MCAST

Recognizing the sequential state of knots for educational purposes.

Alvin Cassar

Supervisor: Frankie Inguanez

June - 2023

A dissertation submitted to the Institute of Information and Communication Technology in partial fulfilment of the requirements for the degree of BSc (Hons) in Software Development

Authorship Statement

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr. Frankie Inguanez.

15/09/2023

Date

Alvin Cassar

Signature

Copyright Statement

In submitting this dissertation to the MCAST Institute of Information and Communication Technology, I understand that I am giving permission for it to be made available for use in accordance with the regulations of MCAST and the Library and Learning Resource Centre. I accept that my dissertation may be made publicly available at MCAST's discretion.

15/09/2023

Date

Alvin Cassar

Signature

Abstract

Knots play a pivotal role in a wide range of applications and are integral to ensuring safety in various activities. This research paper focuses on fundamental knots, aiming to categorize the step-by-step procedures required for their construction. The primary objective of this study was to develop a robust object detection model capable of predicting the state of a rope during knot tying in real-time. The model's robustness across different environmental variables was also a key consideration, evaluated through multiple experiments.

In pursuit of these objectives, the YOLO algorithm was employed to detect sequential knot steps within a controlled environment. The reef knot served as the foundational knot for the initial model setup, followed by the construction of a model for the more intricate bowline knot, allowing for a robustness comparison.

The study undertook the challenging task of constructing a custom dataset, given the absence of readily available datasets for this specialized domain. The reef knot model dataset consisted of 2,800 images, while the bowline knot model dataset comprised 2,900 images, categorically divided into five distinct classes representing each knot step. The dataset was further split to facilitate model development.

A proof-of-concept prototype was developed to demonstrate the feasibility of real-time knot step recognition. The Reef Knot model achieved an impressive mAP of 98.8%, while the Bowline model excelled with an mAP of 99.4%. To further

scrutinize the models and evaluate their robustness, manual testing was conducted under ten diverse environmental scenarios. Results indicated that the Reef Knot model achieved an accuracy rate of 68%, while the Bowline model demonstrated an average correct predictability rate of 50%. Notably, real-time testing is anticipated to enhance accuracy significantly due to increased frames per second and computational capacity for step prediction.

With further research and development, these models and algorithms could be improved by developing upon the proof-of-concept program, and also increasing the dataset sizes to facilitate for different environments.

Keywords: Object Detection, Machine Learning, Knots, Ropes, State Recognition.

Acknowledgements

I extend my sincere thanks to my mentor, Mr. Frankie Inguanez, for his invaluable guidance in making this research possible, with his great knowledge in this field of study. I'm also grateful to my family, girlfriend, and friends for their unwavering support.

Table of Contents

Authorship Statement i

Copyright Statement ii

Abstract iii

Acknowledgements v

Lists of Figures viii

Lists of Tables ix

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation of the Study . . . 1
- 1.3 Purpose of the Study . . . 2
- 1.4 Hypothesis and Research Questions . . . 2
- 1.5 Document Structure . . . 3

2 Literature Review 4
- 2.1 Introduction . . . 4
- 2.2 Maching Learning Algorithms . . . 5
  - 2.2.1 Supervised Learning . . . 6
  - 2.2.2 Unsupervised Learning . . . 8
  - 2.2.3 Semi-Supervised Learning . . . 10
- 2.3 Studies Detecting Rope for Robotic Programs . . . 11
  - 2.3.1 Research 1 - Vision based Topological State Recognition . 11
  - 2.3.2 Research 2 - Learning to Untangle Ropes with RGB-D Perception . . . 12
  - 2.3.3 Research 3 - Untangling Dense Non-Planar Knots . . . 13
- 2.4 Building a Custom Dataset . . . 14
  - 2.4.1 Research 4 - Mapping for Automated Edge-Based Pallet Racking Inspections . . . 14
  - 2.4.2 Research 5 - Autonomous Pallet Racking Inspection System - Mobilenetv2 . . . 15
- 2.5 Evaluating Results . . . 17
  - 2.5.1 Research 6 - Automatic Fabric Defect Detection . . . 17

3 Research Methodology **19**
3.1 Data Procurement 19
3.1.1 Dataset Collection 21
3.1.2 Dataset Pre-Processing 22
3.2 Model Setup and Training 23
3.2.1 Configuring YOLO Model 24
3.2.2 Training 26
3.3 Evaluation of Results 28
3.3.1 Proof-of-Concept Prototype 30

4 Analysis of Results and Discussion **32**
4.1 Introduction 32
4.2 Knot Detection Experiments 32
4.2.1 Metrics Before and After Augmentation 32
4.2.2 Experimenting Model Training for Different Knot 34
4.3 Comparing Models Manually 37
4.3.1 Setting up Environments for Testing 37
4.3.2 Evaluating Results from Manual Testing 41
4.4 Real-Time Prototype Results 49

5 Conclusions and Recommendations **50**
5.1 Limitations 51
5.2 Suggestions for Further Research and Improvement 52

References **53**

Appendix A Video to Images Converter **55**

List of Figures

2.1 Supervised Learning [1] 7
2.2 Unsupervised Learning [1] 9
2.3 Semi-Supervised Learning [1] 10
2.4 Labelling and Backgrounds [2] 12
2.5 Rope Segmenting [3] 13
2.6 Rope Segmenting [4] 13
2.7 Racking Labelling [5] 14

3.1 Prototype Pipeline 20
3.2 Reef Knot Steps 22
3.3 YoloV7 Charts 25
3.4 Sample of Ground Truth Images Compared to a Sample of Predictions during Training 27
3.5 Prototype Design 31

4.1 F1 Score Comparing the difference of datasets 34
4.2 Bowline Steps 35
4.3 Environment 1 Example 40
4.4 Environment 2 Example 40
4.5 Environment 3 Example 40
4.6 Environment 4 Example 40
4.7 Environment 5 Example 40
4.8 Environment 6 Example 40
4.9 Environment 7 Example 41
4.10 Environment 8 Example 41
4.11 Environment 9 Example 41
4.12 Environment 10 Example 41
4.13 Prototype Sample Image 49

List of Tables

2.1 A table comparing research on warehouses 16

3.1 Reef Knot Dataset Distribution 23
3.2 YoloV5 Models Training Evaluation 26
3.3 Reef Knot Dataset after Augmentation 28
3.4 Example of a simple Confusion Matrix 28

4.1 Reef Knot Results before Augmentation 33
4.2 Reef Knot Results after Augmentation 33
4.3 Bowline Knot Dataset after Augmentation 36
4.4 Bowline Knot Results 37
4.5 Manual Experiments for Reef Knot 45
4.6 Manual Experiments for Bowline 48

Chapter 1

Introduction

1.1 Background

Knots have many applications and find utility among individuals of various backgrounds, spanning different age groups and skill levels. In scenarios entailing potentially risky activities, it becomes important to ensure that precise and suitable knot-tying techniques are applied to guarantee safety. This research paper is uniquely focused on basic knots, with the primary objective of categorizing the sequential steps required to construct these knots. By placing a specific emphasis on these elemental steps, the study aspires to bolster safety protocols for individuals who may possess limited expertise in knotting techniques, ultimately enhancing their safety in everyday pursuits and undertakings.

1.2 Motivation of the Study

My personal motivation for this research project derives from my prior experience in training an algorithm to recognize different knots, a project undertaken during my second year of study. This initial exposure ignited my fascination with knot detection and recognition. While resources like animated visuals and guides exist

for learning knot-tying, none provide visual assessment and feedback on the tying process. This study aims to address this void by introducing a novel approach that leverages technology for improved knot-tying instruction and proficiency.

Furthermore, my motivation draws from over a decade of involvement in scouting, where I've witnessed the pivotal role of knot-tying skills in practical and safety contexts. I envision this research as an opportunity to innovate knot-tying education, offering a fresh perspective that has yet to be explored.

1.3 Purpose of the Study

The purpose of this study is to have a well-trained object detection model that can predict the state of a rope while doing a knot. It is important that the model created can support real-time predictions to have a better understanding of the results while a knot is actually being done, and not by images only. Having the robustness of the model be accurate in different environmental variables is also important for this research. The robustness would be evaluated by having multiple experiments done, such as creating a different dataset for a much more complex knot and comparing their results.

1.4 Hypothesis and Research Questions

The hypothesis of this research is that by making use of machine learning it will be possible to recognize the state of the rope while identifying the different phases in tying a knot.

1. What key features and characteristics of the data set are needed for proper

training of a model?

2. What algorithms can be used for the recognition of the different phases in tying a knot?
3. What metrics need to be evaluated to be able to compare the results of a model?

1.5 Document Structure

The document structure starts with the Introduction (Chapter 1) and continues to the Literature review (Chapter 2). This highlights the importance of having a great knotting technique to be as safe as possible at all times. An overview of different types of machine learning algorithms is research. Finally, different research papers were researched that use these algorithms to detect custom objects. In the Methodology (Chapter 3), the prototype pipeline is shown with information on how a dataset was created from scratch, as no public dataset was found online. In this section how the algorithm was chosen was also discussed. The Results and Discussion (Chapter 4) had multiple experiments on how the results of the models were evaluated while also, while some experiments had manual testing. The Conclusion (Chapter 5) summarises the research paper, which also includes the limitations and any suggestions for future improvement and development.

Chapter 2

Literature Review

2.1 Introduction

Knots play a crucial role across a range of disciplines, including sea sports, rock climbing, construction, and the medical sector. Understanding the significance of knots in these fields is essential for ensuring safety, efficiency, and functionality. In sea sports, knots are vital for securing equipment, sails, and lines, while in rock climbing, they are instrumental in anchoring ropes and providing stability. The construction industry relies on knots for securing materials, lifting heavy objects, and creating sturdy structures. Even in the medical sector, knots are utilized for surgical procedures and wound closure. With such widespread importance, there is a wealth of educational content available on knot-tying techniques and their applications.

As stated by [6], [7], and [8], technology has changed the way people learn. Making it faster and more engaging for students while enabling them to learn at their own pace. A part from this technology also allows people to broaden their general knowledge and learn more about any topics that might interest them even if they are not entirely academic.

Furthermore, the emergence of computer vision technology has opened up new

avenues for knot-making and recognition. The use of computer vision is steadily increasing and is being used in many different scenarios. This integration of computer vision and knot-making demonstrates the evolving nature of this age-old practice and its potential for innovation in various fields.

2.2 Maching Learning Algorithms

As humans gaze upon an image, they have the ability to swiftly understand the objects within it, and pinpoint the locations and small details. The human visual system operates with remarkable speed and accuracy, enabling us to engage in complex tasks like driving with seemingly effortless ease. The development of rapid and precise object recognition algorithms holds the potential to empower computers to operate vehicles autonomously without relying on specialized sensors, construct tools capable of delivering real-time environmental data to human users, and unlock the possibilities of versatile, responsive robotic systems.

In recognition systems, classifiers play a huge role in the process of identification. To discern and identify objects, these systems employ dedicated classifiers tailored to each specific object of interest, subjecting them to evaluation across a range of positions and scales within an image. Techniques such as Deformable Parts Models (DPM) adopt a sliding window methodology, consistently applying the classifier throughout the image [9].

Deep learning has emerged as a superior performer in comparison to classical target detection methods. The traditional approach involves the painstaking manual extraction of features, a process that demands years of data collection and expertise within relevant domains. In contrast, deep learning takes a more data-driven and representational approach, learning distinctive features from vast quantities of data. This deep learning model emulates the human brain's visual perception sys-

tem, directly extracting features from the original image and progressively processing them layer by layer to capture the image's high-dimensional information. This innovative approach has witnessed remarkable success within the field of computer vision [10].

The mix of machine learning and computer vision plays an important role when trying to recognize and track human actions and objects with multimedia streams. An array of prediction and analysis methodologies, including will-established techniques like supervised learning, unsupervised learning, and semi-supervised learning, come into play. These methods leverage a diverse range of machine learning algorithms, each with its unique strengths and applications.

2.2.1 Supervised Learning

Supervised learning is a foundational machine learning type characterized by its way of learning and the way the data is labeled. In supervised learning, algorithms learn to make predictions or decisions without human intervention by analyzing examples that consist of both input features and their corresponding target labels. This approach is versatile and is commonly used for classification tasks, where data points are set into discrete classes, and regression tasks, where numerical values are predicted. For instance, a supervised learning algorithm can be trained to classify emails as spam or not spam based on labeled email datasets. The key strength of supervised learning lies in its ability to generalize from known examples to make informed predictions on new, unseen data [1]. Three types of supervised learning are:

- **Region-Based Convolutional Neural Network:** RCNN is an object detection model that combines region proposal and CNN-based features. It first generates region proposals using selective search and then extracts features

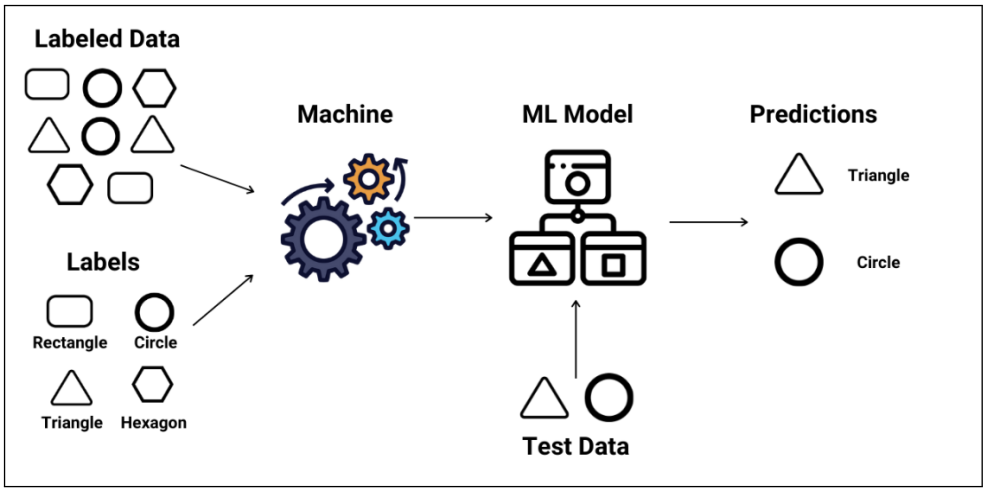

Figure 2.1: Supervised Learning [1]

from each proposed region. These features are used for object classification and bounding box regression.

RCNN's accuracy is commendable, but it is computationally intensive and slow due to the multi-stage process. It made the way for the development of faster object detectors like Fast R-CNN and Faster R-CNN. Despite its relative slowness, RCNN remains valuable in cases where precision is crucial, such as medical image analysis and fine-grained object recognition [11].

- **You Only Look Once:** YOLO is an innovative deep learning-based object detection algorithm. Unlike traditional two-stage object detectors like RCNN, YOLO takes a single pass through the neural network to detect objects in real-time. YOLO divides an image into a grid and predicts bounding boxes and class probabilities for objects within each grid cell. This approach allows YOLO to be extremely fast while maintaining accuracy.

  YOLO's speed and efficiency make it ideal for real-time applications like autonomous vehicles, surveillance, and robotics. It's capable of detecting multiple objects in a single image frame, even when they overlap [11, 12].

- **Support Vector Machine:** SVM is a powerful supervised learning algorithm used for both classification and regression tasks. It works by finding a hy-

perplane that best separates data points of different classes while maximizing the margin between them. In the case of classification, SVM aims to create a decision boundary that effectively separates data points into distinct categories. It's particularly useful when dealing with linearly separable data, but it can be extended to non-linear separations using kernel functions.

SVMs are known for their ability to handle high-dimensional data and are robust against overfitting. They excel in scenarios where the margin between classes is well-defined, making them suitable for tasks such as text classification, image classification, and bioinformatics. SVMs are interpretable and offer a principled way to make predictions based on learned support vectors.

2.2.2 Unsupervised Learning

Unsupervised learning, in contrast, tackles the challenge of making sense of unlabeled data. Here, algorithms are employed to uncover hidden patterns, structures, or representations within the data without the presence of predefined labels. Unsupervised learning tasks frequently include clustering, which groups similar data points together, and dimensionality reduction, which simplifies the data by reducing the number of features while preserving vital information. For instance, businesses can use unsupervised learning to segment their customer base into distinct groups based on shared characteristics, helping them tailor marketing strategies. This approach is particularly valuable when there's a need to explore and understand data without prior knowledge of its inherent structure [1].

- **K-Means Clustering:** K-Means is a versatile and widely used unsupervised clustering algorithm. It's particularly effective for data segmentation and pattern discovery. The algorithm partitions the dataset into 'k' clusters, where 'k' is a user-defined parameter. It does so by assigning data points to the

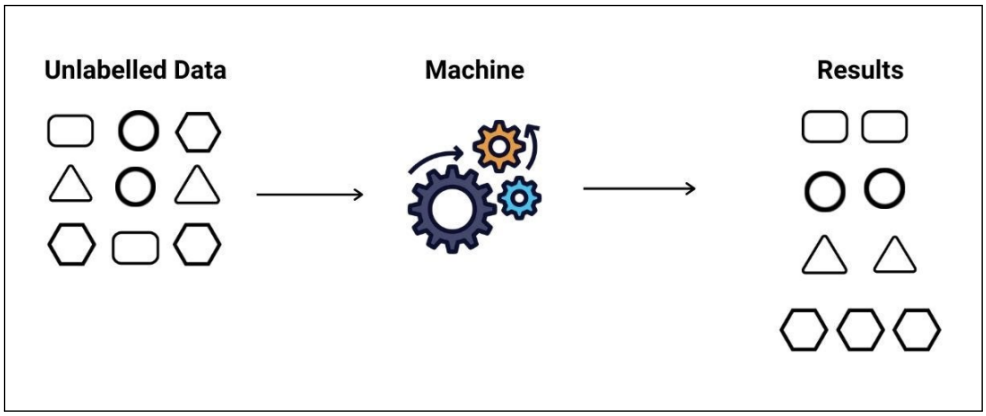

Figure 2.2: Unsupervised Learning [1]

cluster with the nearest mean. K-Means aims to minimize the cluster variance, making each cluster as internally alike as possible while maximizing the inter-cluster variance, ensuring distinct separation [13].

K-Means is employed in various applications, including customer segmentation, image compression, and document categorization. It's relatively efficient and easy to implement, making it a go-to choice for exploratory data analysis and data preprocessing.

- **Principal Component Analysis:** PCA [13] is a fundamental unsupervised dimensionality reduction technique used to extract essential features from high-dimensional data. It operates by identifying the directions (principal components) along which the data varies the most. These components are orthogonal and ranked by the amount of variance they capture. By projecting data onto a lower-dimensional subspace defined by the top principal components, PCA reduces the data's dimensionality while preserving as much information as possible. It is commonly used in fields such as image processing, genetics, and finance to uncover underlying structures in complex datasets.

2.2.3 Semi-Supervised Learning

Semi-supervised learning bridges the gap between supervised and unsupervised learning. In this hybrid approach, algorithms are trained on a dataset containing a small portion of labeled data and a more substantial portion of unlabeled data. By combining the insights derived from both types of data, semi-supervised learning often yields improved generalization and accuracy. A classic example of semi-supervised learning is in speech recognition, where a small set of spoken phrases may be labeled, while a large pool of unlabeled audio data is available. By leveraging both types of data, semi-supervised learning can enhance the accuracy and performance of speech recognition systems, showcasing the power of this approach in leveraging limited labeled data to improve learning outcomes [1].

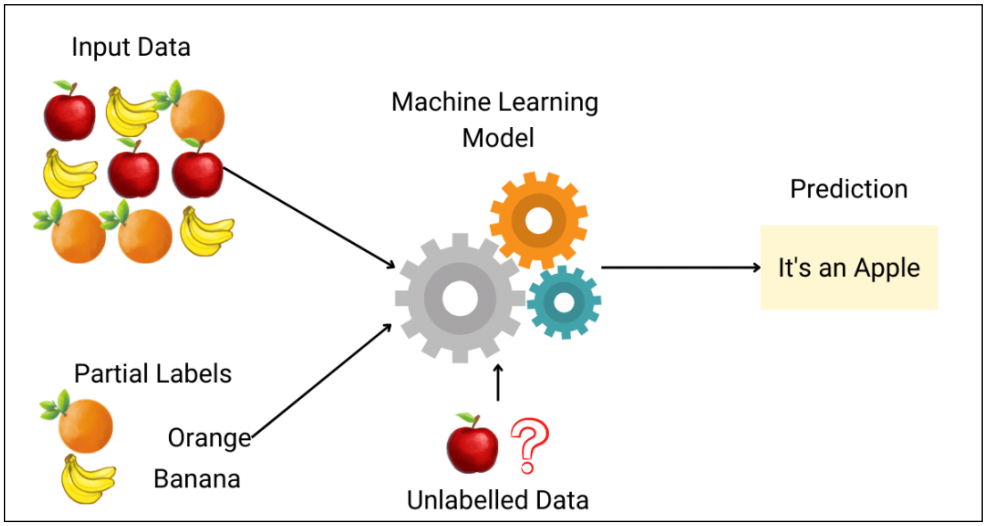

Figure 2.3: Semi-Supervised Learning [1]

- **Label Propagation:** Label Propagation is a straightforward and effective semi-supervised learning algorithm. It starts with a small set of images, all having labeled data points, and then propagates these labels to the unlabeled data points based on their similarity or proximity in the feature space. The idea is that similar data points should have similar labels. Label Propagation can be applied to various tasks, including classification and graph-based semi-supervised learning.

- **Self-Training:** Self-training is a simple yet effective semi-supervised learning approach. It begins with a limited set of labeled data and uses a machine learning model to make predictions on the unlabeled data. It then assigns the predicted labels to the unlabeled data points with high confidence. These newly labeled data points are added to the training set for the next iteration. Self-training iteratively refines the model and improves its performance with the help of pseudo-labeled data.

2.3 Studies Detecting Rope for Robotic Programs

Presently, there is a notable absence of studies dedicated exclusively to the application of computer vision in the realm of detecting distinct steps within knots. While other researchers have ventured into related domains, such as those involving rope detection, the specific focus on rope state recognition remains relatively uncharted territory.

Prior research endeavors have concentrated their efforts on training models with the primary objective of facilitating robotic arms in the intricate task of knot untangling. These studies represent an adjacent field where computer vision plays a pivotal role but deviates from the precise context of step-by-step knot analysis and recognition.

2.3.1 Research 1 - Vision based Topological State Recognition

A study on ropes by Yu et al. [2] was used to train robotic arms to untangle knots automatically. Throughout their research, they used a method on how to label the rope, and eventually, Yolov3 was used to detect cross points and endpoints which could achieve the goal of topological state recognition.

For the training, a total of 1633 images were taken manually. The training was done on four different table clothes, as well as random obstacles in the images to further test the model accuracy when training. While these obstacles were in the images, these items never covered the camera's sight of the target. The dataset had 800 images containing one rope while the other 833 contained two ropes. With the help of the program LabelImg the dataset was classified, as seen in Figure 2.4 each endpoint and where the rope overlaps itself were marked and labeled accordingly.

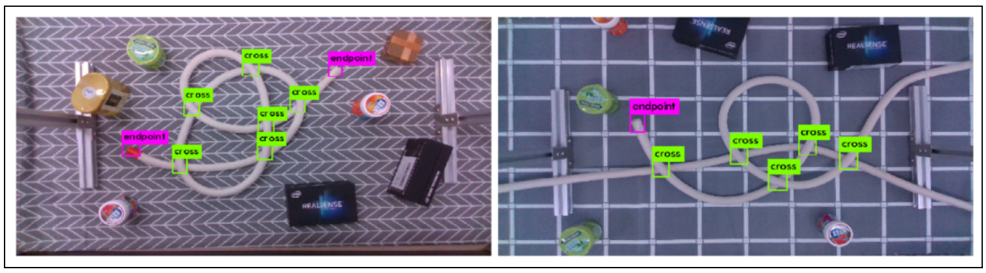

Figure 2.4: Labelling and Backgrounds [2]

2.3.2 Research 2 - Learning to Untangle Ropes with RGB-D Perception

Lui et al. [3] used a different method to detect the rope structure. Firstly, removing the background by place fitting and splitting the rope structure into segments as shown in Figure 2.5. To obtain the order of these segments and to be able to represent the correct rope configuration a rule was set; every point must have 2 neighbouring points, except the 2 endpoints. By using a point system, they were able to compute a feature vector that could be computed directly. A graph structure was used to define the optimal graph that represents the structure of the rope, by using a particle filter algorithm this would find the highest-scoring rope configuration.

Figure 2.5: Rope Segmenting [3]

2.3.3 Research 3 - Untangling Dense Non-Planar Knots

A different approach building on the research before, Grannen et al. [4] also used a marking system on the ropes as seen in Figure 2.6. This segmenting system would differentiate in the markings, by marking the 2 endpoints and the overlapping parts of the ropes, and not a segment every couple of centimeters. This would end up with a much smaller linear graph as seen in the figure example. These segments would get the annotations of (+) for when the rope is passing from overlapping segments and (-) for when the rope is passing from under overlapping parts.

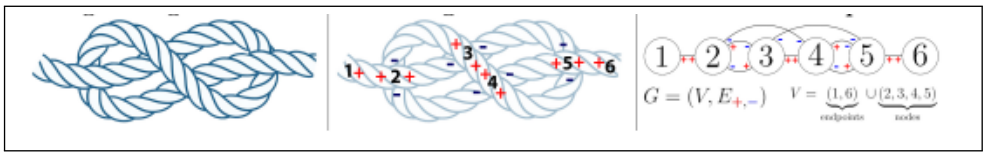

Figure 2.6: Rope Segmenting [4]

2.4 Building a Custom Dataset

2.4.1 Research 4 - Mapping for Automated Edge-Based Pallet Racking Inspections

Hussain et al. [5] used the new state of the art technology YoloV7 to automate the classification of pallet racking of five different classes: horizontal, vertical, support, vertical damage, and support damage. To show the trained architecture's realistic inference speed, a benchmark on its performance not only on architectural and computational performance but also on post-deployment metrics like frame-per-second (FPS).

The dataset was collected manually, and the recordings of three different warehouses' pallet racks were made using smartphones. After this annotation of the data was done carefully on the five different classes stated before. The data annotation was done on images in an occupied warehouse, while the stock was being placed on the racks. To work around this problem only annotating the item would be done if 25% or less of the item is occluded where the occluded part would also be annotated. Where the occlusion would take more than 25% of the class, only the showing part would be marked as shown in Figure 2.7.

Figure 2.7: Racking Labelling [5]

This research architecture was designed to identify pallet racking in a wide range of warehouses. It was logical to predict that various locations would have distinct external variables to which the model-trained architecture would have to adjust. For example, warehouse A may have more lighting than warehouse B, because of the location. It may also vary if the dataset was taken in day or night shifts. To tackle this situation a random brightness level varying between -11% and +11% was applied to each frame of the videos. The dataset consisted of 2094 samples, where 1905 were used for training, 129 were used for validation and the last 60 samples were used for testing.

2.4.2 Research 5 - Autonomous Pallet Racking Inspection System - Mobilenetv2

Another research by Hussain et al. [14] used a different approach than the previous paper to solve similar problems on pallet racking damages. Instead of using YoloV7, the MobileNetV2-SSD architecture was researched and used. When the research was being done, no public dataset of warehouse pallet racking was found, by mounting a smartphone to a forklift, videos were taken and the frames were split with one second intervals to gather images.

The dataset preprocessing was done in two parts. First, the process of removing the images containing no racking and secondly, resizing all the images to 416 x 416 pixels. As this was an object detection localization problem., bounding boxes were used, by using the Computer Vision Annotation Tool (CVAT) by OpenCV which made it simpler to calculate the predictions.

Augmentation of the images was also done to increase the model's capability of predicting correctly. Images randomly selected were cropped by 0-3% and a random rotation to each was set from -6 to +6 degrees. This rotation and cropping

were done to further improve the scenarios this model would be tested against. As the forklift would have different amounts of weight distribution while also moving around, this would improve how the camera installed would be moved and wobble around. A random brightness level between -5% and +5% would be applied while adding a blur effect to some images to continue simulating real-life situations. Variances in the moving speed, warehouse location, and time of day would all be factors the model would be trained against. With the use of PyTorch, TensorFlow, and Keras frameworks this was done while using the Google Collab GPUs found online. Google, Microsoft Azure, and Amazon Web Services all have these free resources to be used for investigations like this.

Their second research [14] had slightly better results than the previous research [5], with a mean over precision (mAP) of 92.7% surpassing the other research using YoloV7 by 1.6% mAP as seen in Table 2.1. When compared with the research using RCNN architecture [15] this used much less computational power as the RestNet-101 backbone has 44.5 million learnable parameters. Having high-end devices would not be suitable for that demand making it unusable to the average customer. Although the second paper has high results, the authors may believe that when the model is used in the real world there would be different outcomes. This may come into effect because the model was trained on only a single damage class of vertical damage and one good class of no damage.

Table 2.1: A table comparing research on warehouses

	Dataset Size	Classes	Architecture	mAP@0.5
[5]	2094	5	YoloV7	91.1%
[15]	75	1	Mask-RCNN-ResNet-100	93.45%
[14]	19,717	2	MobileNetV2-SSD	92.7%

2.5 Evaluating Results

2.5.1 Research 6 - Automatic Fabric Defect Detection

Jin and Nui [16] proposed a teach-student YoloV5 model to perform defect detection on different materials and different defects. In the research done before, they found 2 primary databases that could help them conduct this research (TILDA and Xuelang Tianchi AI Challenge).

The TILDA database had a dataset of 300 fabric images divided into six categories, five different defects, and 1 normal classification. The images were downsized to 256 x 256 pixels. The Xuelang Tianchi AI Challenge dataset contained 3331 labeled images, split into 2163 images containing no defect while the other 1168 images had from one to multiple defects. This dataset contained 22 types of defects. Both databases were split in 70% as the training set while the other 30% was used as a test set.

Their results using the TILDA dataset when comparing the teacher and student networks had an area under the ROC curve (AUC) of 0.988 and 0.965 respectively and a mean average precision (mAP) of 0.451 and 0.428. The Xuelang Tianchi AI Challenge dataset also was tested on the teacher and student networks and ended up having slightly worse AUC results of 0.981 and 0952 while a mAP of 0.447 and 0.406.

This research on an automatic fabric defect detection method based on YoloV5 was valuable as it was a way to detect defects in fabrics, with the use of technology. The use of a teacher-student network architecture allows for real-time detection with high accuracy. The proposed method was evaluated using both public databases and manually collected fabric images while comparing the AUC, mAP, and other metrics. Overall, this research had the potential to significantly

improve the accuracy of defect detection and increase the automation level of the textile industry.

Chapter 3

Research Methodology

3.1 Data Procurement

The rationale behind embarking on this research initiative is deeply rooted in the conspicuous absence of prior scholarly investigations dedicated to the innovation and advancement of knot learning techniques. Intriguingly, the domain of knot creation and understanding has remained relatively uncharted territory in the field of computer vision and machine learning. One striking limitation that has hindered progress in this domain is the conspicuous lack of accessible, open-source datasets containing comprehensive representations of the distinct stages involved in knot formation. This dearth of foundational data presents a substantial barrier to researchers and practitioners interested in this specialized topic.

It is worth noting that the time-intensive process of curating a substantial repository of images specifically related to knot-tying, with meticulously annotated details capturing the intricacies of each knot's development, has invariably contributed to the scarcity of previous research makings in this area. Furthermore, the problem of accessible data has constrained the potential for innovation and experimentation, hindering the development of knot learning algorithms.

In light of these critical limitations, the primary objective of this research is to

fill this void and bridge the gap that has persisted in the field of knot learning techniques. Through diligent efforts in data collection, annotation, and augmentation, this study aspires to lay the foundation for a comprehensive open-source dataset focused on knot creation.

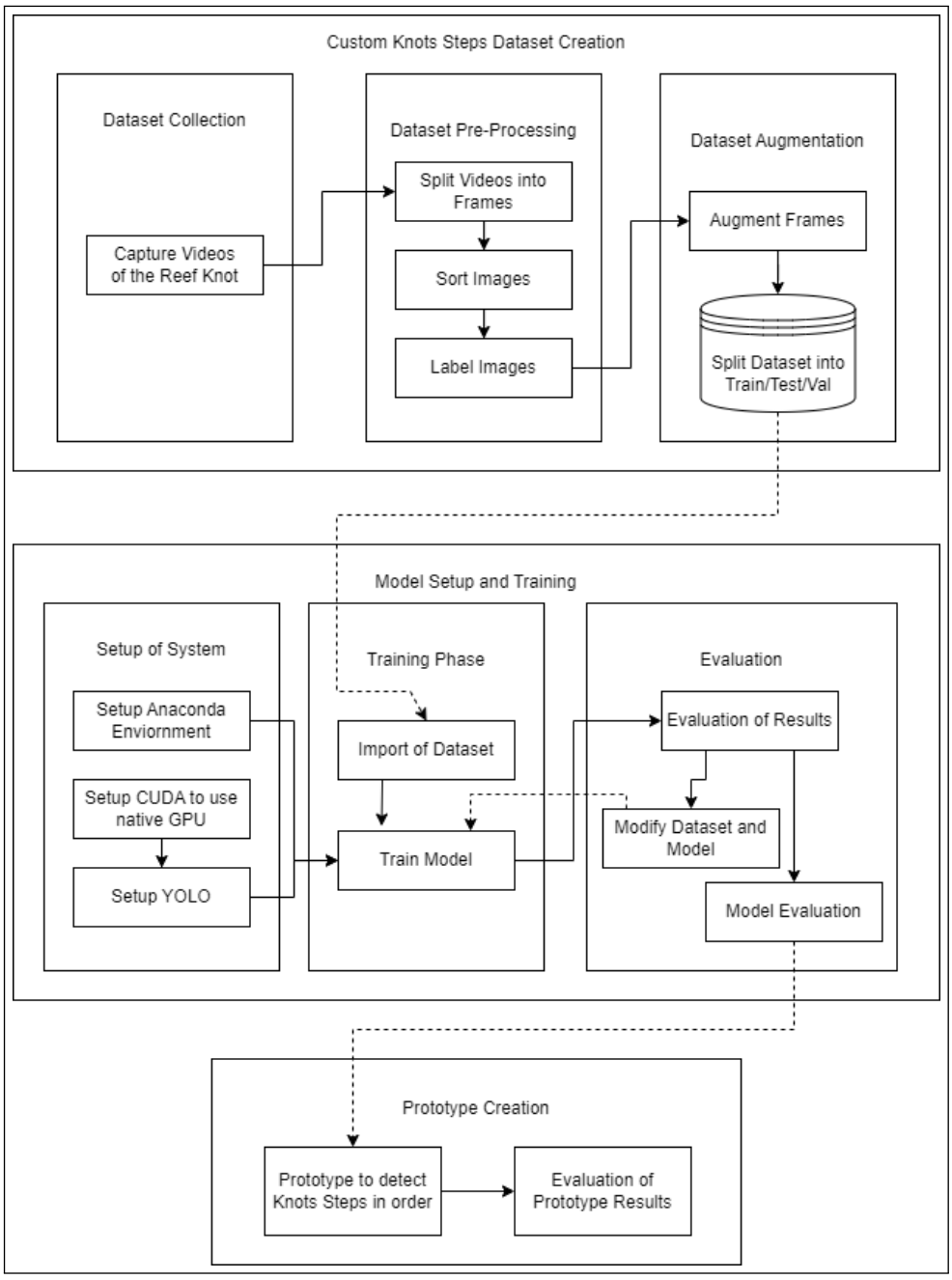

Figure 3.1: Prototype Pipeline

3.1.1 Dataset Collection

To answer the research question *"What key features and characteristics of the data set are needed for proper training of a model?"* certain steps had to be done. Different variables had to be taken into consideration, such as camera, lighting, backgrounds, rope variety, and algorithms to be used.

To answer this research question a pipeline found in Figure 3.1 was created. This provides a comprehensive visual representation of the entire research process, spanning from the collection of the dataset to the results acquisition from a prototype. The dataset acquisition phase involved the utilization of a Full HD 1080P, 12.0 Mega Pixels AUSDOM Camera, positioned above the monitor in a configuration designed to capture top-down views. To ensure the quality and consistency of the collected data, conditions were established, emphasizing minimal obstructions within the camera's field of view, and a deliberate effort to eliminate or significantly reduce objects from the captured shots.

Distinguishing itself from prior studies such as Song et al. [2], which often employed different backgrounds, the research incorporated a deliberate variation in backgrounds. Specifically, the dataset was collected against two distinct settings: a brown plain desk and an alternate setting featuring a white cloth adorned with an intricate colourful pattern. For the purpose of this research, the Reef Knot was chosen as a beginner knot. Five sample steps shown below in Figure 3.2 were used for this knot, these were taken from the popular knot-making website[1]. As seen on the animated knots website, learning to make knots is as simple as looking and doing, with no actual way of checking if you are on the correct step.

Multiple ropes were chosen to take the videos for the dataset. Eight different colours of ropes were used during the filming, while also having different widths of ropes for better control when coming to the testing of the model. Materials of

[1] `https://www.animatedknots.com/square-knot`

the ropes also varied, having some ropes being nylon and others being polyester. This choice seeks to enrich the dataset's diversity and enhance its applicability in real-world scenarios, recognizing the influence of background variation on the robustness and adaptability of computer vision models.

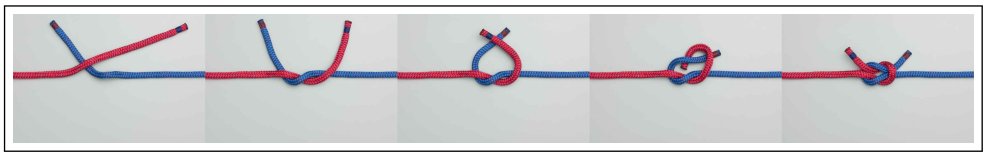

Figure 3.2: Reef Knot Steps

3.1.2 Dataset Pre-Processing

The journey of preparing the dataset for algorithmic utilization involved a multi-faceted process, orchestrated through several iterative stages. Commencing with the capture of fifteen distinct videos and accumulating to a runtime of eighteen minutes, the preliminary dataset underwent a series of transformations to render it useable for algorithmic processing. A pivotal step in this endeavor was the development and deployment of a custom splitter program, found in the appendix, extracting every tenth frame from the video footage. This extraction process, coupled with a naming convention, endowed each frame and eased the transition to a more organized dataset. Subsequently, a critical phase of manual data creation ensued a painstaking process that made the dataset from its initial count of 3400 images to a more refined collection of approximately 1600 images, resulting in a substantial reduction of nearly 50% of the original dataset. This process unfolded with deliberate caution, with attention to detail to ensure that images were not uselessly removed. These multifaceted dataset preparation efforts aimed to not only streamline the data for algorithmic compatibility but also to enhance its quality, ultimately facilitating more robust and accurate algorithmic training and evaluation.

To initiate the data refinement process, images having any obstructions, often ne-

cessitating the manual removal of occluding hands, were the first to undergo. Given the intricacies involved in maintaining unobstructed camera views throughout the procedure. Furthermore, images, where the rope exhibited blurriness or where the reef knot step remained obscured, were systematically removed from the dataset, resulting in a carefully curated selection of 1600 images. Subsequently, the dataset underwent a process of categorization, whereby each image was meticulously annotated using the YoloLabel program. This annotation involved the creation of rectangular bounding boxes, which encompassed the target item, accompanied by a corresponding text file containing class labels and coordinates, denoted in an XX XY YY YX format. To facilitate the subsequent training and evaluation phases, the dataset was thoughtfully partitioned into distinct subsets: training, validation, and testing, adhering to the conventional split ratio of 60%, 20%, and 20%, respectively. This partitioning yielded approximately 1,000 images for the training set, while allocating 300 images each for the validation and testing sets, ensuring a comprehensive yet manageable dataset for subsequent algorithmic development and assessment.

Table 3.1: Reef Knot Dataset Distribution

Class	Train	Val	Test	Total
reef_1	195	65	65	325
reef_2	204	68	69	341
reef_3	188	62	64	314
reef_4	176	58	60	294
reef_5	204	68	69	341
Total	967	321	327	1615

3.2 Model Setup and Training

The images split into their corresponding folders were used to train the model to eventually be able to detect the knot steps one after the other. For the training to be done, multiple steps had to be done beforehand. Firstly, setting up CUDA was essential, this would run the YOLO algorithm. A dedicated laptop with an

NVIDIA Ge Force RTX 4070 Laptop GPU, Intel Core i7-13700H, 2400MHz 14 Core CPU, and 16GB RAM was used to run this algorithm. Anaconda was used to create a separate environment and to track what is being used on the GPU and make sure that the training is not affected by other programs. This was all recorded on the WANDB platform [17], which is a tool that can be accessed by an online browser to see how the training, testing, or validation process is going. This gives a more in-depth view of the results and an even better logging system to be able to compare runs with each other.

3.2.1 Configuring YOLO Model

Building a model configuration that can effectively predict a reef knot step was also needed to answer the next research question *"What algorithms can be used for the recognition of the different phases in tying a knot?"*. This had to be efficient in the training stage, have an accuracy that had a high success rate, and be able to detect the reef knot steps in real time.

Initially, in the pursuit of developing an effective methodology for this research, I tried using YOLOv7 for training a model. My initial experiments involved running the model with a batch size set at 10, a single worker, and executing 3 epochs, which took approximately 4 minutes and 30 seconds to complete. Subsequently, I extended the training to 80 epochs, spanning approximately 82 minutes in duration. However, during this prolonged training, a significant challenge emerged as evidenced by the results monitored through WANDB. Specifically, the training process displayed a noticeable drop in performance at epoch 10, followed by a lack of substantial improvement over the subsequent 70 epochs, as clearly depicted in the table below. This critical observation prompted a reevaluation of my approach, leading to using a different Yolo algorithm as seen below.

Upon transferring the dataset into the YOLOv5 environment, it became important

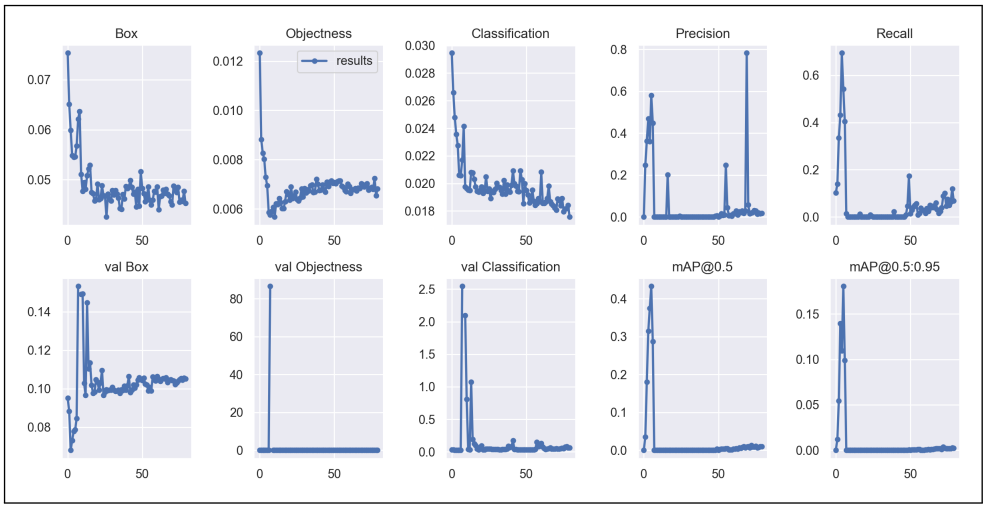

Figure 3.3: YoloV7 Charts

to conduct a thorough assessment to identify the optimal YOLO model for my research. YOLO, standing as one of the most widely employed object detection algorithms, offers a variety of pre-existing models readily available for training. To streamline this process and ensure both efficiency and compatibility with the computational resources at hand, a preliminary examination was undertaken to calculate the runtime required for executing three epochs. This preliminary test served a dual purpose: first, it aimed to make sure that running 200 epochs would not be excessively time-consuming, and secondly, it served to validate the computational capabilities of the laptop used for training.

The results of this runtime analysis, as presented in Table 3.2, underscored significant disparities among the various YOLOv5 models. Notably, yolov5s, yolov5n, and yolov5m exhibited markedly faster performance in comparison to their counterparts. Consequently, it was a judicious decision to select yolov5m as the model of choice for the extensive training endeavor, involving the deployment of four workers and spanning a duration of 200 epochs. The rationale behind opting for such an extensive number of epochs was to mitigate the need for further retraining, as this would require recommencing the training process from its inception. This strategic selection sought to strike a balance between time efficiency and robust model development, ensuring that the chosen model would adequately meet

the research objectives without the need for subsequent refinements.

Table 3.2: YoloV5 Models Training Evaluation

Model	Epochs	Workers	Time Taken
yolov5n	3	8	240s
yolov5n	3	4	200s
yolov5s	3	8	300s
yolov5s	3	4	200s
yolov5m	3	8	N/A
yolov5m	3	4	210s
yolov5l	3	8	N/A
yolov5l	3	4	1260s
yolov5x	3	8	N/A
yolov5x	3	4	2640s

3.2.2 Training

Following some initial testing under identical conditions to those used during the training phase, it became evident that a discrepancy had emerged in the dataset employed for model development. This discrepancy was due to the necessity to label the dataset, which resulted in all the images being horizontally flipped. Consequently, when deploying the trained model for inference, it exhibited sub-optimal performance, occasionally failing to detect the target object, the reef knot step in this case. To rectify this issue and enhance the dataset's diversity, a decision was made to employ data augmentation techniques. The Roboflow program proved instrumental in this endeavor, firstly inputting all of the data into the program. After resizing each image to 640x640 pixels the implementation of a simple yet effective flip augmentation on each image within the dataset.

By applying this augmentation, the original dataset, comprising 1615 images, was expanded to a more extensive collection of 2817 images. This augmented dataset, as detailed in Table 3.3, was subsequently reconfigured into distinct training, validation, and testing sets, using the same ratios as in the initial model development process. The model was then systematically retrained, mirroring the methodology

Figure 3.4: Sample of Ground Truth Images Compared to a Sample of Predictions during Training

employed during its initial training, with the augmented dataset now providing a richer and more representative foundation for model refinement and validation. This strategic augmentation approach aimed to address the dataset's shortcomings and improve the model's capacity to accurately detect and recognize the target object, ultimately contributing to the robustness and reliability of the research outcomes.

Table 3.3: Reef Knot Dataset after Augmentation

Class	Train	Val	Test	Total
reef_1	342	114	115	571
reef_2	359	119	121	599
reef_3	327	109	110	546
reef_4	295	98	99	492
reef_5	365	121	123	609
Total	1688	561	568	2817

3.3 Evaluation of Results

Another research question would be answered by calculating different metrics that determine and assess between a good and a bad trained model. Several metrics were used to assess the performance of models trained on the dataset and determine the most effective model. These metrics include Precision, Recall, Mean Average Precision (mAP), F1 Score and Accuracy. To compute these metrics, specific indicators are used in the formulation of each metric: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Table 3.4: Example of a simple Confusion Matrix

		Predicted Class	
		P	N
Actual Class	P	TP	FN
	N	FP	TN

Accuracy is a metric that measures the overall correctness of predictions made by a model. It is calculated as the ratio of correct predictions (both true positives and true negatives) to the total number of predictions, expressed as:

$$Accuracy = \frac{TP+TN}{FP+TP+FN+TN} \tag{3.1}$$

While accuracy is a commonly used metric, it may not be suitable for imbalanced datasets where one class significantly outnumbers the other. In such cases,

high accuracy can be achieved by simply predicting the majority class, even if the minority class is poorly predicted. Precision and recall provide a more nuanced view of model performance in such situations.

Precision is a metric that measures the accuracy of positive predictions made by a model. It is calculated as the ratio of true positive predictions to the total positive predictions, expressed as:

$$Precision = \frac{TP}{TP+FP} \tag{3.2}$$

Precision tells us how many of the items predicted as positive are actually true positives. A high precision indicates that the model is good at not falsely labeling negative cases as positive.

Recall, also known as sensitivity or true positive rate, measures the ability of a model to correctly identify all relevant instances. It is calculated as the ratio of true positive predictions to the total actual positive instances, expressed as:

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

Recall helps us understand how many of the actual positive cases were successfully predicted by the model. A high recall indicates that the model is good at capturing most of the positive cases.

The F1 Score is an important metric in machine learning, especially when you want to balance precision and recall. It combines both precision and recall into a single value, making it useful when you want to evaluate a model's performance while considering both false positives (FP) and false negatives (FN). The

F1 Score is particularly helpful in situations where there is an imbalance between positive and negative classes or when you want to strike a balance between precision and recall.

$$F1 = \frac{precision * recall}{precision + recall} \tag{3.4}$$

Mean Average Precision is a metric commonly used in object detection and information retrieval tasks. It considers precision at multiple levels of recall and then averages them. It quantifies the quality and completeness of ranked retrieval results. mAP is calculated by plotting precision-recall curves for different levels of confidence thresholds and then taking the average of the areas under these curves.

mAP is particularly useful when dealing with models that return ranked results, such as when ranking search results or detecting multiple objects in an image.

$$mAP = \frac{1}{n} \sum_{k=n}^{k=1} \mathrm{AP}_k \tag{3.5}$$

3.3.1 Proof-of-Concept Prototype

To assess the feasibility and functionality of step prediction in the trained models, a proof-of-concept prototype was developed. This prototype served as an initial test to validate the viability of the approach and set the foundation for future advancements in this domain. The construction of the prototype involved direct modifications within the YOLO code file 'detect.py,' ensuring compatibility with real-time model testing. The prototype featured a user interface comprising a camera window on the left and a display of the current step to be executed

on the right as seen in Figure 3.5. Upon successful detection of a step, the subsequent step that needed to be performed next would be presented on the right side, facilitating a seamless user experience. After the prototype is stopped the whole prototype running process would be saved to better analyze the results further.

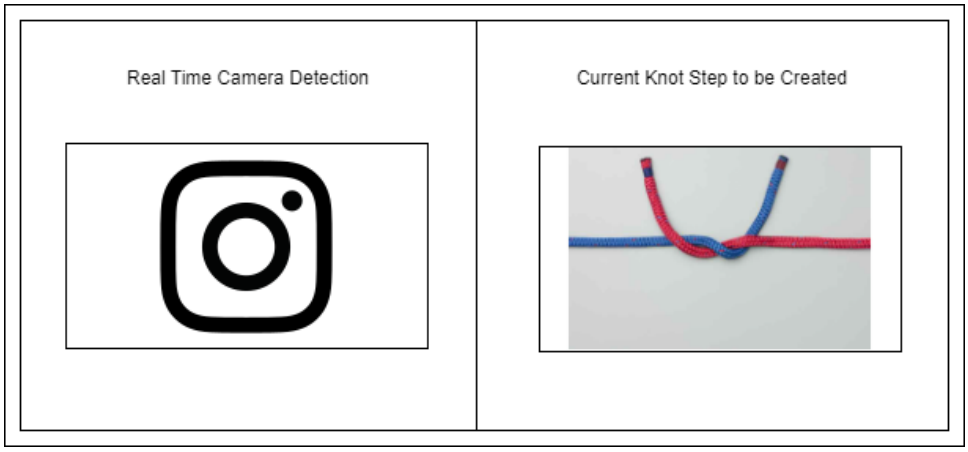

Figure 3.5: Prototype Design

Chapter 4

Analysis of Results and Discussion

4.1 Introduction

The objective of this chapter is to present the outcomes derived from the prototype discussed in Chapter 3. These results are intended to comprehensively address all research questions and offer a thorough examination of the gathered data. The data obtained throughout the model's testing has been meticulously analyzed to effectively categorize and structure the significant discoveries. To examine the results of the trained model, multiple experiments explained below had to be considered.

4.2 Knot Detection Experiments

4.2.1 Metrics Before and After Augmentation

As detailed in the Research Methodology section, it was opted to enhance the Reef Knot dataset by using the Roboflow program. This augmentation process involved the addition of more images, thereby significantly increasing the dataset's size from an initial 1600 images to a total of 2800 images. The impact of this

augmentation on key performance metrics is evident in the results presented in both Table 4.1 and Table 4.2.

Notably, the augmentation had the most pronounced effect on the precision metrics, with a substantial increase observed in step 2. Additionally, improvements were also noteworthy in steps 1 and 3, particularly when assessing recall and mean average precision. On average, when considering all three metrics; precision, recall, and mean average precision the enhancements resulted in an impressive 12% increase in precision, a remarkable 35% boost in recall, and a noteworthy 28% improvement in mean average precision. These enhancements underscore the significant value of dataset augmentation in augmenting model performance and overall research quality.

Table 4.1: Reef Knot Results before Augmentation

Class	P	R	mAP50
reef_1	0.888	0.579	0.625
reef_2	0.746	0.665	0.759
reef_3	0.892	0.444	0.537
reef_4	0.848	0.735	0.841
reef_5	0.957	0.741	0.827
Average	0.866	0.633	0.718

Table 4.2: Reef Knot Results after Augmentation

Class	P	R	mAP50
reef_1	0.962	0.965	0.963
reef_2	0.989	0.958	0.994
reef_3	0.999	1	0.995
reef_4	1	0.994	0.995
reef_5	0.99	1	0.995
Average	0.988	0.983	0.988

Below, the F1 Scores tables were plotted for these two distinct models. A change was introduced by adjusting the confidence threshold from its default value of 0.25, as defined by YOLO, to a more stringent 0.5. Upon closer examination of Figure 4.1, it became evident that the model trained without the inclusion of augmented images struggled to correctly predict the various steps, often yielding low confidence scores for its predictions or failing to make predictions for certain

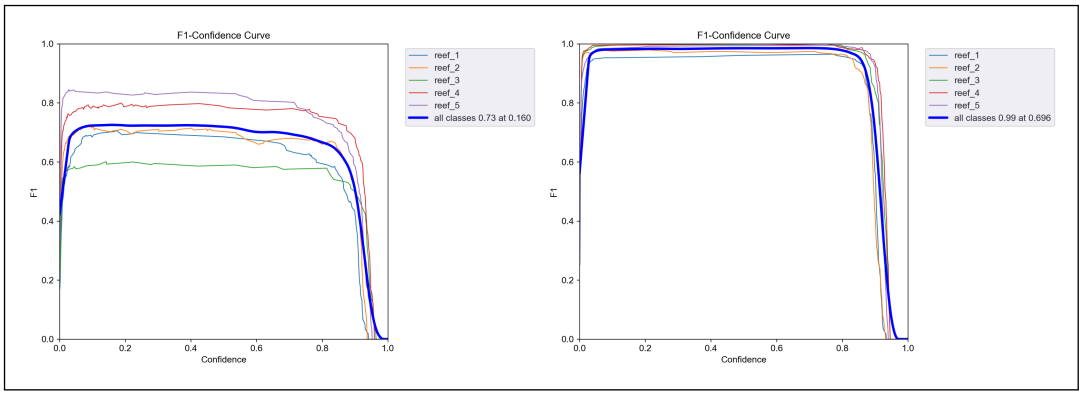

Figure 4.1: F1 Score Comparing the difference of datasets

steps altogether. Its F1 Score ranged from a mere 16% to a modest 71%.

In stark contrast, as illustrated in the figure below, the augmented model exhibited markedly improved predictive capabilities, consistently achieving confidence levels exceeding 70% to 99%. This stark contrast in performance highlights the significant impact of dataset augmentation in enhancing the model's ability to accurately identify and predict the steps involved.

4.2.2 Experimenting Model Training for Different Knot

Building upon the positive outcomes achieved through the expansion of the previous dataset, a decision was made to create an entirely distinct dataset for a different domain. Simultaneously, a new model would be crafted from this dataset. It was determined that introducing the "Bowline Knot" as the second, more sophisticated knot would be a highly advantageous choice. Aiming to prove whether the dataset building process was sufficiently comprehensive or if additional details needed to be included.

The same structure was used to build the "Bowline Knot" dataset as the "Reef Knot" dataset. First, find the correct steps on the Animated Knots website[1]. In

[1] `https://www.animatedknots.com/bowline-knot`

Figure 4.2 one might already notice, that there is not much difference between the steps, but skipping one of these steps will not achieve the finalized knot result.

Figure 4.2: Bowline Steps

Building of Bowline Knot Dataset

The meticulous process of constructing the dataset started with the acquisition of over sixteen videos, featuring various colours of ropes against two distinct backgrounds. This compilation of visual data totaled approximately twenty minutes of footage, serving as the foundation for the dataset. Leveraging the same video-to-frame splitter program previously developed, I transformed this video content into a dataset comprising an impressive array of more than 3500 individual images. However, this initial compilation was just the beginning; cleaning and curation were essential to ensure the dataset's quality.

Despite taking great care during video capture to minimize motion blur and obstructions, a substantial portion of the dataset required removal. From the original 3500 images, it was pruned down to approximately 1700 images. The rationale behind this reduction lay in the inherent intricacy of the Bowline Knot's steps, which demanded higher precision in image selection compared to the Reef Knot. Eliminated images often featured partially obscured ropes or frames that captured incomplete sequences, such as a frame containing half of step 3 and step 4. Retaining these images would have introduced confusion during the program's classification and training stages.

To enhance the dataset's diversity and robustness, Roboflow was used, employing

data augmentation techniques. Initially, the resizing of all images to a uniform dimension of 640 x 640 pixels. Subsequently, applying horizontal image flips effectively doubles the dataset size. This augmentation strategy resulted in a final dataset containing 2900 meticulously curated images for the Bowline Knot.

In line with the data management principles, the dataset was segmented into three subsets: training, validation, and testing. The distribution was allocated as 60%, 20%, and 20% respectively. This strategic partitioning, illustrated in Table 4.3, was designed to optimize the training outcomes by ensuring a well-balanced representation of data across the classes.

Table 4.3: Bowline Knot Dataset after Augmentation

Class	Train	Val	Test	Total
bowline_1	388	129	131	648
bowline_2	309	103	103	515
bowline_3	307	102	103	512
bowline_4	363	121	121	605
bowline_5	375	125	126	626
Total	1742	580	584	2906

Training Bowline Knot Model

Following the success of the Reef Knot model, the time had come to embark on the task of training a model for the Bowline Knot dataset for the purpose of testing and comparing. For this experiment, the same YoloV5 model, specifically the YoloV5m variant, which is readily available for download from the Yolo Github repository[2] was used.

Running the model with consistent parameters, including 200 epochs, 4 workers, a batch size of 16, an image size of 640 pixels, and customized data.yaml file accommodating the five distinct classes of the Bowline Knot, to start the training process. It's worth noting that the YoloV5m model is a variant of the YoloV5 ar-

[2] `https://github.com/ultralytics/yolov5`

chitecture optimized for real-time object detection tasks as not all YoloV5 models are viable for this result. After a four-hour training session, a successfully generated model was tailored to the Bowline Knot dataset. This model was equipped with the capability to evaluate its performance using the val.py command included in the YoloV5 package. The val.py command provided insights by furnishing precision, recall, and mAP metrics.

The results, as illustrated in Table 4.4, revealed metrics that surpassed those achieved by the Reef Knot model, despite the Bowline Knot's intricate and complex series of steps. This outcome underscored the model's proficiency in tackling challenging knot recognition tasks.

Table 4.4: Bowline Knot Results

Class	P	R	mAP50
bowline_1	0.995	1	0.995
bowline_2	0.99	0.999	0.989
bowline_3	0.984	0.99	0.995
bowline_4	0.978	1	0.995
bowline_5	1	0.996	0.995
Average	0.989	0.997	0.994

4.3 Comparing Models Manually

4.3.1 Setting up Environments for Testing

To validate the accuracy and reliability of the Reef Knot and Bowline models, an experiment was devised. Its primary aim was to manually assess whether the models' performance aligned with the previously presented results and to gauge their overall reliability. To ensure the fairness of these tests, meticulous setup procedures were undertaken.

The experiment entailed configuring various test environments, each subject to

specific alterations. Some environments underwent one change, others two, and a few were entirely transformed. Additionally, this experiment served as a test of the models' robustness under different conditions. Below is a list with an example, these examples are the actual images that were used for doing the Reef Knot step 1 tests.

List of Setup for Environment:

- Environment 1: Setup exactly like the training phase, while using an orange coloured rope.
- Environment 2: A completely new green coloured background to that used in training, and using a white rope.
- Environment 3: A background containing 3 random objects with different levels of colours and sizes, was used to confuse the program, a green rope was used for this test.
- Environment 4: The lighting inside the room was switched off, and the only light coming from the screen monitor was used while using a yellow rope
- Environment 5: While having the light switched off, a blue light was placed near the environment area. This created a darker image and also made the image include more grains. This used a white rope.
- Environment 6: A totally different scenario for the trained model, having a different camera angle, while a face was placed behind the steps to make sure the model was robust enough, for testing of this model an orange coloured rope was used.
- Environment 7: The setup was completely the same as the training model, but removed my hands to make sure that the steps' features were not con-

fused and the model would be detecting the hand's placement instead of the actual knot, a thick white rope was tested in this scenario.

- Environment 8: A wire was used instead of a rope, to check if the material trained on would affect the results.
- Environment 9: A green background which the model has never seen for training was used and a green rope was used on top of the background. To make sure the model would be able to make a difference between the depth of the rope and the background.
- Environment 10: A completely different rope style was used from that it was trained on while having the same setup. The rope was multi colours, having hints of white, blue, orange, and black.

Examples of Environments Used:

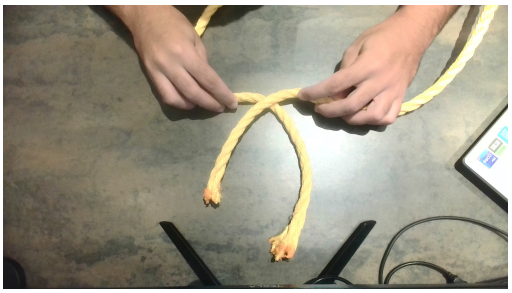

Figure 4.3: Environment 1 Example

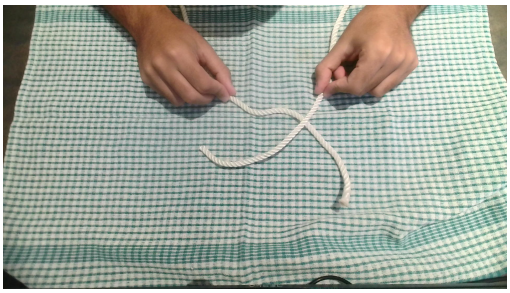

Figure 4.4: Environment 2 Example

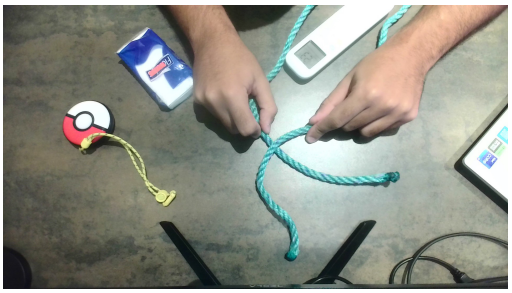

Figure 4.5: Environment 3 Example

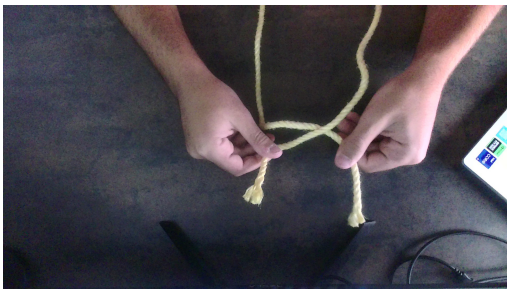

Figure 4.6: Environment 4 Example

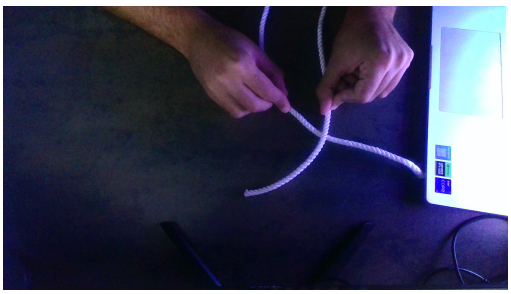

Figure 4.7: Environment 5 Example

Figure 4.8: Environment 6 Example

Figure 4.9: Environment 7 Example

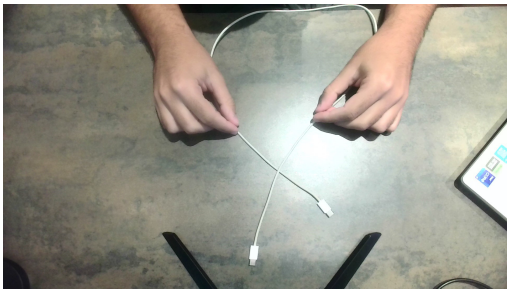

Figure 4.10: Environment 8 Example

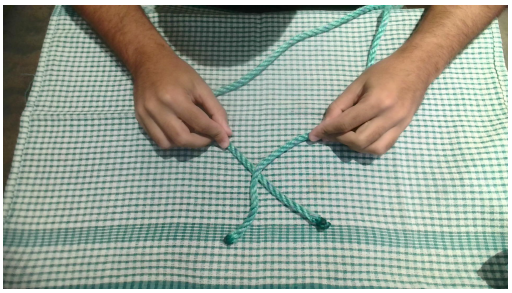

Figure 4.11: Environment 9 Example

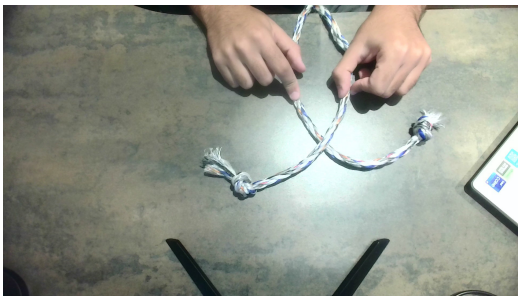

Figure 4.12: Environment 10 Example

4.3.2 Evaluating Results from Manual Testing

The outcomes of the manual Reef Knot and Bowline models can be observed in the following tables, specifically in Table 4.5 and Table 4.6. These tables provide a comprehensive display of results, including averages per environment and per individual reef knot or bowline step.

The values within the tables correspond to the predicted steps generated by the model. For example, in the case of ”reef 1” and environment number 2, step 1 was successfully identified. Conversely, in environment number 1, no step detection occurred and this is the reason why ”0” is used in the table. Additionally, there are scenarios where multiple steps were detected within a single image. For instance, when attempting to predict bowline step 2 in environment 3, two distinct steps were identified.

It is essential to note that these results should not be interpreted as indicative of the model’s inability to predict accurately in all situations. It is reasonable to as-

sume that in real-time testing, the indicated step will likely be predicted correctly with a slight movement of the knot step or hands, as the model processes more than just a single frame at a time.

Evaluating The Reef Knot Manual Testing Results For Each Step

Reef 1: The average detection rate for Reef Knot step 1 across all environments is 80%. One of the reasons that it was predicted incorrectly in environment 5 was the different grainy images used created with the blue light effect.

Reef 2: Reef Knot step 2 had an average detection rate of 90%, indicating a higher success rate in identification across all environments. Also, this was not predicted correctly in environment 5 due to the same reason in step 1.

Reef 3: Step 3 had an average detection rate of 40%, suggesting that it was less consistently recognized compared to other steps. A huge reason for these results is because step 3 is made from step 1 and step 2 on top of each other as can be seen in Figure 3.2 found in Chapter 3. The environments where the background or rope type was completely different were the reason why most of step 3 was not predicted correctly.

Reef 4: Reef Knot step 4 had an average detection rate of 60%, indicating moderate success in identification. The failure of detection was in environments where most of the other steps did not have a good prediction as well.

Reef 5: Step 5 had an average detection rate of 70%, demonstrating a relatively consistent identification rate across the scenarios. I think this had a notably high success rate compared to the other four steps. However, additional testing is necessary to assess whether introducing a different tight knot might impact the results.

Evaluating The Reef Knot Manual Testing Results For Each Environment

Environment 1: In this scenario, which closely resembles the training phase, with an orange rope, the model achieved an average detection rate of 80%. It performed well under familiar conditions.

Environment 2: This environment introduced a new green background and a white rope, differing from the training setup. Despite these changes, the model still achieved a 60% average detection rate, indicating moderate adaptability to novel backgrounds.

Environment 3: In a more complex setup with random objects of varying colours and sizes against a green background, the model maintained a solid average detection rate of 80%. This suggests the model's ability to handle small random objects in the area of the environment.

Environment 4: With the room's lighting turned off and illumination solely from the screen monitor, along with a yellow rope, the model achieved an 80% average detection rate. The reason for the high accuracy might be that the monitor lighting was enough to resemble to desk background used in the training of the model

Environment 5: Similar to the previous environment, this setup had the lights off, but a blue light source was introduced. This caused a darker and grainier image. The model did not manage to successfully predict any of the steps due to the harsh changes in the environment image.

Environment 6: In a scenario with a different camera angle and the presence of a face behind the steps, the model used an orange rope. It achieved a 60% average detection rate, demonstrating a degree of robustness even in unfamiliar settings.

Environment 7: The model showed impeccable results for this scenario, having a rate of 100% success. The reason for this result may be because in other tests the hands might be covering part of the knot step. This implies that removing occlusions from above the knot gives better results.

Environment 8: Testing with a wire instead of a rope aimed to assess the effect of material differences. The model's adaptability dropped, with a 40% average detection rate, suggesting sensitivity to material variations, although having the same environment setup as training.

Environment 9: The introduction of an unseen green background and a green rope to the model's training setup aimed to test depth perception. The model exhibited good performance with an 80% average detection rate, indicating an ability to distinguish between the rope and background.

Environment 10: Using a completely different multi-coloured rope in a familiar setup challenged the model's ability to recognize new rope styles. However, it achieved a 100% average detection rate, suggesting flexibility in rope-colour recognition meaning that the rope colour would have no to little effect on the results of the model.

In summary, the model's performance varies across different environments, with factors such as lighting, background, material, and rope style affecting its detection rates. The overall average detection rate for all reef knot steps across all scenarios is not provided in the table but had an average rate of 68%. This overall average gave a comprehensive view of the model's performance in Reef Knot Model recognition across diverse environmental conditions that were set. Having steps such as step 3 be created out of steps 1 and 2 together does impose some challenges. Also having step 4 be created by having 2 step 2's on top of each other causes some problems. Overall, it shows adaptability and robustness in several challenging scenarios, while some conditions, like lighting changes and

material differences (wire), appear to be more challenging for the model.

Table 4.5: Manual Experiments for Reef Knot

Env. Number	reef 1	reef 2	reef 3	reef 4	reef 5	Average
1	0	2	3	4	5	80%
2	1	2	2	0	5	60%
3	1	2	1	4	5	80%
4	1	2	3	4	2	80%
5	0	0	0	0	0	0%
6	1	2	0	3	5	60%
7	1	2	3	4	5	100%
8	1	2	0	0	0	40%
9	1	2	0	4	5	80%
10	1	2	3	4	5	100%
Average	80%	90%	40%	60%	70%	

Evaluating The Bowline Knot Manual Testing Results For Each Step

Bowline 1: The average detection rate for Bowline step 1 across all environments is 60%. On average, step 1 was correctly identified in 60% of the scenarios. Showing that having severe background changes affects this step. Environments 4,5,6 and 9 all have a change in the background.

Bowline 2: Bowline step 2 had an average detection rate of 40%, indicating a moderate success rate in identification across all environments. This result shows the same defect when detecting for bowline step 1, but also one can see that in 2 cases this detected step 4 instead. The reason for this is the close resemblance between the steps.

Bowline 3: Step 3 had an average detection rate of 15%, suggesting that it was less consistently recognized compared to other steps. Most of the results show that no prediction was placed on the image due to the low confidentiality rates. Other environments predicted wrong, some predicting step 2 or step 4 instead of step 3.

Bowline 4: Bowline step 4 had an average detection rate of 60%, indicating moderate success in identification. Although this had a high percentage of predicting correctly, it was not consistent enough.

Bowline 5: Step 5 had the same average detection rate of 60%, demonstrating a relatively consistent identification rate across the scenarios. Having problems in the same environments just like steps 1 and 4.

Evaluating The Bowline Knot Manual Testing Results For Each Environment

Environment 1: This environment achieved an average of 80% detection rate. This suggests that steps 2 and 3 resemble each other when looking at the ground-rooted knot steps, one can see this reason.

Environment 2: The introduction of a new green background and a white rope posed a challenge for step 5, resulting in an 80% average detection rate. However, it maintained high performance.

Environment 3: When compared to a more complex background with random objects while using green rope. The model did not do as well as it did in the reef knot. Having a scenario where 2 steps were detected on top of each other. This resulted in a 60% average detection rate.

Environment 4: In a low-light condition with a yellow rope, the model encountered difficulties, achieving a 40% average detection rate unlike having a high rate in the reef knot manual testing.

Environment 5: A similar low-light scenario with a blue light source and a white rope led to a 0% average detection rate, highlighting the sensitivity to lighting changes and grainy image texture. This can also be seen that this environment

affected both the reef knot and the bowline models.

Environment 6: With a different camera angle and the presence of a face, the model struggled, achieving a 20% average detection rate.

Environment 7: Removing hands from the setup to focus on bowline steps led to an 80% average detection showing promising results.

Environment 8: Testing with wire instead of rope resulted in a 60% average detection rate, showing some adaptability to material differences, a simple reason for this result may be because the thin wire is easier to hold.

Environment 9: An unseen green background and green rope posed a challenge, with a 0% average detection rate across all bowline steps meant that the dataset should be updated to have more background count.

Environment 10: Testing with a multi-coloured rope in a familiar setup resulted in a 60% average detection rate, indicating slight adaptability to different rope styles.

In summary, the bowline model's performance varied across different backgrounds mostly. A resulting 50% average rate across all bowline knot steps and environment scenarios is presented. Factors such as lighting, material, and rope style affected its detection rates. A possible reason for a low result in all bowline steps may be the reason for the intricate design of the knot. Having many characteristics of the step be similar to one another caused most of the problems.

Table 4.6: Manual Experiments for Bowline

Env. Number	bow 1	bow 2	bow 3	bow 4	bow 5	Average
1	1	2	2	4	5	80%
2	1	2	3	4	0	80%
3	1	4,5	3	0	5	60%
4	0	0	4	4	5	40%
5	0	0	0	0	0	0%
6	0	0	0	4	0	20%
7	1	2	0	4	5	80%
8	1	4	0	4	5	60%
9	0	0	0	0	0	0%
10	1	2	3,4	5	5	60%
Average	60%	40%	15%	60%	60%	

Summary for Both Models for Manual Testing

The analysis of both the Reef Knot and Bowline Models further shows the necessity to enhance the dataset's diversity, particularly in terms of background variations and, crucially, adapting to different lighting conditions. An additional noteworthy observation from the evaluation was the significant impact of camera and picture quality, as exemplified in Environment 5. The introduction of blue light and the resulting grainy texture had an identical adverse effect on both models, resulting in a predictability rate of 0% across all ten scenarios.

Many of the challenges encountered during testing were attributed to the issue of hands obstructing the camera's partial view of the knot. This particular scenario can be addressed effectively in real-life usage, as live feedback would promptly rectify such issues for the user. In addition, another factor to the mixed results observed among steps 2, 3, 4, and 5 of the bowline was the presence of a distinct loop in each of these steps. This unique feature might have influenced the model's predictive accuracy, further highlighting the complexity of knot recognition in these specific instances.

4.4 Real-Time Prototype Results

The proof-of-concept prototype as seen below in Figure 4.13, provided significant insights into the real-time step prediction process within the created models. Through manual testing, a notable observation emerged - that the hands frequently obstructed the critical central portion of the knot. The user interface design, which incorporated a camera window and real-time step guidance, shed light on the underlying causes of these issues. By enabling users to observe their hand movements, it became evident that adjustments to holding the knot could be made to optimize the camera's view, leading to improved results. These findings establish a solid foundational basis for future research and development endeavors in this domain.

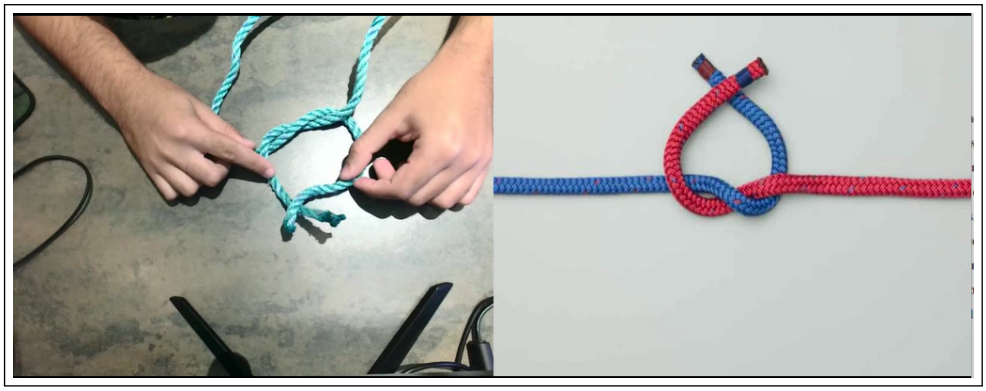

Figure 4.13: Prototype Sample Image

Chapter 5

Conclusions and Recommendations

The purpose of this study was to research the capabilities of computer vision and object detection to identify the sequential steps of a knot. This endeavor was driven by the aspiration to achieve real-time predictions with a high level of accuracy and reliability, a task that has significant implications across various domains, including maritime safety, outdoor activities, and emergency situations.

This research paper delved into the application of the YOLO algorithm for the detection of fundamental knot steps within a controlled environment. In this study, the reef knot was used as a baseline basic knot. Later on, this same model setup and structure was used to build a model for a different knot, the bowline knot, which was much more complex. This mainly was used to compare how robust the model built for the basic reef knot would differ for this new knot.

Throughout the course of this research, several key research questions were addressed. In order to create a dataset tailored to the specific novel topic at hand, a manual dataset construction process was undertaken. It's worth noting that no readily available open-source dataset was accessible for this specialized domain. The dataset employed for the Reef Knot model comprised a total of 2,800 images, while the Bowline model dataset consisted of 2,900 images. These images were distributed across five distinct classes, each corresponding to one of the five

steps of the knot. Furthermore, the dataset was subdivided into training, validation, and testing sets to facilitate model development and evaluation.

Furthermore, a proof-of-concept prototype was built for the purpose of this research, to prove that a simple state recognition prototype could be built on top of the YoloV5 algorithm. To continue evaluating the model's robustness, a comprehensive analysis of crucial metrics was conducted. Initially, these metrics were assessed by executing the 'val.py' command provided by the YoloV5 algorithm. The F1-Score was then analyzed using the plotted graphs generated through the WANDB program. Notably, the Reef Knot model exhibited an mAP of 98.8%. Meanwhile, the Bowline model achieved an mAP of 99.4%. To further scrutinize the models and assess their robustness, manual testing was undertaken to obtain more precise results. This involved the creation of ten distinct environmental scenarios for testing both the Reef Knot and Bowline models. The outcomes revealed that the Reef Knot model achieved an accuracy rate of 68%, while the Bowline model demonstrated an average correct predictability rate of 50%. However, it is important to keep in mind that having a real-time prototype for testing will greatly improve the accuracy as a video would have high fps, thus many more computations to guess the correct step.

5.1 Limitations

Due to the scarcity of readily available datasets in this subject, the necessity came to create a custom dataset. This undertaking proved to be labor-intensive and constrained the scope of this proof of concept to exclusively identify between only two types of knots. The manual compilation and cleaning of the images from the dataset taken posed a challenge, resulting in a relatively modest dataset size.

It's important to highlight that subsequent algorithms, such as YoloV7, exhibit a higher level of training intensity compared to the YoloV5 model. To effectively utilize these models, it becomes imperative as seen in Chapter 3 to synchronize multiple GPUs and concurrently execute the training process, which was not possible for this research.

5.2 Suggestions for Further Research and Improvement

A proposal for further improving this study would be to increase the variations within the dataset itself. Having extra backgrounds, different lighting conditions, and different camera qualities would greatly improve the robustness of the models, in different scenarios. Another improvement for the dataset would be to increase the number of different knots. The different knots can also be items like detecting the state recognition of fabrics to tie a tie or a bow. One can even train a model on a fishing line to have a new way fishermen learn to make knots.

One suggestion for future research is the development of a mobile application utilizing the trained models. Such an application could serve as a tool for assessing the practical effectiveness of knot teaching to novice users. Subsequent investigations into this open application could involve the collection of user feedback through questionnaires, followed by a comprehensive analysis of the gathered data. This approach would provide a valuable opportunity to refine and optimize the application's usability and instructional efficacy.

References

[1] Ravish Raj. Supervised, unsupervised and semi-supervised learning.

[2] Yu Song, Kang Yang, Xin Jiang, and Yunhui Liu. Vision based topological state recognition for deformable linear object untangling conducted in unknown background. *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2019.

[3] Wen Hao Lui and Ashutosh Saxena. Tangled: Learning to untangle ropes with rgb-d perception. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013.

[4] Priya Sundaresan and Jennifer Grannen. Untangling dense non-planar knots by learning manipulation features and recovery policies. *Robotics: Science and Systems XVII*, 2021.

[5] Muhammad Hussain, Hussain Al-Aqrabi, Muhammad Munawar, Richard Hill, and Tariq Alsboui. Domain feature mapping with yolov7 for automated edge-based pallet racking inspections. *Domain Feature Mapping with YOLOv7 for Automated Edge-Based Pallet Racking Inspections*, Sep 2022.

[6] Juan-Ignacio Pozo, María-Puy Pérez Echeverría, Beatriz Cabellos, and Daniel L. Sánchez. Teaching and learning in times of covid-19: Uses of digital technologies during school lockdowns. *Frontiers in Psychology*, 12, 2021.

[7] Karla Gutierrez. Four ways technology is changing how people learn [infographic].

[8] Webmaster. How technology has changed teaching and learning - sentinel: 9, Sep 2021.

[9] Sampurna Mandal and Ghosh Ankush. Deformable part - an overview | ScienceDirect topics.

[10] Sawan Rai. Traditional vs deep learning classification models | an empirical study!

[11] Rohith Gandhi. R-CNN, fast r-CNN, faster r-CNN, YOLO — object detection algorithms.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection.

[13] Albers Uzila. K-means clustering and principal component analysis in 10 minutes.

[14] Muhammad Hussain, Tianhua Chen, and Richard Hill. Moving toward smart manufacturing with an autonomous pallet racking inspection system based on mobilenetv2. *Journal of Manufacturing and Materials Processing*, 2022.

[15] Fahimeh Farahnakian, Lauri Koivunen, Tuomas Makila, and Jukka Heikkonen. Towards autonomous industrial warehouse inspection. *2021 26th International Conference on Automation and Computing (ICAC)*, 2021.

[16] Rui Jin and Qiang Niu. Automatic fabric defect detection based on an improved yolov5. ., Sep 2021.

[17] LittleBigCode-AI Solution Creator. Why do you need to use weights&biases to track your ML project?

Appendix A

Video to Images Converter

```
import cv2

def splitVideoIntoFrames():
    vid = cv2.VideoCapture("reef_knot/reef_knot_long.mp4")

    if(vid.isOpened() == False):
        print("Error Loading Video")

    counter = 0
    while(vid.isOpened()):
        ret, frame = vid.read()

        if ret == False:
            break

        if (counter%10 == 0):
            cv2.imwrite(f"reef_knot/all_frames/reef_{int(counter/10)}.jpg", frame)
        counter+=1

    vid.release()
    cv2.destroyAllWindows()

splitVideoIntoFrames()
```