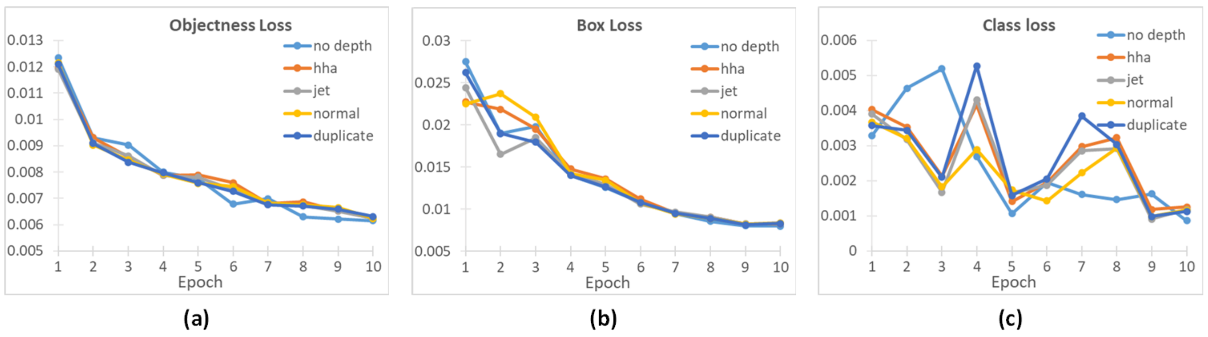


Figure 4. 8 Loss functions at the end of each epoch calculated on the validation dataset for models with late fusion of depth data. The base model with no depth is included for comparing the three losses: (a) objectness loss, (b) box loss, and (c) class loss.

Figure 4.9 displays the performance metrics for the models with late fusion on the validation dataset at the end of each epoch. Fluctuations in recall (Figure 4.9(a)) and class loss (Figure 4.8(c)) occured synvhronously on the validation dataset. Thus, even though the objects and their bounding boxes were found, the confidence value of the correct grasp type was low in the earlier epochs, resulting in increased false negatives. Note that the class confidence needs to be at least 0.5 or 50% for the prediction to be classified as a true positive. An increased number of false negatives resulted in a low recall value according to Equation 3.12 in Section 3.7. However, the models learned to rectify this problem, as both the class loss and recall metric converged to acceptable values towards the final epochs.

The performance metrics on the validation and test datasets of the models with late fusion that had the best metrics on the validation dataset throughout training can be seen in Table 4.3. On the validation dataset, the base model had the best performance, except for recall, which was 0.949 for the base model against 0.963 for the model that used jet colormap encoding. However, on the test dataset, the models fused with depth data outperform the base model across all metrics, regardless of the method of depth encoding. In particular, the model that used jet color maps as depth encoding had the best performance on the recall and AP metrics, with values of 0.813 and 0.768, respectively. The model that used HHA encoding had the best performance on the precision and mAP metrics, with values of 0.881 and 0.594, respectively.

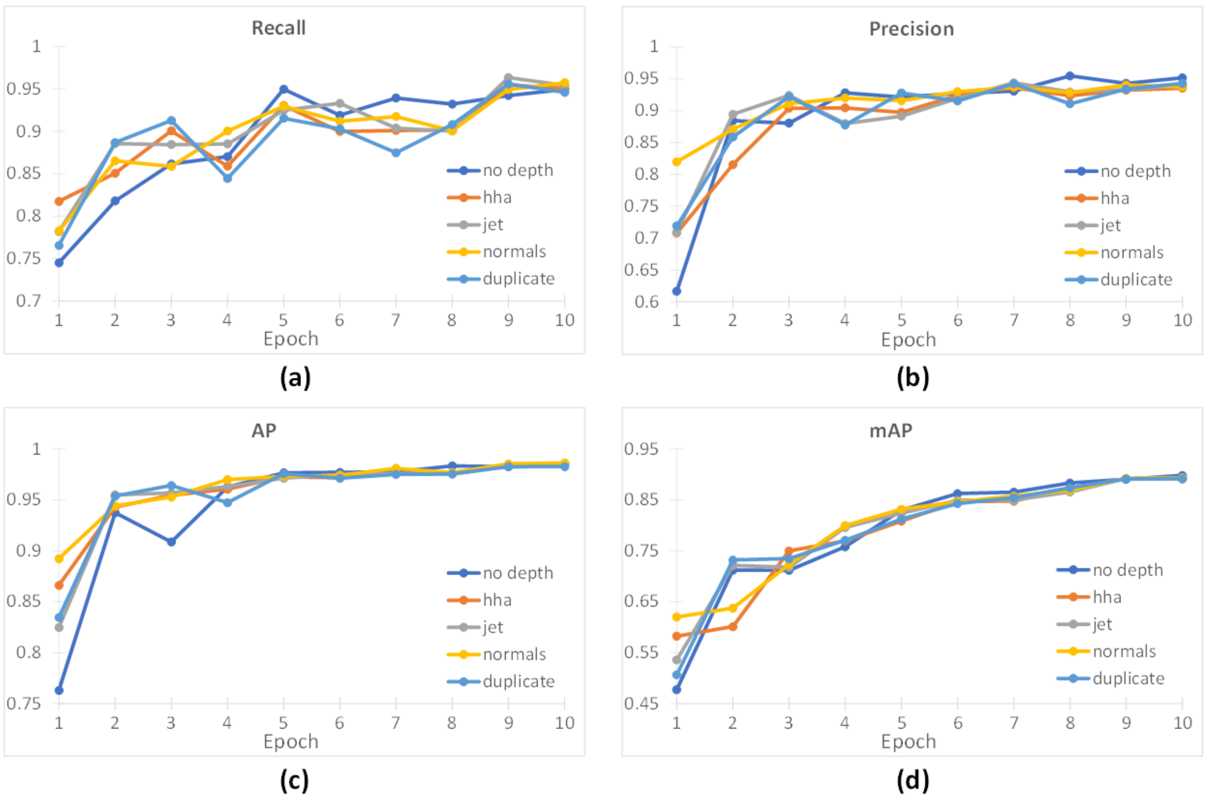


Figure 4.9: Performance metrics calculated on the validation dataset at the end of each epoch for models with late fusion of depth data**.** Base model with no depth also included for comparing the performance metrics, namely: (a) recall, (b) precision, (c) average precision (AP) and (d) mean average precision (mAP)

Table 4.3: Performance metrics of the best performing models with late fusion of depth data on the validation and test datasets**.** Best values for each metric are in bold.

Depth Encoding	Validation Set				Test Set			
	Precision	Recall	AP	mAP	Precision	Recall	AP	mAP
No Depth*	**0.952**	0.949	**0.986**	**0.898**	0.827	0.760	0.704	0.527
Duplicate	0.943	0.946	0.983	0.892	0.857	0.792	0.758	0.585
Surface Normals	0.937	0.957	**0.986**	0.893	0.847	0.794	0.757	0.580
Jet Colormap	0.933	**0.963**	0.983	0.892	0.836	**0.813**	**0.768**	0.570
HHA	0.935	0.950	0.985	0.893	**0.881**	0.792	0.766	**0.594**

4.5 Discussion

Table 4.4 shows a comparison of the base model with the models that used fused feature maps and had the best values for the performance metrics evaluated on the test dataset. It can be observed that for each performance metric, fusion of depth data resulted in improved performance on novel objects. The model that used early fusion with duplicated depth maps for depth data encoding had the best precision out of all the models that used fused feature maps, which was an improvement of 7.1% over the base model.

Table 4.4: Comparison of grasp-type detection performance with and without the fusion of estimated depth data.

Metric	**Base model value**	**Best fused model value**	**Percent change**	**Encoding type of best model**	**Fusion type of best model**
Precision	0.827	0.886	+7.1%	duplicate	early
Recall	0.760	0.813	+7.0%	jet	late
AP	0.704	0.768	+9.1%	jet	late
mAP	0.527	0.594	+12.7%	hha	late

Apart from the precision metric, the late fusion strategy resulted in the best performing models on all other metrics, namely recall, AP and mAP. Jet colormap encoding with late fusion had improvements of 7.0% and 9.1% over the base model on the recall and AP metrics, respectively.

While the model that used HHA encoding with late fusion had an improvement of 12.7% over the base model on the mAP metric.

mAP provides a measure of the bounding-box detection accuracy compared to the ground truth bounding box. Accurately detecting the bounding box is crucial, as the bounding box can be used to determine the target object, as discussed in Section 2.3.2. Therefore, an improvement of 12.7% over the base model for the mAP metric on a dataset of novel objects has the most significant practical implications.

Table 4.5: Improvement reported due to the fusion of depth data on the task of grasp-type detection by other studies compared with the current method of this thesis.

	Metric	Before depth	After depth	Percent change
Shi et al. [25]	Accuracy	85.43%	93.91%	+9.9%
Luke et al. [47]	Accuracy	95.50%	95.90%	+0.4%
Current method	mAP	52.70%	59.40%	**+12.7%**

Shi et al [25] and Luke et al [47] reported that the inclusion of depth data improved model performance on the task of grasp-type detection. The improvements are displayed in Table 4.5. The current method of this thesis achieved the highest improvement in model performance before and after the fusion of depth data. This is despite the fact that both Shi et al. and Luke et al. acquired depth information through dedicated hardware, which was not needed by the current method. However, this comparison cannot be considered fair, due to a lack of standardized benchmark for the task of grasp-type detection. The improvements reported by Shi et al. was on a dataset that was not entirely comprised of novel objects, as they also include different viewpoints of the same object in their training and test datasets. Luke et al. only reported improvements on the validation dataset, and in fact, report no improvements on the test dataset that contained novel objects. Furthermore, both Shi et al. and Luke et al. reported the *accuracy* metric as their dataset only has one object per image, and the entire image was classified into a grasp type (with no bounding box prediction), making the task simpler, but impractical in real-world scenarios. The lack of a challenging dataset and no bounding box predictions makes the accuracy metric in Shi et al. and Luke et al. much easier to score high on, compared to the mAP metric in this thesis.

Chapter 5
Conclusion and Future Work

5.1 Conclusion

The primary aim of this thesis research was to determine whether the fusion of RGB data with estimated depth data can improve the generalization capabilities of vision-based grasp-type detection models. The primary aim was achieved by meeting the objectives of this thesis:

Objective 1: Develop a new dataset that can be used to evaluate model performance for the task of grasp-type detection on novel objects in real world scenarios.

A dataset of 6,729 RGB images was developed, where images were taken from a viewpoint simulating the perspective of a camera attached to a prosthetic hand. The images contained objects in cluttered real-world scenarios, where each object was identified using a bounding box and assigned a grasp type label of either neutral wrist palmar grasp or pronated wrist precision grasp. In total, over 18,000 bounding boxes were annotated for 53 unique objects encountered during activities of daily living. The dataset was split into train (4.6 K images), validation (1.2 K images), and test set (1 K images), where the train and validation datasets had different images from the same scenes, while the test set consisted of images of novel objects that were from different scenes than the train and validation sets. Ensuring that the test set has novel objects allowed the evaluation of models in real world scenarios and comparison of the generalization capabilities of different models. Different augmentation techniques were employed such as training with 2×2 mosaics, randomly translating, scaling, and flipping the images as well as modifying their HSV values.

Objective 2: Develop a grasp type detection model that can fuse RGB data with depth data using different strategies.

YOLOv5 v6.0 was used as the base model to detect bounding boxes of objects in the image and classify their grasp type. MiDaS was used as the depth map estimator and ResNet18 was used as the depth feature extractor. Three novel fusion strategies were developed, where fused feature maps were obtained by channel-wise concatenation of the depth feature maps with the RGB feature maps. In early fusion, the depth feature maps were fused with the RGB feature maps near the bottom of the neck. In intermediate fusion, the fusion took place near the top of the neck. In late fusion, the fusion took place as the RGB feature maps exited the neck and entered the classification head.

Objective 3: Evaluate different strategies to fuse RGB data with depth data. Determine which strategy is optimal for model performance on novel objects.

Early fusion gave the best performance on the precision metric with a value of 0.886 and an improvement of 7.1% over the base model. Late fusion gaves the best performance on all the other metrics. On the recall metric, late fusion had the best value of 0.813, which was an improvement of 7% over the base model. On the AP metric, late fusion had the best value of 0.768, which was an improvement of 9.1% over the base model. Lastly, late fusion had the best value of 0.594 on the mAP metric, which was a 12.7% improvement over the base model. Moreover, all models with late fusion outperformed the base model, which was not true for early or intermediate fusion. Therefore, it can be concluded that late fusion is the optimal strategy for fusing RGB data with depth data. It is speculated that keeping the RGB and depth feature extraction pipelines separate allows better preservation of the class separability between different grasp types, enabling the classifier to better distinguish between the different classes.

Objective 4: Determine which method of encoding estimated depth maps gives optimal model performance on novel objects

Four methods of encoding depth maps were evaluated: duplicated depth maps, surface normals, jet colormap, and HHA encoding. The naïve approach of duplicating the single channel depth map into three channels gave the best performance of 0.886 on the precision metric. However, this was achieved using early fusion, which was found to be suboptimal considering the other performance metrics. HHA encoding achieved a comparable performance of 0.881 on the precision metric using late fusion, which was found to be the optimal fusion strategy. Jet colormap encoding was found to have the best performance on the recall and AP metrics, with values of 0.813 and 0.768, respectively. HHA encoding achieved the best performance of 0.594 on the mAP metric. Therefore, both jet colormap and HHA provide good results, but neither can be considered optimal.

Objective 5: Determine whether fusing estimated depth data improves model performance on novel objects. If so, determine the most optimal combination of fusion strategy and encoding method

Fusing estimated depth data with RGB data was found to improve model performance on novel objects. Late fusion of depth data with RGB data was found to be the optimal fusion strategy. A combination of late fusion and HHA resulted in the best performance on the precision and mAP metrics, with values of 0.881 and 0.594, respectively. However, HHA encodings are computationally expensive to generate, which is an important factor when considering that the entire grasp-type selection process must be capable of running in real-time. On the contrary, jet colormaps are computationally inexpensive, which makes them a better suited for real-time applications. Late fusion combined with jet colormap encoding had the best performance on the recall and AP metrics, with values of 0.813 and 0.768, respectively. However, late fusion combined with jet colormap encoding also had the worst precision and mAP amongst all encoding types when considering models with late fusion (see Table 4.3 under Test Set). In conclusion, when accurate bounding-box predictions are desired, the combination of late fusion with HHA encoding is recommended, since this combination has the best performance on the mAP metric, which is a measure of the proximity of the detected bounding box to the ground-truth bounding box. When low computational resources are available and performance can be compromised, then the combination of late fusion with jet colormap encoding is recommended. A trade-off between performance and computational cost needs to be made to determine the most optimal combination.

5.2 Future Work

A dataset containing objects with only two possible grasp types was sufficient to meet the objectives of this thesis; however, electric-powered prosthetic hands are capable of more grasp types. The methods described in this thesis are not restricted to the grasp types chosen for the construction of the dataset. In fact, the strategies of fusing RGB and depth data along with the methods of encoding depth data are independent of the number of grasp types. Therefore, any number of grasp types can be detected by modifying the classification head of the grasp type detection DCNN. Given that the set of grasp types achievable by a prosthetic hand is known, a larger dataset can be developed consisting of a wide variety of objects that can be grasped using the set of grasp types. The grasp type detection DCNN can then be trained on the new dataset. Experiments need to be performed to determine how well the proposed method scales to a larger number of grasp types.

While tests on a GeForce RTX 3080 GPU show that the combined DCNN pipeline, which includes the MiDaSv3.0-large depth estimator, YOLOv5-large object detector and ResNet18 depth feature extractor, provides real-time performance with a combined latency of approximately 30 ms, further tests need to be conducted on a prosthetic hand with an amputee to verify the real-time applicability of the proposed method. Such tests can be performed with two potential setups. The first involves the prosthetic device communicating with a server through a wireless protocol, such as WiFi. The DCNN would be preloaded on a high-performance GPU on the server, and upon request from the prosthetic hand, the server would return the predictions made by the model, including grasp types and bounding boxes of objects in the scene. In this case, the prosthetic hand would require minimal electronics, only that necessary to acquire RGB data, communicate with the server and actuate the hand. This would make the prosthetic hand lightweight and power-efficient; however, it would make it dependent on a reliable connection with the server. The second approach to testing the real time applicability of the proposed method involves performing model predictions on the prosthetic device itself. This would require dedicated hardware, which would increase the weight and power consumption of the prosthetic hand. On the positive side, the device would not need to stay connected to a server. With current advances in dedicated embedded systems for DCNNs, this could be feasible with minimum tradeoffs. Furthermore, model quantization and pruning techniques can be employed on the proposed pipeline, to compress the model, resulting in a much smaller memory footprint and lower latencies, allowing for better performance on embedded systems.

Further experiments are needed to understand why handcrafted depth encoding methods (e.g., HHA) are able to provide performance gains. The optimization strategy used to train the grasp-type selection CNN should allow the best set of parameters to be learnt for a given task with enough training, nullifying any initial gains provided by a handcrafted method. Experiments need to be conducted by fine tuning the hyperparameters used for training the model for each encoding method, to ensure that the models are not getting stuck in local minima. Furthermore, the source of the improvement in performance of the grasp-type detection DCNN after RGB data was fused with depth data is not clear. The depth maps were estimated from the RGB images, so the information was already present in the RGB data. Therefore, it is worth investigating knowledge distillation techniques to train a single model using the DCNN pipeline developed in this thesis and extract necessary features directly from RGB images without the need for a separate depth-feature extraction pipeline.

References

[1] “Midwest ProCAD (Medford, WI).” https://www.midwestprocad.com/ (accessed Jan. 24, 2022).

[2] “TRS Inc (Boulder, CO).” https://www.trsprosthetics.com/ (accessed Jan. 24, 2022).

[3] L. Resnik, S. Ekerholm, M. Borgia, and M. A. Clark, “A national study of Veterans with major upper limb amputation: Survey methods, participants, and summary findings,” *PLOS ONE*, vol. 14, no. 3, p. e0213578, Mar. 2019.

[4] J. Z. Zheng, S. de La Rosa, and A. M. Dollar, “An investigation of grasp type and frequency in daily household and machine shop tasks,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 4169–4175.

[5] G. C. Matrone, C. Cipriani, E. L. Secco, G. Magenes, and M. C. Carrozza, “Principal components analysis based control of a multi-dof underactuated prosthetic hand,” *Journal of NeuroEngineering and Rehabilitation*, vol. 7, no. 1, p. 16, Dec. 2010.

[6] B. Massa, S. Roccella, M. C. Carrozza, and P. Dario, “Design and development of an underactuated prosthetic hand,” in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, vol. 4, pp. 3374–3379.

[7] V. L. Winslow, *Classic Human Anatomy in Motion: The Artist’s Guide to the Dynamics of Figure Drawing.* Watson-Guptill, 2015.

[8] *Open Bionics Hero Arm User Guide*. Accessed: Jan. 24, 2022. [Online]. Available: https://openbionics.com/hero-arm-user-guide/.

[9] *Ottobock Bebionic User Guide*. Accessed: Jan. 24, 2022. [Online]. Available: https://www.ottobockus.com/media/local-media/prosthetics/upper-limb/files/14112_bebionic_user_guide_lo.pdf.

[10] *Mobius Bionics Luke Arm System Specification Sheet*. Accessed: Jan. 24, 2022. [Online]. Available: https://www.mobiusbionics.com/wp-content/uploads/2019/09/Mobius-Bionics-LUKE-Product-Spec-Sheet.pdf.

[11] *Ossur i-limb hand Clinician Manual*. Accessed: Jan. 24, 2022. [Online]. Available: https://assets.ossur.com/library/41042/i-Limb%20Access%20Other.pdf.

[12] Touch Bionics, “my i-limb^{TM} App: Quick Reference Guide for i-limb^{TM} quantum.” Accessed: May 01, 2022. [Online]. Available: https://irp-cdn.multiscreensite.com/acf635e2/files/uploaded/MY%20ILIMB%20APP%20MANUAL.pdf.

[13] H. Lindner, W. Hill, L. Norling Hermansson, and A. J. Lilienthal, “Cognitive load in learning to use a multi-function hand.,” *In MEC20, Fredericton, New Brunswick, Canada (Symposium canceled). University of New Brunswick*. 2020.

[14] L. C. Smail, C. Neal, C. Wilkins, and T. L. Packham, “Comfort and function remain key factors in upper limb prosthetic abandonment: findings of a scoping review,” *Disability and Rehabilitation: Assistive Technology*, vol. 16, no. 8, pp. 821–830, 2021.

[15] Y. W. Liu, F. Feng, and Y. F. Gao, “HIT prosthetic hand based on tendon-driven mechanism,” *J Cent South Univ*, vol. 21, no. 5, pp. 1778–1791, 2014.

[16] J. T. Belter, J. L. Segil, A. M. Dollar, and R. F. Weir, “Mechanical design and performance specifications of anthropomorphic prosthetic hands: A review,” *Journal of Rehabilitation Research and Development*, vol. 50, no. 5, pp. 599–618, 2013.

[17] A. D. Vigotsky, I. Halperin, G. J. Lehman, G. S. Trajano, and T. M. Vieira, “Interpreting Signal Amplitudes in Surface Electromyography Studies in Sport and Rehabilitation Sciences,” *Frontiers in Physiology*, vol. 8, Jan. 2018.

[18] C. M. Light, P. H. Chappell, B. Hudgins, and K. Engelhart, "Intelligent multifunction myoelectric control of hand prostheses," *Journal of Medical Engineering and Technology*, vol. 26, no. 4. pp. 139–146, Jul. 2002.
[19] O. W. Samuel *et al.*, "Intelligent EMG Pattern Recognition Control Method for Upper-Limb Multifunctional Prostheses: Advances, Current Challenges, and Future Prospects," *IEEE Access*, vol. 7, pp. 10150–10165, 2019.
[20] S. Došen, C. Cipriani, M. Kostić, M. Controzzi, M. C. Carrozza, and D. B. Popović, "Cognitive vision system for control of dexterous prosthetic hands: Experimental evaluation," *Journal of NeuroEngineering and Rehabilitation*, vol. 7, no. 1, p. 42, Dec. 2010.
[21] D. P. McMullen *et al.*, "Demonstration of a semi-autonomous hybrid brain-machine interface using human intracranial EEG, eye tracking, and computer vision to control a robotic upper limb prosthetic," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 784–796, 2014.
[22] Y. He, R. Kubozono, O. Fukuda, N. Yamaguchi, and H. Okumura, "Vision-Based Assistance for Myoelectric Hand Control," *IEEE Access*, vol. 8, pp. 201956–201965, 2020.
[23] B. Zhong, H. Huang, and E. Lobaton, "Reliable Vision-Based Grasping Target Recognition for Upper Limb Prostheses," *IEEE Transactions on Cybernetics*, vol. 52, no. 3, pp. 1750–1762, Mar. 2022.
[24] Y. He, R. Shima, O. Fukuda, N. Bu, N. Yamaguchi, and H. Okumura, "Development of distributed control system for vision-based myoelectric prosthetic hand," *IEEE Access*, vol. 7, pp. 54542–54549, 2019.
[25] C. Shi, D. Yang, J. Zhao, and H. Liu, "Computer Vision-Based Grasp Pattern Recognition with Application to Myoelectric Control of Dexterous Hand Prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 9, pp. 2090–2099, Sep. 2020.
[26] H. Martin, J. Donaw, R. Kelly, Y. Jung, and J.-H. Kim, "A novel approach of prosthetic arm control using computer vision, biosignals, and motion capture," in *IEEE Symposium on Computational Intelligence in Robotic Rehabilitation and Assistive Technologies (CIR2AT)*, Dec. 2014, pp. 26–30.
[27] C. Shi, L. Qi, D. Yang, J. Zhao, and H. Liu, "A Novel Method of Combining Computer Vision, Eye-Tracking, EMG, and IMU to Control Dexterous Prosthetic Hand," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2019, pp. 2614–2618.
[28] S. Došen and D. B. Popović, "Transradial Prosthesis: Artificial Vision for Control of Prehension," *Artificial Organs*, vol. 35, no. 1, pp. 37–48, 2011.
[29] M. Gardner, R. Woodward, R. Vaidyanathan, E. Burdet, and B. C. Khoo, "An unobtrusive vision system to reduce the cognitive burden of hand prosthesis control," in *13th International Conference on Control Automation Robotics & Vision (ICARCV)*, Dec. 2014, pp. 1279–1284.
[30] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan, and K. Nazarpour, "Deep learning-based artificial vision for grasp classification in myoelectric hands," *Journal of Neural Engineering*, vol. 14, no. 3, May 2017.
[31] D. Andrade, A. Ishikawa, A. Munoz, and E. Rohmer, "A hybrid approach for the actuation of upper limb prostheses based on computer vision," in *Latin American Robotics Symposium (LARS) and Brazilian Symposium on Robotics (SBR)*, Nov. 2017, pp. 1–6.
[32] S. Tang, R. Ghosh, N. Thakor, and S. Kukreja, "Orientation estimation and grasp type detection of household objects for upper limb prostheses with dynamic vision sensor," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2016, pp. 99–102.

[33] M. Markovic, S. Dosen, D. Popovic, B. Graimann, and D. Farina, “Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis,” *Journal of Neural Engineering*, vol. 12, no. 6, Nov. 2015.

[34] A. Eitel, J. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, “Multimodal deep learning for robust RGB-D object recognition,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 681–687.

[35] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, “Learning Rich Features from RGB-D Images for Object Detection and Segmentation,” in *European conference on computer vision. Springer, Cham*, 2014, pp. 345–360.

[36] A. Aakerberg, K. Nasrollahi, C. B. Rasmussen, and T. B. Moeslund, “Depth value pre-processing for accurate transfer learning based RGB-D object recognition,” in *IJCCI Proceedings of the 9th International Joint Conference on Computational Intelligence*, 2017, pp. 121–128.

[37] T. Feix, J. Romero, H. B. Schmiedmayer, A. M. Dollar, and D. Kragic, “The GRASP Taxonomy of Human Grasp Types,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 66–77, Feb. 2016.

[38] O. van der Niet Otr, H. Reinders-Messelink, H. Bouwsema, R. Bongers, and C. van der Sluis, “The i-limb pulse hand compared to the i-limb and DMC plus hand,” in *Myoelectric Controls/Powered Prosthetics Symposium (MEC)*, 2011, pp. 260–261.

[39] J. Ribeiro *et al.*, “Analysis of Man-Machine Interfaces in Upper-Limb Prosthesis: A Review,” *Robotics*, vol. 8, no. 1, p. 16, Feb. 2019.

[40] R. Merletti and P. Parker, *Electromyography: Physiology, Engineering, and Noninvasive Applications*, vol. 11. John Wiley & Sons., 2004.

[41] O. W. Samuel *et al.*, “Resolving the adverse impact of mobility on myoelectric pattern recognition in upper-limb multifunctional prostheses,” *Computers in Biology and Medicine*, vol. 90, pp. 76–87, Nov. 2017.

[42] L. Hargrove, K. Englehart, and B. Hudgins, “The effect of electrode displacements on pattern recognition based myoelectric control,” in *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, 2006, pp. 2203–2206.

[43] S. M. Engdahl, B. P. Christie, B. Kelly, A. Davis, C. A. Chestek, and D. H. Gates, “Surveying the interest of individuals with upper limb loss in novel prosthetic control techniques,” *Journal of NeuroEngineering and Rehabilitation*, vol. 12, no. 1, Jun. 2015.

[44] N. Bu, Y. Bandou, O. Fukuda, H. Okumura, and K. Arai, “A semi-automatic control method for myoelectric prosthetic hand based on image information of objects,” in *International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Nov. 2017, pp. 23–28.

[45] J. DeGol, A. Akhtar, B. Manja, and T. Bretl, “Automatic grasp selection using a camera in a hand prosthesis,” in *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug. 2016, pp. 431–434.

[46] G. Morgan, K. Nazarpour, P. Degenaar, A. Alameer, and G. Ghazaei, “An exploratory study on the use of convolutional neural networks for object grasp classification,” in *2nd IET International Conference on Intelligent Signal Processing (ISP)*, 2015, pp. 1–5.

[47] L. Taverne, M. Cognolato, T. Butzer, R. Gassert, and O. Hilliges, “Video-based Prediction of Hand-grasp Preshaping with Application to Prosthesis Control,” in *International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 4975–4982.

[48] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Advances in neural information processing systems*, vol. 25. 2012.

[49] V. Nair and G. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines,” *in International Conference on Machine Learning*. 2010.

[50] M. Z. Alom *et al.*, “A State-of-the-Art Survey on Deep Learning Theory and Architectures,” *Electronics (Basel)*, vol. 8, no. 3, p. 292, Mar. 2019.

[51] H. Ramchoun, M. Amine, J. Idrissi, Y. Ghanou, and M. Ettaouil, “Multilayer Perceptron: Architecture Optimization and Training,” *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 1, p. 26, 2016.

[52] Shun-ichi Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomputing*, vol. 5, no. 4–5, pp. 185–196, 1993.

[53] 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 (CVPR)*, 2016, pp. 770–778.

[54] C. Shi, D. Yang, J. Zhao, and H. Liu, “Computer Vision-Based Grasp Pattern Recognition with Application to Myoelectric Control of Dexterous Hand Prosthesis,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 9, pp. 2090–2099, Sep. 2020.

[55] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2016, pp. 2818–2826.

[56] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[57] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[58] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” in *ECCV Lecture Notes in Computer Science, vol 8693. Springer, Cham.*, 2014, pp. 740–755.

[59] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *Journal of Big Data*, vol. 6, no. 1, Dec. 2019.

[60] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934.*, Apr. 2020 [Online]. Available: http://arxiv.org/abs/2004.10934.

[61] S. Bozinovski, “Reminder of the first paper on transfer learning in neural networks, 1976,” *Informatica (Slovenia)*, vol. 44, no. 3, pp. 291–302, Sep. 2020.

[62] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587.

[63] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788.

[64] G. Jocher, “ultralytics/yolov5: v6.0 - YOLOv5n ‘Nano’ models, Roboflow integration, TensorFlow export, OpenCV DNN support.” Zenodo, Oct. 12, 2021.

[65] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767.*, Apr. 2018 [Online]. Available: http://arxiv.org/abs/1804.02767.

[66] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12993–13000, Apr. 2020.

[67] M. Strbac, M. Markovic, and D. B. Popovic, “Kinect in neurorehabilitation: Computer vision system for real time hand and object detection and distance estimation,” in *11th Symposium on Neural Network Applications in Electrical Engineering*, Sep. 2012, pp. 127–132.

[68] M. Markovic, S. Dosen, C. Cipriani, D. Popovic, and D. Farina, “Stereovision and augmented reality for closed-loop control of grasping in hand prostheses,” *Journal of Neural Engineering*, vol. 11, no. 4, Aug. 2014.

[69] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper Depth Prediction with Fully Convolutional Residual Networks,” in *Fourth International Conference on 3D Vision (3DV)*, Oct. 2016, pp. 239–248.

[70] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep Ordinal Regression Network for Monocular Depth Estimation,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2002–2011.

[71] F. Liu, Chunhua Shen, and Guosheng Lin, “Deep convolutional neural fields for depth estimation from a single image,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 5162–5170.

[72] R. Li, K. Xian, C. Shen, Z. Cao, H. Lu, and L. Hang, “Deep Attention-Based Classification Network for Robust Depth Prediction,” in *In Asian Conference on Computer Vision Springer, Cham*, 2018, pp. 663–678.

[73] A. Roy and S. Todorovic, “Monocular Depth Estimation Using Neural Regression Forest,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5506–5514.

[74] D. Eigen, C. Puhrsch, and R. Fergus, “Depth Map Prediction from a Single Image using a Multi-Scale Deep Network,” *Advances in neural information processing systems.*, vol. 27. 2014.

[75] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, Mar. 2022.

[76] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision Transformers for Dense Prediction,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 12159–12168.

[77] K. Xian *et al.*, “Monocular Relative Depth Perception with Web Stereo Data Supervision,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 311–320.

[78] J. Cho, D. Min, Y. Kim, and K. Sohn, “DIML/CVL RGB-D Dataset: 2M RGB-D Images of Natural Indoor and Outdoor Scenes,” *arXiv preprint arXiv:2110.11590.*, Oct. 2021 [Online]. Available: http://arxiv.org/abs/2110.11590.

[79] Z. Li and N. Snavely, “MegaDepth: Learning Single-View Depth Prediction from Internet Photos,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2041–2050.

[80] C. Wang, S. Lucey, F. Perazzi, and O. Wang, “Web Stereo Video Supervision for Depth Prediction from Dynamic Scenes,” in *International Conference on 3D Vision (3DV)*, Sep. 2019, pp. 348–357.

[81] W. Wang *et al.*, “TartanAir: A Dataset to Push the Limits of Visual SLAM,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 4909–4916.

[82] K. Xian, J. Zhang, O. Wang, L. Mai, Z. Lin, and Z. Cao, "Structure-Guided Ranking Loss for Single Image Depth Prediction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 608–617.

[83] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape Open Dataset for Autonomous Driving and Its Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2702–2719, Oct. 2020.

[84] Y. Yao *et al.*, "BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 1787–1796.

[85] Q. Wang, S. Zheng, Q. Yan, F. Deng, K. Zhao, and X. Chu, "IRS: A Large Naturalistic Indoor Robotics Stereo Dataset to Train Deep Models for Disparity and Surface Normal Estimation," in *IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2021, pp. 1–6.

[86] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893.

[87] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," in *International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242.

[88] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5987–5995.

[89] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2020, pp. 1571–1580.

[90] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546.* May 01, 2019. [Online]. Available: http://arxiv.org/abs/1905.00546.

[91] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *International Conference on Machine Learning PMLR*, May 2019, pp. 6105–6114.

Appendix
Depth Feature Extractor Selection

This appendix presents the results of experiments conducted to evaluate which deep convolutional neural-network (DCNN) architecture would be best suited as a depth feature extractor. All experiments were performed assuming late fusion and repeated for the different forms of depth encoding, namely duplicated depth channels, surface normal, jet colormap, and HHA. See Section 3.5.3 for details on late fusion and Section 3.4.2 for details on depth encoding. YOLOv5 v6.0 was used as the object detector. The AP and mAP metrics are reported on the validation and test datasets. All experimental procedures are the same as those reported in Section 3.6. Models were pretrained on the ImageNet [57] dataset.

Table A.1 and Table A.2 display results for using the 18-layered and 34-layered versions of the Residual Network [53] as depth feature extractor, respectively. Table A.3 shows the results for using the Inception v3 [55] as the depth feature extractor. The results in Tables A.4 and A.5 were obtained using a 50-layered ResNext [88]. The ResNext used to obtain results in Table A.4 was implemented using cross stage partial network (CSPNet) [89], while the results in Table A.5 were obtained by pre-training the ResNext on ImageNet as well as one-billion unlabeled images using a semi-supervised technique [90]. Table A.6 presents results for using EfficientNet-B3 [91] as the depth feature extractor.

The primary criteria for the candidacy of the above-mentioned architectures were their size, in terms of number of parameters. The chosen depth feature extractors have approximately 12 million to 25 million parameters. In comparison, the YOLOv5 v6.0 model, used as the object detector, has 46.5 million parameters. Considering that YOLOv5 v6.0 needs to extract features from 3-channel RGB data while the depth feature extractor needs to extract features from single channel depth maps, the ratio of depth feature extractor parameters to RGB feature extractor parameters is justified. Moreover, smaller models help minimize computational cost, which is essential for real time performance.

The results show that on the validation set, ResNet18 and Inception v3 had the highest AP value of 0.986 (Table A.3), while ResNet18 (Table A.1) had the best mAP of 0.893. On the test dataset, ResNet18 (Table A.1) outperformed all other depth feature extractors, achieving the best performance on the AP and mAP metric with values of 0.768 and 0.594 respectively.

Based on the empirical evidence presented in this appendix, ResNet18 was selected as the depth feature extractor of choice for all other experiments in this thesis. While this is not an exhaustive search, the results prove that the choice of depth feature extractor is important.

Table A.1: Results for using 18-layered Residual Network [53] as depth feature extractor

	YOLOv5 v6.0 + ResNet18			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	0.983	0.892	0.758	0.585
Surface Normals	**0.986**	**0.893**	0.757	0.580
Jet Colormap	0.983	0.892	**0.768**	0.570
HHA	0.985	**0.893**	0.766	**0.594**

Table A.2: Results for using 34-layered Residual Network [53] as depth feature extractor

	YOLOv5 v6.0 + ResNet34			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	**0.983**	**0.892**	0.681	0.486
Surface Normals	0.982	**0.892**	0.726	0.519
Jet Colormap	0.980	0.889	0.708	0.510
HHA	0.982	0.889	**0.732**	**0.524**

Table A.3: Results for using Inception v3 [55] as depth feature extractor

	YOLOv5 v6.0 + Inceptionv3			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	**0.986**	0.887	**0.673**	**0.519**
Surface Normals	0.980	0.885	0.666	0.500
Jet Colormap	0.982	**0.888**	0.666	0.505
HHA	0.981	**0.888**	0.667	0.501

Table A.4: Results for using 50-layered cross stage partial network (CSPNet) ResNext [89] as depth feature extractor

	YOLOv5 v6.0 + CSP ResNext50			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	0.971	0.870	0.709	0.536
Surface Normals	0.971	0.864	**0.717**	0.527
Jet Colormap	0.971	0.864	0.696	0.532
HHA	**0.977**	**0.874**	0.708	**0.539**

Table A.5: Results for using 50-layered ResNext [88] with a backbone trained on one billion unlabeled images [90] as depth feature extractor

	YOLOv5 v6.0 + SWSL ResNext50			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	**0.985**	0.889	0.691	0.517
Surface Normals	0.983	0.889	0.731	0.539
Jet Colormap	**0.985**	**0.892**	**0.745**	**0.554**
HHA	0.984	**0.892**	0.722	0.536

Table A.6: Results for using EfficientNet-B3 [91] as depth feature extractor

	YOLOv5 v6.0 + EfficientNet-B3			
Depth Encoding	**Validation Set**		**Test Set**	
	AP	**mAP**	**AP**	**mAP**
Duplicate	**0.977**	**0.880**	0.665	0.478
Surface Normals	0.954	0.819	0.674	0.479
Jet Colormap	0.975	0.872	0.676	0.503
HHA	0.967	0.871	**0.710**	**0.520**