UNIVERSITY WEST

Industrial vision system for conveyor picking of complex shapes

Shiva Tashakori

A THESIS SUBMITTED TO THE DEPARTMENT OF ENGINEERING SCIENCE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE WITH SPECIALIZATION IN AI AND AUTOMATION

AT UNIVERSITY WEST

2026

Date:	September 14, 2026
Author:	Shiva Tashakori
Examiner:	Prabhu Kettavarampalayam Ramanathan
Advisor:	Isak Brogren Lantz
Program:	Master Program in AI and Automation
Main field of study:	Automation with a specialization in AI and Automation
Credits:	60 Higher Education credits
Keyword:	Word1, Word2, Word3, Word4, Word5
Template:	University West Master Thesis template for LaTeX
Publisher:	University West, Department of Engineering Science S-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: www.hv.se

Summary

Autonomous grasping of irregular objects is a growing need in industries such as recycling and automotive parts handling, where components vary widely in shape, weight, and surface texture. A robot that can only approach objects at a fixed angle with a predefined gripper fails when the object class or its orientation changes. This thesis investigates how a vision based system can make those decisions automatically, without manual rules for each object type.
A camera, an object detector, an orientation estimator, and a machine learning classifier were integrated into a single pipeline running in real time. Trials with two object types showed that accounting for orientation reduces grasp failures, particularly for irregular shapes that land at unpredictable angles. Gripper selection driven by a model trained on empirical data generalied across object classes in a way that hard coded rules cannot.
Main finding is that combining orientation awareness with data driven gripper selection meaningfully improves grasp reliability. Rather than programming the robot for each new object, the system learns from trial outcomes and adapts. This makes it a more scalable approach for industrial environments where the range of handled objects keeps changing.

Preface

This thesis was written as part of a Master of Science program in AI and Automation at University West. The work was carried out in the robotics lab during the spring and summer of 2026.

I would like to thank Abdulrahman Al Hamwi, a fellow student in AI and Automation, for his support and involvement throughout the project. His contributions to the practical side of the work were valuable and are appreciated.

I am also grateful to my supervisor Isak Brogren Lantz and to Morgan Nilsen for their guidance and patience throughout the course of this work.

Affirmation

This master degree report, A Short descriptive title of the work, was written as part of the mas-ter degree work needed to obtain a Master of Science with specialization in Robotics degree at University West. All material in this report, that is not my own, is clearly identified and used in an appropriate and correct way. The main part of the work included in this degree project has not previously been published or used for obtaining another degree.

Signature by the author

Date

Shiva Tashakori

Contents

Summary iii

Preface iv

Affirmation v

Contents vi

Symbols and glossary 1

1 Introduction 2
1.1 Purpose 3
1.2 Aim and Objectives 3
1.3 Investigative Questions 4
1.4 Limitations 4
1.5 Outline 4

2 Literature Review 5
2.1 Real-Time Object Detection and the YOLO Family 5
2.2 Transfer Learning for Small Industrial Datasets 6
2.3 Coordinate Transformation from Image to Robot Space 7
2.4 **Object Orientation Estimation** 7
2.5 **Grasping Challenges in Practice** 8
2.6 **End Effector Types and Object Compatibility** 8
2.7 **Machine Learning for Gripper Selection** 9

3 Method 11
3.1 Research Strategy 11
3.1.1 Approach 11
3.1.2 Research Aims 11
3.2 System Architecture 12
3.2.1 Modules and Interfaces 12
3.3 Hardware Setup 13
3.3.1 Camera 13
3.3.2 Robot 13
3.3.3 Grippers 14
3.3.4 Background Surface 14
3.4 Data Collection and Labelling 15

Contents

3.4.1 Dataset . . . 15
3.4.2 Relation to Benchmark Data . . . 15
3.5 Object Detection . . . 16
3.5.1 From YOLOv8 to YOLO11 . . . 16
3.5.2 Transfer Learning from a Limited Dataset . . . 16
3.6 Coordinate Transformation . . . 17
3.6.1 Linear Calibration . . . 17
3.6.2 Homography-Based Calibration . . . 17
3.7 Grasp Orientation . . . 18
3.7.1 Position in the Pipeline . . . 18
3.7.2 Two Configurations Compared . . . 19
3.8 Robot Communication and Control . . . 19
3.8.1 Setup Redefined Each Session . . . 19
3.8.2 Orientation in the Message . . . 19
3.9 Challenges and Limitations . . . 19
3.9.1 Object Availability . . . 20
3.9.2 Depth Images . . . 20
3.9.3 Variable Lighting . . . 20

4 Implementation . . . 22
4.1 Training Run . . . 22
4.2 Software Setup . . . 22
4.3 Robot Side Message Handling . . . 22

5 Results and Discussion . . . 24
5.1 YOLO Performance . . . 24
5.1.1 Linear Calibration . . . 27
5.1.2 Homography Calibration . . . 28
5.1.3 YOLO11 with Homography Calibration and XGBoost . . . 29
5.2 Discussion . . . 30

6 Conclusion . . . 32
6.1 Future Work and Research . . . 32

Bibliography . . . 34

Symbols and glossary

OPC	An open standard for communication of real-time plant data between con-trol devices from different manufacturers. Typically, bidirectional communi-cation between a PLC and a computer. Formerly, but no longer, an acronym for Object Linking and Embedding for Process Control.
knock knock	Who is there

1. Introduction

Picking and placing irregular metal objects is one of the parts of recycling that has stayed still manual. A production line can move material, weigh it and cut it automatically, but deciding what an individual piece is and then physically picking it out of a stream of mixed material is still work that people do by hand in many facilities. The pieces themselves are the reason. Scrap does not arrive in neat shapes with flat faces and known dimensions, it arrives bent, tangled, flattened and broken, and no two pieces of it are quite the same.

Industrial robots handle repetition extremely well and handle variation poorly. A standard pick and place cell works because the part arrives in the same place, in the same orientation, every cycle, so the robot can be taught one position and repeat it indefinitely. Remove that guarantee and the whole approach stops working, since there is no fixed position left to teach. Any robot expected to pick irregular scrap therefore needs to work out for itself where each object is and how to approach it, which turns a mechanical problem into a perception problem.

This is not only a practical inconvenience. A recent review of learning-based robotic grasping notes that most solutions on the market still rely on traditional object recognition and are not suited to grasping unknown objects with varying shapes and textures, which is exactly the situation in high-mix, low-volume settings where conventional pre-programmed robots need extensive reprogramming for every new part [1]. Metal scrap recycling is a concrete example of that gap rather than an edge case. Recent work on automated aluminium grasping shows facilities being actively redesigned around sensor-driven, adaptive sorting because manual and fixed-rule sorting cannot keep pace with the variability of incoming material [2]. This project addresses that same variability at the picking stage, where an individual object still has to be located and grasped rather than only classified.

This project is a part of the Super Sorter initiative at Mission 0 House, which is concerned with automating exactly this stage of the recycling process. The work reported here was carried out alongside a fellow student who ran a comparative study of three different end effectors, a mechanical two finger gripper, an electrical gripper and a suction gripper, on the same family of irregular metal objects. That study answers which gripper suits which object. It does not answer where the robot should go or how it should approach, and that gap is what this thesis addresses.

Building a vision system to close that gap raises a question that turns out to be more interesting than it first appears. Finding an object in a camera image is a solved problem in the sense that trained detection models do it reliably. Turning that detection into a position the robot can actually reach is where the difficulty lives, because the camera and the robot do not share a coordinate system and the relationship between them depends on where the camera happens to be looking from. A camera mounted directly above a table behaves differently from one mounted to the side at an angle, and getting that relationship wrong produces failures that look random from the outside while following a consistent pattern underneath.

The system described in this thesis was built in stages around a Kinect v2 camera, a

YOLO11 detection model trained on images collected in the laboratory, a coordinate transform, and an ABB IRB 1600 industrial robot programmed in RAPID. Each stage was added only once the one before it behaved predictably, so that when something failed it was usually clear which part was responsible. A machine learning model was added last, on top of the working vision pipeline, to predict how the gripper should be turned before it closes, since position on its own says nothing about how an object is lying.

1.1 Purpose

The purpose of this degree project is to explore how an AI-based vision system can improve robotic picking in situations where objects differ in shape, size, and physical properties. In industrial environments, successful picking depends not only on locating the object but also on selecting an appropriate grasping strategy and end effector, and getting that combination wrong is what turns a routine pick into a failed one.

This project matters from an industrial perspective because unsuccessful picks lead to production delays, rework, and reduced system reliability, and because industrial recycling in particular is already moving toward automated, sensor-driven sorting facilities that depend on this kind of reliable picking [2]. By studying both successful and failed picking attempts in a laboratory environment using an ABB robot, this work aims to better understand how vision-based information can be used to increase the success rate. From a scientific perspective, the project contributes practical insights into how machine learning-based vision systems perform when applied to real robotic systems rather than controlled benchmark scenarios, which recent grasping literature still identifies as largely unsolved for irregular, previously unseen objects [1].

1.2 Aim and Objectives

The main aim of this degree project is to investigate how an AI-based vision system can support more reliable robotic picking by providing more suitable grasp point estimates, leading to fewer failed pick attempts when especially handling objects with irregular shapes.

The specific objectives are to:

- Create a pipeline to build an AI model to identify picking positions of complex objects.
- Investigate how an AI-based vision system can be integrated into an industrial PLC environment to support a robotic picking application.
- Identify bottlenecks for reconfiguring AI models between different types of end-effectors of the robot.

1.3 Investigative Questions

To fulfil the purpose and aim, the following investigative questions have been formulated:

1. How can an AI-based vision system be integrated into an industrial PLC-controlled environment to support robotic picking of complex objects?

2. How does object-gripper compatibility affect picking outcomes when using the same AI-based vision pipeline?

1.4 Limitations

This degree project is limited to a five-month timeframe and laboratory-based experiments using a restricted set of metal objects and available end-effectors. The evaluation is performed under controlled conditions with a limited dataset, which constrains the amount of model adaptation and the range of object geometries that can be examined.

Large-scale data collection, continuous online learning, long-term system optimization, and full industrial deployment are not included in this project, and the results should therefore be interpreted as laboratory-level findings rather than full industrial validation. The coordinate transformation used in the final pipeline is a homography fitted to a single working plane, which is discussed together with its own boundaries in Chapter 3; extreme lens distortion correction is considered beyond the immediate scope.

1.5 Outline

Chapter 2 reviews existing work on vision guided robotic grasping, covering object detection, the estimation of position and orientation from images, and the ways other authors have approached picking objects that do not have a fixed shape.

Chapter 3 describes the method. It covers the hardware, the dataset and how the detection model was trained, the two coordinate transformation methods that were tried, the machine learning step that predicts approach angle, and the communication between the vision software and the robot controller. The reasoning behind each choice is given alongside the choice itself, together with the problems that appeared along the way.

Chapter 4 reports the testing. It begins with how well the detector performed on data it had not seen, then works through three complete grasping runs carried out on the physical robot, each one changing a single part of the pipeline so that the effect of that change could be seen. The chapter closes by interpreting these results together and separating what has been demonstrated from what has only been made to work.

Chapter 5 concludes, summarising what the system achieved, what its limits are, and what should be tested next.

2. Literature Review

This chapter reviews the theoretical background and related research within industrial computer vision and robotic grasping. It covers the transition from traditional feature-based pipelines to modern appearance-based deep learning methods, highlighting the computational methods required for successful pick-and-place automation.

2.1 Real-Time Object Detection and the YOLO Family

A sorting robot on a conveyor belt has very little time to decide what it is looking at. The detection algorithm must find the object, classify it, and return a position fast enough for the arm to react. This is the problem the YOLO family was built to solve.

Object detection means finding every object in an image and placing a labelled box around each one. Early methods scanned the image many times with a sliding window, which was too slow for real time use. YOLO changed that by treating detection as a single step: one pass through a neural network produces all boxes and labels at once [3]. The name, You Only Look Once, describes the idea directly. YOLOv8, released by Ultralytics in 2023, switched to an anchor free detection head. Each predicted box is described relative to its centre point instead of a preset template, which removed the need to manually tune anchor sizes. YOLOv8 reached an mAP of 53.9 percent on the COCO benchmark and became one of the most widely used detectors in production systems [4].

YOLO11, released in September 2024, kept the anchor free design but replaced the internal building blocks. The C2f module was replaced with C3k2, which extracts features with fewer parameters. An SPPF block handles multiple scales, and a C2PSA block adds spatial attention so the model focuses on the most important regions of the image [5]. The result is higher accuracy with a smaller model compared to YOLOv8 [6]. A study on orchard imagery confirmed that YOLO11 outperforms YOLOv8 on small and partially occluded objects [7].

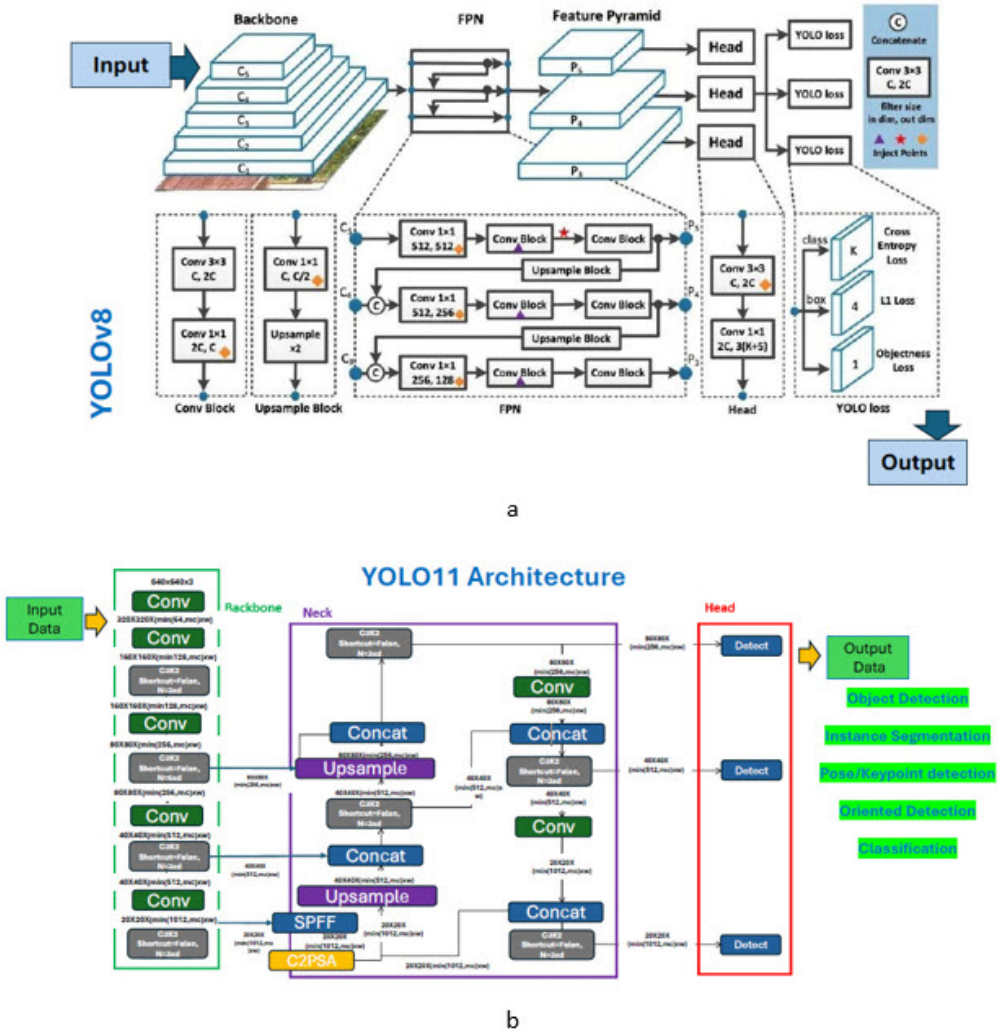

Figure 2.1: (a) Architecture of YOLOv8 showing the backbone, FPN and anchor-free detection head vs (b) YOLO11 structure with C3K2 blocks, SPPF and C2PSA attention module [7].

2.2 Transfer Learning for Small Industrial Datasets

Only 232 images were available to train the object detector in this project. Most deep learning models need tens of thousands. Transfer learning is the reason a small dataset like this is still enough.

Training a deep network from scratch requires enough data for the model to first learn basic visual patterns like edges and textures before it can recognise specific objects. Transfer learning skips that step. The model starts with weights already trained on a large general dataset, so the low-level visual knowledge is already there. Only the later layers are updated on the new, smaller dataset [8]. YOLO models pretrained on COCO, which has 80 object classes and roughly 330,000 images, already understand what shapes and surfaces look like. Fine-tuning on the 232 metal scrap images teaches the final layers to recognise the six component classes while

leaving the general visual knowledge unchanged [9]. The model trains faster, needs less data, and generalises better. A study on using YOLO with transfer learning for satellite imagery found that pretrained models strongly outperform scratch-trained ones when the training set is small [10]. The same effect was shown for industrial defect detection with only a few hundred samples [11]. Both results support the approach used here.

2.3 Coordinate Transformation from Image to Robot Space

The camera sees an object at a pixel position. The robot arm needs a position in millimetres. These two coordinate systems are completely different, and connecting them correctly decides whether the gripper lands on the part or misses it. A camera produces pixel coordinates. A robot arm works in millimetres relative to its base frame. The mapping between the two is called hand-eye calibration when the camera is placed near the robot [12]. The calibration finds the camera intrinsics such as focal length and optical centre, and the geometric relationship between the camera and robot frames. For a fixed overhead camera with no perspective distortion at the picking surface, the transformation simplifies to a scale factor that converts pixels to millimetres and an offset that corrects for the physical gap between the camera and the robot origin [13]. When the camera is rotated relative to the robot axes, the detected pixel coordinates are rotated around the image centre before scaling. Any error in this calibration shifts every grasp by the same fixed amount. A 5 mm calibration error means every pick is off by 5 mm regardless of how well the detector performed. Research on industrial pick and place systems identifies calibration error as one of the most common causes of consistent grasp failure [14].

2.4 Object Orientation Estimation

A robot that always approaches at the same fixed angle works fine for symmetric objects. For anything elongated or irregular, the gripper often contacts the wrong part of the object. Measuring orientation before the arm moves is what prevents this.

When the robot wrist does not rotate to match the object, the contact geometry changes with every pose the object lands in. A gripper approaching a wire coil at the wrong angle may catch one finger inside the coil and leave the other outside, producing no useful grip [15]. Estimating the object angle and rotating the wrist to match is the direct fix.

Three methods were studied in this project [16]. The bounding box aspect ratio method classifies an object as horizontal if its box is wider than tall, and vertical otherwise. It requires almost no computation but gives only two possible orientations, 0 degrees and 90 degrees. Objects at any other angle are misclassified.

Contour analysis fits a minimum-area rotated rectangle around the detected object boundary. The angle of the longest side gives a continuous orientation value between

0 and 180 degrees. Because the rectangle fits the actual edge of the object, it works reliably on irregular shapes [16].

Principal Component Analysis applies eigenvalue decomposition to the pixel positions of the segmented object. The direction of the largest eigenvalue gives the dominant orientation axis. PCA is the most precise of the three and works even on partially occluded or deformable shapes, but it requires more computation than the other methods [16].

Figure 2.3 — Left: the axis-aligned bounding box gives only 0 or 90 degrees. Centre: contour analysis gives a continuous angle by fitting a rotated rectangle to the object edge. Right: PCA finds the dominant axis with highest precision[16].

YOLO11 also supports oriented bounding boxes natively, outputting the object angle as part of the detection result without any separate post-processing step [17]. For a conveyor based system where speed matters, this is a useful option.

2.5 Grasping Challenges in Practice

Robotic grasping that works well on a test bench can fail repeatedly in a real lab. Metal scrap is reflective, sharp edged, and lands in unpredictable poses. Knowing why grasps fail is just as important as knowing how to plan them.

Irregular shapes are the main difficulty. An asymmetric bracket does not have a single stable grasp point that works from every approach direction. The geometric centre of the detected bounding box may sit over a hole or a sharp edge that blocks the fingers [17].

Orientation mismatch causes many failures for finger based grippers. When the robot approaches at a fixed wrist angle and the object is rotated, one finger may land on the object while the other misses it. The grip force then acts along an axis that spins the object rather than lifting it [15].

Calibration error is systematic rather than random, which makes it easy to overlook. Benali et al. report that a constant offset in the pixel to millimetre mapping shifts every grasp attempt by a similar amount, so individual failures can look unrelated even though a consistent pattern emerges once success rates are analysed across many trials [14].

Deformable objects add a further difficulty, since their shape is not fixed between trials. Billard and Kragic note that a flexible object such as a wire coil can settle into different configurations, for example a flat loop or a tangled pile, and that a single fixed grasp strategy is unlikely to succeed across all of them [18]. Reflective metal surfaces also confuse depth sensors, adding noise to the position estimate.

2.6 End Effector Types and Object Compatibility

There is no single gripper that works for all objects. A suction cup lifts a flat aluminium sheet with no effort but cannot grip a wire coil at all. A rigid jaw

gripper handles heavy irregular parts but fails on thin flat plates. Matching the gripper to the object is the core decision this project automates.

Vacuum suction grippers create a pressure difference between a sealed pad and the object surface. They are fast and impose no clamping force, which makes them ideal for large flat smooth surfaces like sheet metal or cardboard. If the surface is curved, porous, or smaller than the pad, the seal breaks and the holding force drops to zero [19].

Mechanical parallel jaw grippers close two rigid fingers around the object and hold it by friction. They handle porous, rough, and irregular surfaces that a suction cup cannot grip, and they can exert much higher forces. The constraint is that both fingers must find a usable contact surface within the jaw stroke [19].

Soft pneumatic grippers use flexible silicone fingers that deform around the object. They conform to curves and irregular shapes that rigid jaws cannot engage, and they spread the contact force over a larger area which reduces slipping. Their payload limit is lower and their fingers are vulnerable to sharp metal edges [18].

The Amazon Robotics Challenge showed the practical result of this trade off. The winning system used a suction gripper for most objects and switched to a parallel jaw gripper for porous or flexible items that could not be sucked [20]. Handling the widest variety of objects required selecting the gripper based on what the object actually was.

Figure 2.2: *The three end-effectors used in this project: OnRobot RG6 (left), Piab Kenos MiniGrip (centre), and Piab piSOFTGRIP 50-3 (right). [16]*

2.7 Machine Learning for Gripper Selection

Choosing a gripper by hand for every object works when the number of object types is small, but does not scale well as variety increases. Learning the gripper decision from labelled trial outcomes is one way around this, and XGBoost is well suited to it because the training data is a structured table with a modest number of rows.

A rule based approach such as if the object is flat use vacuum works for simple cases. It breaks down when the best gripper depends on a combination of features that is hard to write as a rule, or when a new object does not fit any predefined category. A learned model captures those interactions directly from the outcome data [16].

XGBoost stands for Extreme Gradient Boosting. It builds the final model by adding decision trees one at a time. Each new tree is trained specifically on the prediction errors that all previous trees made together [21]. The loss function has two parts: one measures prediction error on the training data, and one penalises model complexity to prevent overfitting on small datasets.

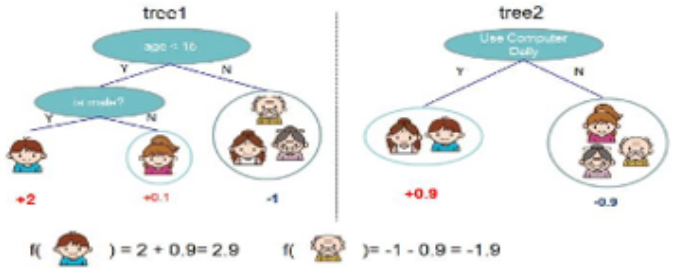

Figure 2.3: *Tree Ensemble Model used in XGBoost: the final prediction for any example is the sum of scores from all trees. Source: Chen and Guestrin [21].*

XGBoost handles missing values without manual imputation, builds trees in parallel across CPU cores, and produces feature importance scores that show which inputs drove the decision most [22]. In an industrial setting it is useful to know not just which gripper was selected but why, for example that object mass was the deciding factor.

Across large tabular benchmarks, tree-based models including XGBoost consistently match or outperform deep neural networks when the dataset contains fewer than roughly 10,000 samples [23]. This project's own gripper-selection dataset, described in Chapter 3, is well within that range, which supports XGBoost as a reasonable choice for this task.

3. Method

Building a robotic picking system that works reliably on irregular metal parts required several things to come together: a way to see the object, a way to translate that view into robot coordinates, a way to judge how to grip it, and a robot program that could act on all of that. Each of these pieces is covered in this chapter, along with the reasoning behind the choices made and the problems that came up along the way.

3.1 Research Strategy

Deciding how to test a system like this shapes everything that follows, so it is worth stating plainly before any technical detail. The strategy set out here follows directly from the aims defined in the original project proposal, and the subsections below cover the overall approach taken and how that approach answers each of those aims.

3.1.1 Approach

An experimental engineering approach guides this project. Rather than testing a theoretical hypothesis, the goal is to build a working system and observe how it behaves under real conditions. Grasp success or failure on real metal scrap objects is the measure used throughout, following the methodology described by [24] , in which engineering experiments recreate physical conditions in a controlled setting and measure results systematically.

3.1.2 Research Aims

Setting out to investigate how an AI based vision system could support more reliable robotic picking of irregularly shaped objects, the original project proposal identified a main aim and two sub-aims. Revisiting each one against the strategy actually followed is worth doing directly here, before the technical detail of later sections, rather than leaving that connection implicit.

Providing more suitable grasp point estimates to reduce failed pick attempts on irregular objects was the central aim, and the pipeline built here addresses it in two parts. Detecting the object and locating it accurately in robot coordinates falls to YOLO11 combined with the homography based coordinate transform described in Section 3.7, while estimating how to approach an irregular object falls to the XGBoost orientation step in Section 3.8. Together these give the robot both a position and an angle rather than position alone, which is the concrete mechanism through which this aim was pursued.

Building a pipeline that identifies picking positions for complex objects, the first sub-aim, is addressed directly by the system architecture in Section 3.2 and the detection and coordinate transformation stages that follow it. Locating an object in

the camera frame and translating that location into a usable robot coordinate is exactly what the object detection, system integration, and coordi nate transformation sections describe, so this sub-aim is met by the vision side of the pipeline as built.

Identifying bottlenecks in reconfiguring AI models between different end effectors, the second sub-aim, was approached indirectly rather than through a dedicated, focused investigation. The system was designed from the outset to work with three gripper types, mechanical, electrical, and suction, through a single consistent message format from the vision side, with gripper-specific handling such as tool data and spatial offsets kept on the robot side. This design choice avoided the need for separate vision code per gripper, but a systematic study of what specifically makes reconfiguration harder or easier between gripper types was not carried out as its own piece of work.

Building the system in stages made it possible to isolate problems before they compounded, and Figure 1.1 shows the resulting pipeline in the order each stage was added. Object detection came first, since every later step depends on a reliable detection output. Orientation estimation and coordinate transformation followed, once detection was stable. Gripper decision making through XGBoost came last, added only after the vision side of the pipeline behaved predictably. This order also meant that when something failed, it was usually clear which stage was responsible: camera capture and detection feed into the coordinate transform and orientation prediction, which together produce the message sent to the robot over TCP/IP, where the ABB controller running RAPID parses it and carries out the grasp.

3.2 System Architecture

Splitting the system into independent modules made it possible to build, test, and later change each part without the others breaking. The subsection below describes what those modules are and the single message format that ties them together.

Figure 3.1: *System pipeline [16]*

3.2.1 Modules and Interfaces

Three modules make up the system, each communicating through a defined in terface rather than sharing internal state. A Python vision and decision pipeline runs on a PC connected to the Kinect camera. An XGBoost decision model is loaded at startup and called at runtime to predict grasp orientation. An ABB IRB robot, running RAPID, receives instructions over TCP/IP and executes the physical grasp.

Keeping these modules separate meant the vision code, the decision model, and the robot controller could each be changed without touching the others. When the calibration method changed later in development, for instance, nothing in the RAPID

program needed to change at all. A single formatted string ties these modules together at runtime, travelling from the Python side to the robot for every detected object: object class, x position in millimetres, y position in millimetres, and orientation angle in degrees. The robot parses this string and acts on it directly, without needing to know how any of those four values were produced.

3.3 Hardware Setup

Three physical components carry the weight of the picking system: a camera to see the object, a robot arm to move and grip it, and a background surface that keeps detection conditions consistent between sessions. The subsections below cover each in turn, along with the different grippers the system was built to support.

3.3.1 Camera

A Microsoft Kinect v2 provides the vision input, mounted on a fixed stand positioned beside the robot rather than directly above the work surface. Looking down and across the table at an angle toward the robot base, rather than straight down, follows from the physical layout of the lab space available. That angled placement has a direct consequence for the coordinate transform, covered in Section 3.7. Since the Kinect sits on a portable mount rather than a fixed rig, it gets repositioned at the start of each lab session, which means recalibration happens every time the system is set up.

3.3.2 Robot

An ABB IRB 1600-6/1.2 carries out the physical grasping, a six axis industrial arm rated for a 250 kilogram payload, installed at University West's robotics laboratory. Communication with the vision system runs over a TCP/IP socket connection, with the grasp sequence itself written in RAPID, described in Section 3.9.

Figure 3.2: *The ABB IRB 1600-6/1.2 and the working environment*

Figure 3.2. depicts the working environment of the practical tests. The robot arm with a mechanical end effector is mounted on a fixed base at one end of the table. On the opposite side, a Microsoft Kinect v2 sits on a portable stand at approximately 146 cm height, which has three mounting legs on the floor to prevent movement during trials. The camera was tilted to its maximum possible angle toward the work surface to capture the full picking area.

3.3.3 Grippers

Picking demands different tools depending on the object, so three types of end effector are used across the project: a mechanical gripper with two fingers, an electrical gripper, and a suction gripper. One consistent message format from the vision system works for all three, with the choice of which gripper to use handled on the robot side rather than requiring separate vision code per tool.

3.3.4 Background Surface

Bringing the working surface closer to the conditions the object detection model was trained on called for a simple fix: a sheet of A3 paper placed under the object on the table for every trial. A flat, uniformly lit background results, similar in appearance to the training images collected by a fellow student, and the sheet's four corners double as fixed, known reference points for the calibration procedure described in Section 3.7.

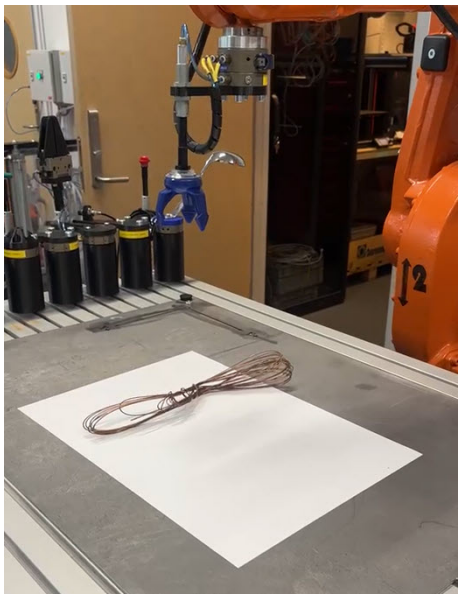

Figure 3.3: *ackground surface setup during a live trial*

Figure 3.3 shows an A3 sheet is placed under the object for every trial. The sheet color matches the background in the training images, moreover its four corners are used as reference points for calibration.

3.4 Data Collection and Labelling

Every detection model needs lablled examples to learn from, and this project's came from a fellow student's collection and annotation effort rather than a public dataset. The size and origin of that dataset are worth stating clearly before the training procedure itself is described in the next section.

3.4.1 Dataset

Object detection relies on a dataset of 232 labelled images [16], showing six component classes placed on the working surface in varying positions and orien tations. A fellow student carried out both the image collection and the manual bounding box labelling for this dataset.

3.4.2 Relation to Benchmark Data

Placing this dataset in context matters, since 232 images is small compared with the datasets typically used to pretrain detection models. Figure 1.4 shows a representative example from the COCO dataset [25] beside a labelled example from the project's own 232 image set, making the difference in scale and visual domain between a general purpose benchmark and a small, task specific dataset visible directly.

3.5 Object Detection

Detecting the object correctly is the foundation every later stage of the pipeline depends on, so the choice of model and the training strategy behind it both matter here. The following subsections walk through the version used, why transfer learning made sense given the dataset size, and how training was actually carried out.

3.5.1 From YOLOv8 to YOLO11

YOLOv8 served as the object detection model at the start of development, chosen at the time as a well established, actively maintained single stage detector with strong real time performance, properties that matter directly for a system meant to detect objects on a moving conveyor and hand off a grasp point to a robot with minimal delay. Development later moved to YOLO11, the newer release in the same model family, once it became available partway through the project.

Three considerations motivated the switch rather than staying with YOLOv8. Ultralytics reports measurable gains for YOLO11 over YOLOv8 on standard benchmarks in both detection accuracy and inference speed, which matters directly for a real time picking application where a slower or less accurate detector translates into missed or delayed grasp attempts [6]. YOLO11 also keeps close API compatibility with the YOLOv8 toolchain, so the switch did not require rebuilding the training pipeline, the dataset format, or the inference code around the model, only retraining on the existing annotated dataset. Finally, since YOLO11 was still actively supported and maintained at the time of the switch, adopting it reduced the risk of building the rest of the system around a detector approaching the end of active development. Because the change was a version upgrade rather than a change in approach, the transfer learning process and training procedure described below applied to both versions without modification.

3.5.2 Transfer Learning from a Limited Dataset

Only 232 labelled images were available for object detection, described in the previous section, and that scarcity is what rules out training a detection model from randomly initialised weights. A convolutional network trained from scratch on a dataset this small would have little chance of learning generalisable visual features on its own, since deep object detectors typically require datasets orders of magnitude larger to converge reliably without overfitting.

Starting instead from a COCO pretrained YOLO11 checkpoint sidesteps this problem. Because the model already encodes general visual features such as edges, textures, and object boundaries learned from a much larger and more varied source dataset, only the task specific layers need to be retrained to recognise the six component classes relevant here [8]. Retraining just those layers on the 232 image dataset proved sufficient to reach useful detection performance, which would not have been a realistic outcome from random initialisation given the same amount of data.

3.6 Coordinate Transformation

Turning a pixel position into a real world robot coordinate is where an angled, non top down camera creates its biggest problem, and the two subsections below trace that problem from its first, simpler attempted solution through to the fix that actually worked.

3.6.1 Linear Calibration

A fixed scale factor and a single fixed rotation angle, applied the same way across the entire image, defined the first version of the coordinate transform: a pixel centre offset (KAMERA MITT X, KAMERA MITT camera axis and the robot origin (KAMERANS a scale factor (PIXEL TILL Y), a physical offset between the FYSISKA X, KAMERANS FYSISKA MM), and a fixed camera rotation angle (KAMERA Y), ROTATION Underlying this is a linear model, one that assumes a constant relationship be tween a movement in pixel space and the corresponding movement in real world space, regardless of where in the image the object appears.

That assumption only holds when the camera looks straight down at the work surface, which is not the case here. Mounting the Kinect at an angle introduces a perspective effect known as foreshortening: objects further from the camera appear smaller, and their apparent position shifts non uniformly across the image. A linear model cannot represent this, so its accuracy depended heavily on how close an object was to the point it had been calibrated against. Grasp attempts near that reference point tended to succeed, while attempts further away accumulated position error, in several cases causing the gripper to close on the wrong part of the object or lose it during the lift and transfer motion. Residual calibration error of this kind shifts every grasp attempt by a similar amount, making individual failures look random while producing a consistent pattern across many trials [14]. A lab recording of a trial under this calibration method shows the effect directly: the copper coil, initially resting on the A3 sheet, gets picked up by the mechanical gripper but ends up off the sheet on the bare table rather than at the intended drop location, having shifted out of the gripper's grasp before the transfer completed.

3.6.2 Homography-Based Calibration

Correcting this called for a different kind of transform altogether: a homography, a projective transformation that maps points from one plane to another and can represent the perspective distortion an angled camera introduces, unlike a linear model. OpenCV's cv2.findHomography computes it from four known point correspondences, the pixel coordinates and corresponding real world coordinates of the four corners of the A3 sheet placed under the object. Since the sheet's physical size and position are known, its four corners give four reliable reference points spread across the working area rather than a single calibration point.

Repositioning the camera on its stand between lab sessions means this cal ibration step runs once at the start of each session. The resulting matrix is stored in calibration.json and loaded by the live pipeline, which applies it to every detected object

position before sending it to the robot.

Grasp accuracy improved noticeably across the working area rather than only near a single point once this change was in place. During one test session, the object was deliberately moved to different positions on the sheet between attempts, and nine of ten attempts succeeded under the homography based calibration.

3.7 Grasp Orientation

Knowing where an object is only solves half the grasping problem, since irregular parts also need to be approached at the right angle. Position alone is not enough to grip the copper coil and similar objects reli ably, since they can lie on the table at any angle. A gradient boosted decision tree trained with XGBoost [21] handles this by predicting a grasp angle directly from properties of the detected bounding box: the object's pixel position and the width and height of its bounding box. Small tabular datasets are exactly where XGBoost tends to perform well [23], and its built in regularisation keeps overfitting risk manageable given the limited number of trial records available [22].

Three other methods for estimating object orientation were considered before settling on this approach. Each differs in how precisely it measures the angle, how fast it runs, and how well it handles objects with irregular shapes. Table 3.1 summarises these differences. The learned model was chosen over all three because it needs no segmentation, no contour extraction and no depth data, working only from values the detector already produces.

Table 3.1: Comparison of three orientation estimation methods.

Method	**Precision**	**Speed**	**Handles irregular shapes**
Bounding box aspect ratio	Low (0 or 90 deg only)	Fastest	Acceptable
Contour analysis	High (continuous 0 to 180 deg)	Fast enough for real time	Good
PCA	Highest (sub-degree)	Slowest	Excellent

Although PCA offers the highest precision, it is computationally expensive for a real time pipeline, and contour analysis requires extracting the object boundary before an angle can be fitted. In this project a learned model was selected instead, taking the bounding box values that YOLO11 already produces and predicting the grasp angle from them. This avoids segmentation and boundary extraction entirely, which keeps the step fast enough for real time use.

3.7.1 Position in the Pipeline

Detection features flow straight from YOLO11's bounding box output into the XGBoost model once an object is detected, and the model returns a predicted orienta-

tion angle in response. Combined with the object's class and its trans formed x and y position, that angle becomes part of a single message sent to the robot over the socket connection, in the form material,x mm,y m,angle.

3.7.2 Two Configurations Compared

Two complete pipelines emerged from this development process and were tested as such. Linear calibration paired with YOLO11 came first and did not in clude an orientation prediction step. Homography based calibration paired with YOLO11 and the XGBoost orientation step came second, giving the robot both a position and an angle for each grasp attempt rather than position alone.

3.8 Robot Communication and Control

Everything the vision pipeline computes eventually has to reach the robot and become physical motion, and that final translation happens in the robot's own programming language, RAPID. What gets set up fresh each session and how the orientation value changed the message format are covered below.

3.8.1 Setup Redefined Each Session

Connecting to the robot happens the same way every time the pipeline starts: the Python script opens a TCP/IP connection to the ABB robot at startup, while the RAPID program on the robot side creates a server socket, listens on port 1025, and blocks until the Python client connects. Once that connection is made, the robot enters a continuous loop waiting for messages.

Recalibration is not the only thing that happens fresh each lab session. Be cause the camera gets repositioned and the robot's tool and work object frames are recalibrated at the same time, the tool data (tool1, graspig tool, tool2) and the work object (GRASPING) used by the RAPID program are redefined and verified each time the system is set up, rather than assumed to remain valid between sessions.

3.8.2 Orientation in the Message

Four fields now travel in the RAPID program's incoming message instead of three, once the XGBoost step contributes an orientation value: object class, x position, y position, and orientation angle, parsed from the socket string. Build ing the robot's target position and rotation for the grasp uses this data directly, pPick.rot := OrientZYX(nOrientation, 0, 0), before the same approach, grip, lift, and return sequence executes.

3.9 Challenges and Limitations

Three specific issues shaped what could be tested and how well the system performed: limited access to trained objects, dropping depth data entirely, and variable

lighting in the lab.

3.9.1 Object Availability

Access to trained objects was not always available during testing, which shaped what could actually be evaluated at any given time. Development and compari son work centred on two objects, a ruler picked up with the suction gripper and a copper coil picked up with the mechanical gripper. The trained copper coil itself was not available in the lab for the calibration validation run described in Section 3.7, so a similar coil shaped wire object was substituted instead, shown in Figure 1.5. Detecting it with YOLO11 still worked, though with a noticeable delay compared with detecting the trained object directly. Objects extending partly beyond the A3 sheet, or reshaped so their form differed substantially from the trained copper coil, went undetected entirely.

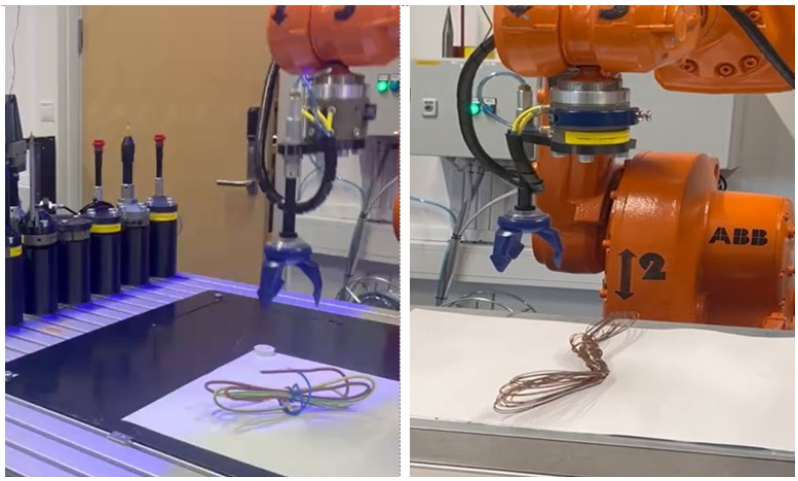

Figure 3.4: *Wire object substituted for the copper coil during calibration*

Figure 3.4 shows alternative of the object 1 or copper coil. YOLO11 still detected it, though with a slight delay compared to the trained object.

3.9.2 Depth Images

Depth information from the Kinect camera went unused in the current system, despite being available. Reflective metal surfaces produced noisy or low quality depth values in preliminary tests, making depth measurements unreliable for the metallic objects this project focuses on, so the decision was made to rely on 2D detection and the coordinate transform alone rather than incorporate depth data of poor quality.

3.9.3 Variable Lighting

Lighting in the lab was not constant, with lamps flickering intermittently during testing. Changes in illumination caused the detection model to occasionally interpret light reflecting off the bare table as an object and attempt a grasp on it, a

known weakness of vision based detection in industrial settings, where illumination changes and surface reflections reduce robustness and can trigger false detections [26]. Figure 3.5 shows the ceiling mounted lighting fixtures in the lab responsible for this variability, and Figure 3.4 shows the effect directly during a trial: a reflection on the table is picked up by the detection pipeline and the pixel coordinates for a false detection are sent toward the robot. Placing an A3 sheet under the object gave a consistent, non reflective background across the detection area and addressed the issue in practice.

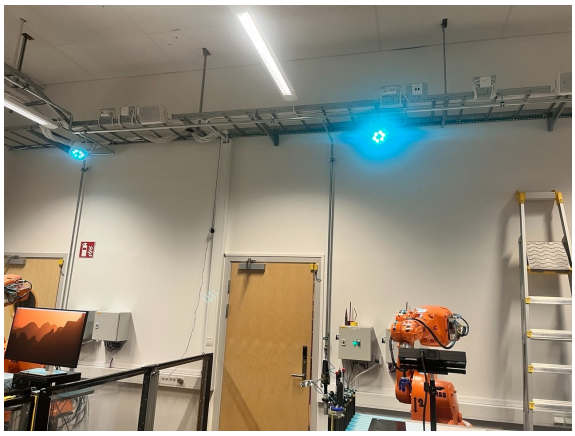

Figure 3.5: *Ceiling mounted lighting in the lab, showing the variable illumina tion sources responsible for reflections on the bare table.*

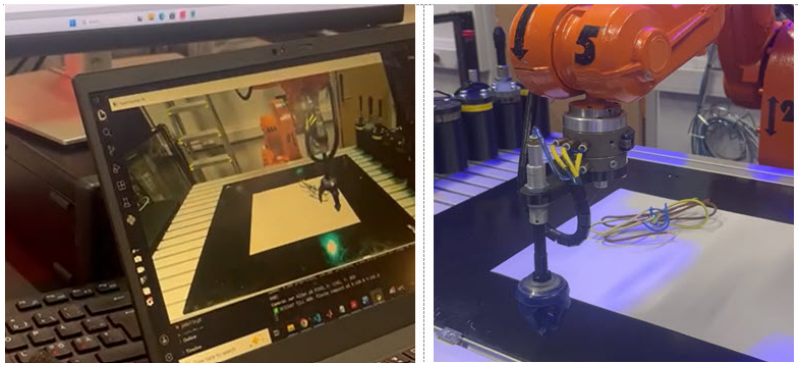

Figure 3.6: *Light reflection detected as an object during a live tria*

The detection pipeline picked up a light reflection from the bare table surface and treated it as a real object, sending incorrect coordinates to the robot. Placing the A3 sheet under the object reduced this effect by covering the reflective metal surface.

4. Implementation

This chapter covers the practical work needed to get the vision pipeline actually running, separate from the research decisions covered in the Method chapter. It includes the training run itself, the software setup problems that had to be solved before the pipeline could work at all, and the low level detail of how the robot side parses and acts on each message.

4.1 Training Run

The detection model was trained on the 232 image dataset described earlier, using all 232 images for training. No portion of the data was held back as a separate validation set for this run. Instead, the model's ability to generalise was checked through live testing on the physical robot with objects the model had not seen during training, where it still detected and classified them correctly in real time. The trained weights were saved and loaded automatically each time the pipeline starts.

4.2 Software Setup

Getting the pipeline running at all required solving a few practical problems before any detection or grasping could be tested.

The Kinect interface library depends on a function that was removed from Python in version 3.8 and later. Running the pipeline on a current Python installation meant patching this at the start of the script so the missing function is redirected to its modern replacement. Without this fix, the program fails immediately when it starts.

Reading frames from the Kinect also needed some handling. Each frame arrives as a raw byte array, which has to be reshaped into a normal image array and converted into the colour format OpenCV expects before detection can run on it.

A small number of libraries carry most of the pipeline's work: Ultralytics YOLO for detection, OpenCV for image handling and the coordinate transform, XGBoost for the angle prediction step, NumPy for array and matrix operations, and Python's socket module for talking to the robot controller.

4.3 Robot Side Message Handling

Each message sent from the Python side follows a fixed format: object class, position in millimetres, and orientation angle. On the robot side, the RAPID program reads this string, splits it at the commas, and converts each part from text into a number it can use.

The orientation angle is converted into a rotation and applied to the target grasp point. The grasp sequence itself is the same for every object regardless of class: the robot moves to a safe height above the grasp point, descends to the surface, closes the gripper, waits briefly for the grip to take hold, lifts back up, moves the object aside, releases it, and returns to its home position. Only the position and the angle change between one grasp and the next, everything else about the sequence stays fixed.

5. Results and Discussion

Three separate grasping runs were carried out on the physical robot across different lab sessions. Each run kept most of the setup unchanged and altered only one part of the pipeline, so that any shift in the outcome could be traced back to the part that was changed. Reported first is the training of the detector, since nothing downstream can work without it, followed by the three runs in the order they actually happened.

5.1 YOLO Performance

Training and checking YOLO11 came before any robot trial, because a grasp attempt only makes sense once the object can be found in the image. All five figures in this section come from the same training session, referred to here as run five, and together they describe what the detector learned and where it stayed weak.

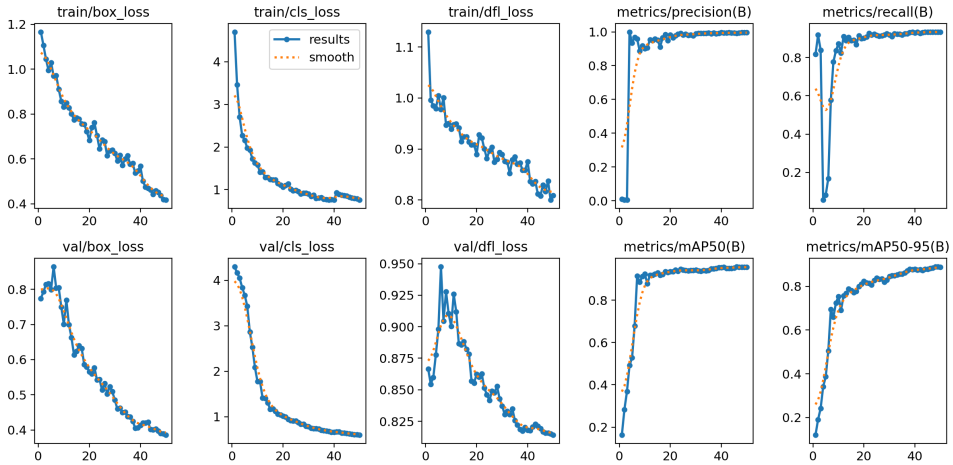

Figure 5.1: *training and validation curves over 50 epochs*

As can be seen in Figure 4.1, both the training losses in the top row and the matching validation losses in the bottom row fall together, and neither turns upward near the end. This means the model kept improving on images it had not seen rather than memorizing the training set. The accuracy measures on the right level off at around epoch 20 and stay flat, so the final values are stable rather than a lucky reading taken at the last epoch.

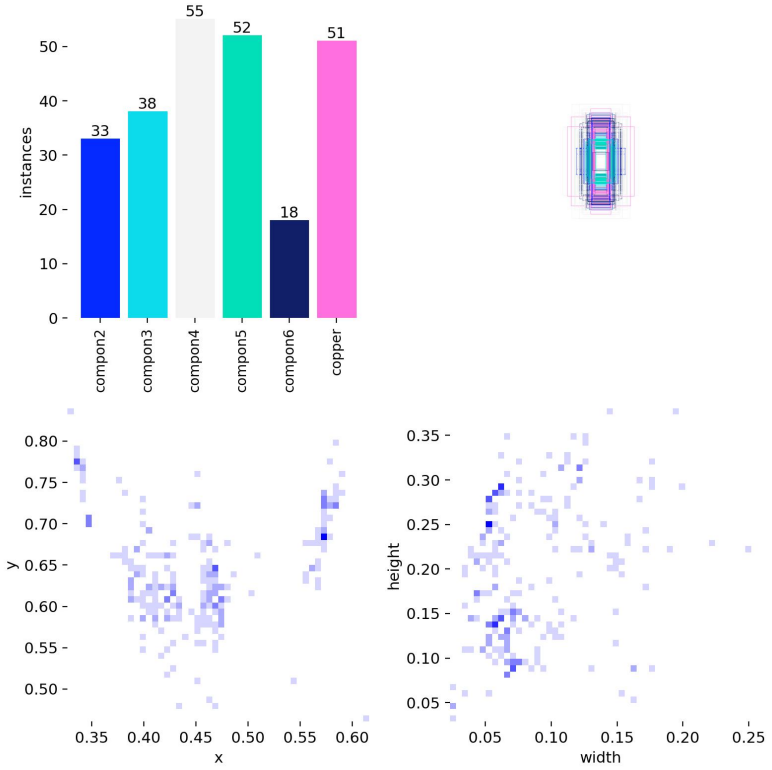

Figure 5.2: *a successful grasp under homography calibration*

As it is shown in Figure 4.2, the six classes are not evenly represented in the dataset. Compon4, compon5 and copper each have around 50 labelled instances, while compon6 has only 18. This gap matters for reading the next figure, becuase a class the model saw fewer times has had less chance to learn what it looks like.

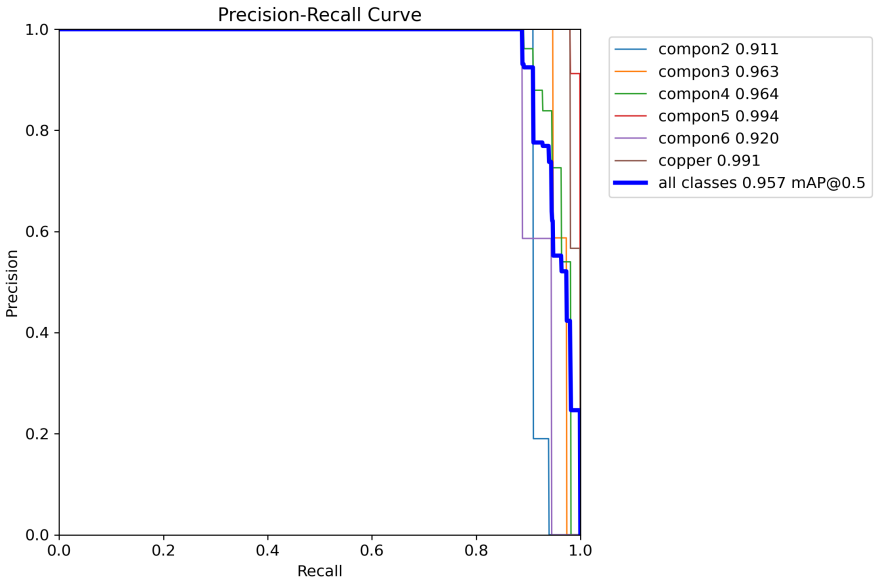

Figure 5.3: *accuracy per class at IoU 0.5.*

Figure 4.3 reveals that accuracy is far from uniform across the six classes. Compon5 and copper sit near the top at 0.994 and 0.991, while compon6 and compon2 trail behind at 0.920 and 0.911. Compon6 being both the class with the fewest examples in Figure 4.2 and the weakest class here points at the amount of training data as the cause, rather than anything unusual about the object itself.

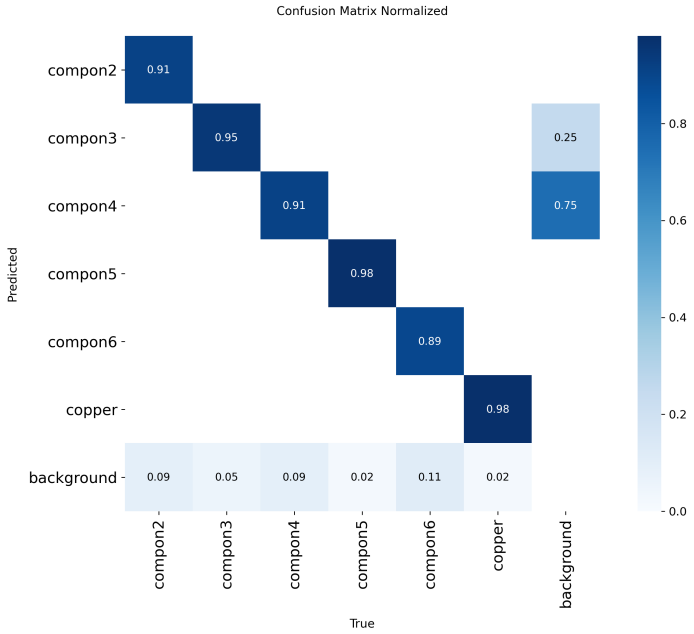

Figure 5.4: *confusion matrix on the validation set*

The diagonal in Figure 4.4 holds the correct identifications and everything off it is a mistake. No component class is ever taken for a different component class, since every error sits instead in the background row or column. The largest single error is background being read as compon3 in 25 percent of cases, which is the same kind of mistake as the false detections described later in section 4.3.

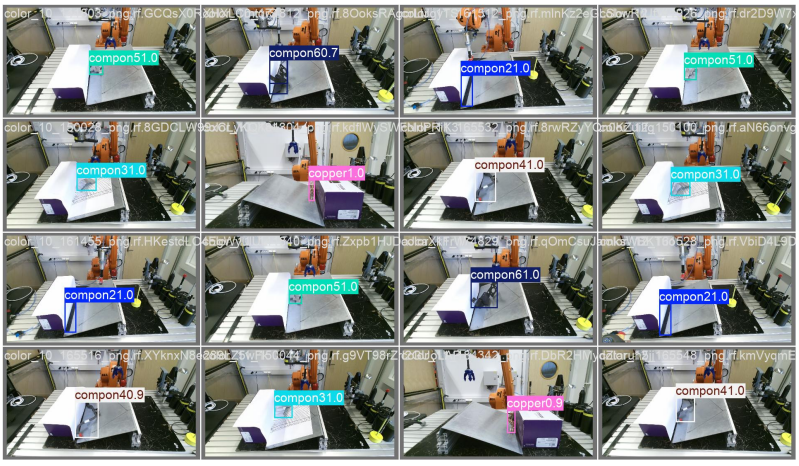

Figure 5.5: *predictions on held out validation images*

According to Figure 4.5, the confidence stays between 0.9 and 1.0 across sixteen images the model never saw during training. The boxes and numbers drawn on them are the model's own output rather than the correct answers, and they hold up across different objects, backgrounds and camera framings. This supports the accuracy figures above as real detection quality on unseen images and not just a favourable split of the data. Numerically the run ended at a precision of 0.9955, a recall of 0.9329, an mAP50 of 0.9564 and an mAP50 to 95 of 0.8879 by epoch 50. Strong as those figures look, the per class picture in Figures 4.2 and 4.3 is the more useful one for the live runs that follow, since it is the weaker classes that would be expected to cause trouble first.

5.1.1 Linear Calibration

Linear calibration was the method built into the earliest working version of the pipeline, and this was the first complete setup taken to the robot for a live grasping test. One in every three grasp attempts succeeded under this configuration, and a recorded trial with the copper coil shows both how the object was picked up and how it was put down.

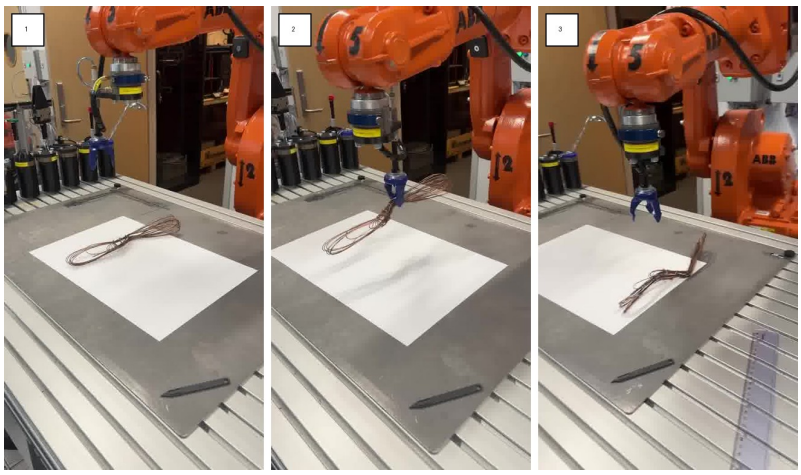

Figure 5.6: *the copper coil being picked up and released*

As can be seen in Figure 4.6, the copper coil starts lying flat on the A3 sheet in panel 1. Panel 2 shows the gripper closing on one end of the coil rather than at a suitable point nearer its middle, leaving the rest of it hanging down to one side. By panel 3 the coil has already been released and dropped in a bunched shape, the release coming quickly rather than in a controlled placement. Mounted at an angle rather than straight above the table, the Kinect camera introduces a problem known as foreshortening, where objects further from the lens appear smaller and their positions shift unevenly across the image. A straight line conversion from pixels to millimetres cannot follow that uneven shift, so its accuracy depends on how close the object sits to the one point it was calibrated against. Correcting this meant replacing the calibration method completly, which is what the second run tested.

5.1.2 Homography Calibration

Replacing the linear transform with a homography calculated from the four corners of the A3 sheet was the only change made for the second run. Detector, gripper and robot program all stayed as they were, so any difference in the outcome could reasonably be put down to the calibration method. Nine of ten grasp attempts succeeded, with the object deliberately moved to a different spot on the sheet between each try.

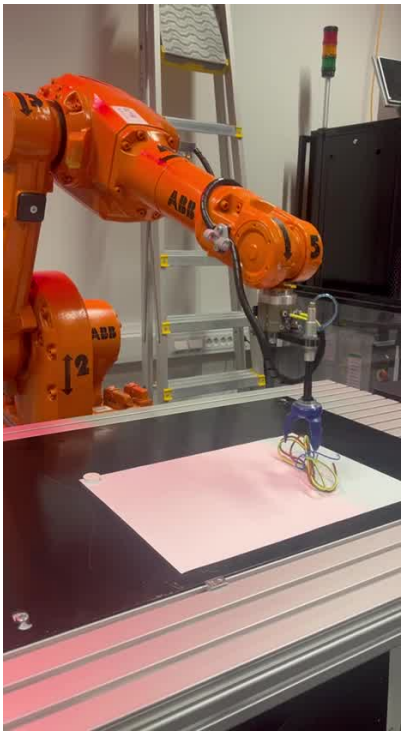

Figure 5.7: *a successful grasp under homography calibration*

Figure 4.7 reveals a cleaner grasp than the one in Figure 4.6: the gripper has closed around the object and lifted it clear of the sheet, holding it level rather than letting it hang from a single end. The same robot and the same gripper are at work, so what changed is the position they were sent to. The object here is a wire stand in rather than the trained copper coil, wich was not available in the lab that day.

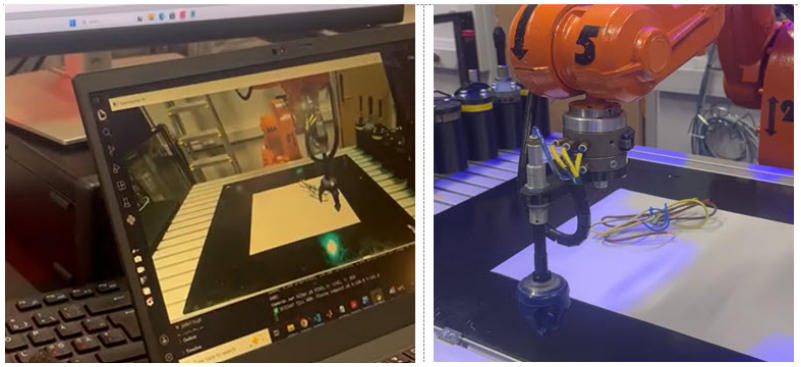

Figure 5.8: *YOLO's wrong detections led to incorrect grasping, caused by light reflection from the background.*

As can be seen in Figure 4.8, the bare metal surface of the table is visible around the edges of the A3 sheet and is lit up by the overhead lighting. This is the cause of the false detections seen during this run: more than once the detector read one of these bright patches as an object and sent its position to the robot exactly as it would for a real one. Working against this run were both of the conditions in the two figures above. Detection of the substituted wire object was slower than for the trained coil, and the lighting kept shifting throughout the session. That the success rate still rose to nine in ten under those conditions is the main reason the calibration change, rather than an easier test day, is credited with the improvement.

5.1.3 YOLO11 with Homography Calibration and XGBoost

Adding XGBoost on top of the working homography setup formed the third and final run. Where the previous two runs gave the robot only a position, this one also gave it an angle to approach from, predicted from the shape and placement of the detected box. Because no object had yet been placed at a clearly tilted angle for this test, the overall outcome barely moved compared with the second run, and grasps that already worked carried on working.

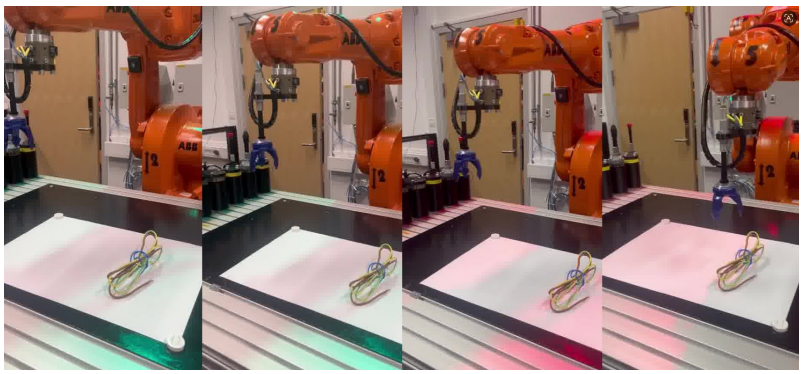

Figure 5.9: **the gripper turning as it lowers onto the object.*

Figure 4.9 the mechanical gripper rotates around its axe to grasp the object from better angle. The RAPID program applies it through pPick.rot := OrientZYX(nOrientation, 0, 0) before the grip begins. In the earlier runs it came

straight down at a fixed angle instead. As the object can be put with orientation on the table, especially object 4 and 6 which have more complexity and bends, applying relative orientation improved grasping compared to the case it was not considered.

5.2 Discussion

Across all three runs, the pattern was simple: get the position right first, then add a sense of direction on top of it, checking each step before moving to the next.

The biggest result is the jump between the first and second runs, from one success in every three attempts to nine in ten. A camera looking at the table from an angle distorts positions in a way a straight line calibration cannot represent, and Figure 4.6 shows what that looked like in practice: the gripper closed on the object but lost it as it shifted during the lift. Zhao et al. describe the same kind of problem, where an object moving during a grasp attempt causes it to be lost unless the robot accounts for that movement [15]. The homography fix used here takes a simpler route, it removes the positioning error that caused the shift rather than predicting the shift itself, and the result shows that was enough. The improvement held even with an unfamiliar object and changing light working against it, which points to the calibration change as the real cause.

This also matches a broader point about industrial pick and place: a fixed, one-time coordinate mapping is easily thrown off by small positioning errors, and closing the loop with visual feedback is one way industry has addressed this [14]. The homography calibration used here is still computed once per session rather than continuously, but it is based on the real shape of the working area instead of one reference point, which is why it held up across the whole sheet. A closed loop version of this is left for future work.

The accuracy numbers need one qualifier. Compon6, the class with the fewest labelled examples, was also the class the detector handled worst, a reminder that a class trained on few examples should be expected to detect less reliably until it gets more data.

The third run added a working orientation step on top of the calibrated pipeline. Figure 4.9 shows the predicted angle reaching the robot and turning the gripper to match how the object is lying, instead of always coming straight down. This is a real result on its own: one detection pass now gives both a grasp position and a grasp angle, without an extra sensor, which is what XGBoost was chosen to do in Section 3.8. How much this improves the success rate is the natural next experiment, picked up in the future work section. Gripper choice for these objects turned out to be less about performance and more about whether a gripper could handle the object at all. The suction gripper could not lift the copper coil, since there was no flat, sealable surface for it to grip. The electrical two finger gripper could not close around most of these shapes either, since its jaw gap did not suit them. In practice this meant that for a given object, usually only one of the three grippers was a realistic option, decided together with a fellow student's trial results and some testing of our own, rather than a performance difference measured across all three. The vision pipeline itself did not need to change for any of this: the same detection, calibration and message format worked no matter which gripper was mounted, since the choice of

gripper was made before the pipeline ran, not by it.

One more pattern worth separating out is the lighting problem from the second run. Light bouncing off the bare metal table, not off the object, triggered false detections, and the confusion matrix in Figure 4.4 confirms this: the model confuses background with an object, never one trained object with another. The A3 sheet fixed this wherever it covers the table, but the rest of the surface is still bare metal, so this was a fix for the tested area rather than for the detector itself.

Overall, these runs show a pipeline that improved sharply once the calibration problem was found and fixed, matching how much calibration and positioning error are already known to matter in industrial pick and place [14]. The detector handles its six classes well, though not equally well, depending on how much data each class had, and the pipeline now gives the robot both a position and an angle for every grasp. That progression carries forward into the conclusions in the next chapter.

6. Conclusion

This thesis set out to let an industrial robot pick irregularly shaped metal objects without a separate rule for each one. The system was built around a Kinect v2 camera, a YOLO11 detector, a coordinate transform, a predicted approach angle and an ABB IRB 1600 robot. Each part was added only once the part before it worked properly. Starting with the main aim, which was to give better grasp points so that fewer attempts fail, the answer is yes. With the first calibration method, only one attempt in three worked. The gripper kept closing on the edge of the object instead of a good point on it. Changing to a four point homography raised this to nine out of ten. That also answers the first question from the introduction. The reason is not that the old method needed better tuning. It assumes one pixel means the same distance everywhere in the image, and that is simply not true when the camera looks from an angle. The first sub aim is met as well, since the system runs all the way from camera image to a real grasp without rules written for each object. For most objects tested, only one gripper could actually do the job: suction could not lift the copper coil, and the two finger gripper could not close around the other shapes. Gripper choice was decided this way, using a fellow student's trial results together with some testing of our own, rather than measured as a performance difference. The pipeline itself did not need to change for any of this, since the same detection, calibration and message format worked with whichever gripper was mounted. The second question was whether the approach angle could come from what the detector already gives, and here the answer is only half complete. No extra sensor was needed, and the angle does reach the robot, becuase the gripper can be seen turning as it comes down. But it was never tested on objects placed at known angles, and the results barely changed. So the mechanism works, while the benefit is still unproven. This leads to how far these conclusions can be trusted. The calibration result is fairly solid. The difference is large, and it appeared even though an untrained object and changing light were both working against it. The rest is weaker. Few attempts were made, and each setup was recorded in only one session, so the numbers show a clear difference rather than a proper success rate.

6.1 Future Work and Research

Each direction below is written as a study that can actually be carried out and measured.

Test the angle prediction on objects placed at known angles. The third run showed the gripper turning to match the predicted angle, but not whether this improves grasp success. Testing several objects at known angles, with and without the angle step, would answer that.

Compare grippers on the same objects. In this project, gripper choice followed which tool could physically handle the object, not a measured comparison between grippers. A proper test, using more than one gripper on the same object under the

same pipeline, would show whether performance actually differs between them.

Test on a moving conveyor. All testing here used objects sitting still on a table, even though the project title mentions conveyor picking. Moving to a conveyor adds tracking and timing, and is the step closest to real production use.

Compare YOLO11 to other detectors. Training a second detector, like Faster R-CNN or RT-DETR, on the same dataset would show whether a better detection score actually leads to better grasp success, since the two are not guaranteed to match.

Add a camera built for reflective metal. The Kinect's depth data was too noisy to use on this material, and light reflecting off the bare table was the main cause of false detections. A laser based sensor made for reflective surfaces, such as a Zivid 3, would give reliable depth and could remove this problem directly instead of relying on a controlled background.

Bibliography

[1] Z. Xie, X. Liang, and R. Canale, "Learning-based robotic grasping: A review," *Frontiers in Robotics and AI*, vol. 10, p. 1038658, 2023.

[2] Y. Wu, T. Oudshoorn, and P. Rem, "Modelling and optimization of an innovative facility for automated sorting of aluminium scraps," *Waste Management*, vol. 189, pp. 103–113, 2024.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. [Online]. Available: https://doi.org/10.1109/CVPR.2016.91

[4] G. Jocher and J. Qiu, "Ultralytics YOLOv8," Computer software, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] Ultralytics, "YOLO11: Architecture overview," Ultralytics Documentation, 2024. [Online]. Available: https://docs.ultralytics.com

[6] ——, "YOLOv8 vs YOLO11 comparison," Ultralytics Documentation, 2025. [Online]. Available: https://docs.ultralytics.com/compare/yolov8-vs-yolo11

[7] R. Sapkota *et al.*, "Comparing YOLOv11 and YOLOv8 for instance segmentation of occluded and non-occluded immature green fruits in complex orchard environments," *arXiv preprint arXiv:2410.19869*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2410.19869

[8] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021. [Online]. Available: https://doi.org/10.1109/JPROC.2020.3004555

[9] Ultralytics, "What is transfer learning?" Ultralytics Glossary, 2026. [Online]. Available: https://www.ultralytics.com/glossary/transfer-learning

[10] E. Pandilova, M. Petrov, V. Spasev, I. Dimitrovski, and I. Kitanovski, "Transfer learning with YOLO for object detection in remote sensing," in *ICT Innovations 2024. Communications in Computer and Information Science*. Springer, 2025, vol. 2436. [Online]. Available: https://doi.org/10.1007/978-3-031-86162-8_9

[11] Loccioni Group, "Building robust industrial applicable object detection models using transfer learning and single-pass deep learning architectures," *arXiv preprint arXiv:2007.04666*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2007.04666

[12] MathWorks, "What is robot hand-eye calibration?" MATLAB & Simulink Documentation, 2024. [Online]. Available: https://www.mathworks.com/help/vision/ug/what-is-robot-hand-eye-calibration.html

[13] S. K. Paul, M. T. Chowdhury, M. Nicolescu, M. Nicolescu, and D. Feil-Seifer, "Object detection and pose estimation from RGB and depth data for real-time, adaptive robotic grasping," *arXiv preprint arXiv:2101.07347*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2101.07347

[14] K. Benali, "Optimizing grasping precision for industrial pick-and-place tasks through a novel visual servoing approach," *arXiv preprint arXiv:2501.14557*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.14557

[15] J. Zhao, J. Liang, and O. Kroemer, "Towards precise robotic grasping by probabilistic post-grasp displacement estimation," *arXiv preprint arXiv:1909.02129*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1909.02129

[16] S. Pherwani, "Comparative evaluation of end-effectors for autonomous picking of irregularly shaped metal scrap: An experimental study for the super sorter initiative at mission 0 house," Master's Thesis, University West, Trollhättan, Sweden, Jun. 2026.

[17] O. Benderius and C. Sprunck, "Object detection using YOLO oriented bounding box for robot grasping applications," *Journal of Industrial Engineering and Management Systems*, 2026. [Online]. Available: https://doi.org/10.1016/j.jiems.2026.100161

[18] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019. [Online]. Available: https://doi.org/10.1126/science.aat8414

[19] J. Mahler *et al.*, "Mechanical search on shelves using a novel bluction tool," *arXiv preprint arXiv:2201.08968*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2201.08968

[20] D. Morrison, A. W. Tow *et al.*, "Cartman: The low-cost Cartesian manipulator that won the Amazon Robotics Challenge," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018. [Online]. Available: https://doi.org/10.1109/ICRA.2018.8463191

[21] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[22] IBM, "What is XGBoost?" IBM Think, 2025. [Online]. Available: https://www.ibm.com/think/topics/xgboost

[23] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" *Advances in Neural Information Processing Systems*, vol. 35, pp. 507–520, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2207.08815

[24] F. A. Coutelieris and A. Kanavouras, "Experimentation methodology for engineers," SpringerBriefs in Applied Sciences and Technology, Springer International Publishing, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-72191-0

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," arXiv preprint arXiv:1405.0312, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1405.0312

[26] Z. He *et al.*, "A comprehensive review of research on surface defect detection of PCBs based on machine vision," *Results in Engineering*, vol. 21, 2025. [Online]. Available: https://doi.org/10.1016/j.rineng.2025.106437