

Rheinisch-Westfälische Technische Hochschule Aachen
Knowledge-Based Systems Group
Prof. Gerhard Lakemeyer, Ph. D.

Master's Thesis

Whole-Body Manipulation on Mobile Robots Using Parallel Position-Based Visual Servoing

Matteo Tschesche

04.05.2022

Supervisors: Prof. Gerhard Lakemeyer, Ph. D.
Prof. Dr. Sebastian Trimpe
Advisor: Till Hofmann, MSc

Contents

1	Introduction	4
2	Acknowledgements	5
3	Background	6
3.1	Object Detection	6
3.2	Visual Servoing	9
3.3	RoboCup Logistics League	12
4	Related Work	14
4.1	Object Manipulation in the RCLL	14
4.2	Other RoboCup Leagues	15
4.3	Outside of RoboCup	16
5	Approach	18
5.1	Overview	18
5.2	Object Detection	20
5.2.1	Requirements	20
5.2.2	3D Projection	21
5.2.3	YOLO	24
5.2.4	Alternatives	25
5.3	Error Calculation	26
5.4	Controlling	28
5.4.1	Main Body Control	28
5.4.2	Gripper Control	29
5.5	Simulation	31
6	Evaluation	32
6.1	Object Tracking	32
6.1.1	Camera Positioning	33
6.1.2	Object Tracking Speed	36
6.1.3	Evaluation Metrics	36
6.1.4	Object Detection Accuracy	39
6.1.5	Sensing	40
6.2	Evaluation of VS and ICP	41
6.2.1	Positioning	44
6.2.2	Alignment	45
6.2.3	Routine	47
6.2.4	Total Time	49
6.3	Grasping Challenge	52
6.4	Robustness against Environment	54
6.4.1	MPS Placement	54

6.4.2	Lighting	55
6.5	Future Work	60
6.5.1	Camera	60
6.5.2	Laser Data	60
6.5.3	Neural Network	61
7	Conclusion	62

1 Introduction

Object manipulation is a common and well studied robotics problem in which a robot needs to interact with an object, e.g., by picking it up, transporting it to a target location, and putting it down. Aligning a mobile robot to the object and grasping it is challenging and requires precise control of body and gripper while simultaneously navigating around the environment and tracking the target object's position. However, due to the wide variety of robots, grippers, environments, and objects, there is no universal manipulation approach to solve this problem for every scenario. This thesis presents a manipulation framework for robots with precise grippers, imprecise main bodies, and a monocular camera like the Carologistics' robot from the RoboCup Logistics League (RCLL).

This is a common setup since robot wheels often slip, especially if they are omnidirectional, and therefore do not allow precise movements. In contrast, a gripper is precise, but has limited reach. Furthermore, monocular cameras are the most widely used sensors.

Visual servoing is a common manipulation approach in which camera images are used to control robots. Features are extracted from the image and compared to the target features. The difference between them defines an error. The robot's velocity is determined based on this error and minimizes it until the target robot pose is reached.

We propose a whole-body motion framework for manipulating objects, which uses a YOLOv4-tiny network to detect the target object and computes the object position in 3D space. This position is used to calculate the target poses for the gripper, and main body. Both target poses are then reached simultaneously by solving 2 position-based visual servoing tasks. After reaching these poses, a gripper routine executes the manipulation, e.g. grasping the target object.

The approach is tested in the RCLL environment using Carologistics' Robotino. The RCLL is a worldwide competition in which teams of autonomous robots build and deliver products in an industry 4.0 environment. To do this, the robots need to pick workpieces from conveyor belts, or shelves and place them on MPS slides, or conveyor belts. We evaluate our proposed method on these tasks and compare it against Carologistics' current manipulation approach.

The 3 chapter describes object detection, visual servoing, and RoboCup Logistics League in more detail. Afterwards, we summarize related work about object detection and manipulation. In the subsequent chapter, we give an overview about the proposed approach and cover its object detection, error calculations, and robot control. Chapter 6 presents the evaluation of our object detector, covers the approach's robustness and describes possible future work. The final chapter summarises this thesis.

2 Acknowledgements

I would like to thank everybody who helped me during my master's thesis. Special thanks to Till Hofmann, who supported me very well even while working on his dissertation, and Nicolas Limpert, who patiently helped me to work with the robots. I also have to thank the Computer Vision chair which gave me good advice on training neural networks and saved me a lot of time.

Lastly, I want to thank my family and friends for their support.

3 Background

This chapter provides background knowledge for visual servoing, which consists of object localization, tracking, and robot controlling. Object detection, which is used for object localization and tracking in 3D space in this thesis, is discussed first. Afterwards, different visual servoing approaches are explained and, lastly, a short description of the RoboCup Logistics League, which is the test environment for the proposed approach, is given.

3.1 Object Detection

Computer vision is an active field of computer science, which is developing algorithms to analyze images and solves many kinds of vision problems. Object detection is a common computer vision task in which objects are located in a given image, for example by bounding boxes, and classified by a finite number of classes. The task description can vary, e.g., by detecting multiple classes, multiple objects per image, or multiple objects of the same class in one image [45, 18].

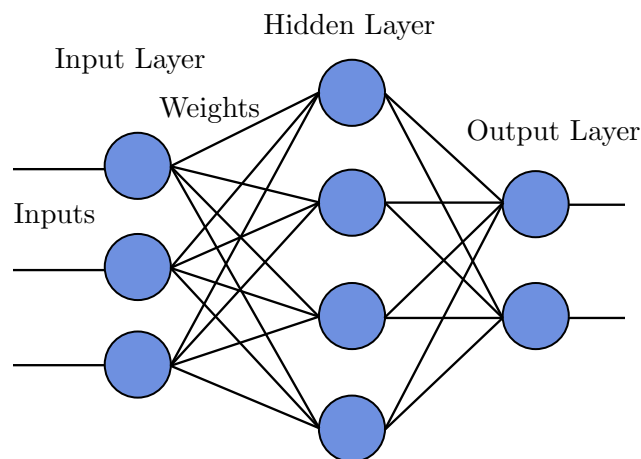


Figure 1: Feed-forward neural network (adapted from [5]).

It is common for computer vision tasks to use machine learning approaches like neural networks to solve them. Figure 1 shows a feed-forward neural network, which consists of an input layer, a hidden layer, and an output layer. Each layer consists of multiple neurons, which are connected to every neuron in the previous and following layer. Each connection has a weight attached to it, which is learned during training. When passing through the network, the neurons in the input layer are set to the input of the network. Afterwards, the values of the connected neurons in the next layer, the hidden layer, are computed depending on the values of the neurons in the previous layer and their

respective weights as well as the hidden layer’s activation function. This is repeated until the values of the output neurons are known.

One way of training a neural network is to use supervised learning in which a data set consisting of inputs and their respective correct outputs, called labels, is split into a training and test set. The training set is used to run the network on each input and compare the network’s output with the label using a loss function to determine its error, also called loss. Furthermore, the gradients of the weights and biases can be computed using the loss by passing gradients backwards through the network. This process is called backpropagation. These gradients are then used for a process called gradient descent in which parameters for the weights and biases are determined to reduce the error. Backpropagation and gradient descent are repeated until the loss converges. The test set is then used to evaluate the networks performance on unseen data. Feed-forward neural networks consist only of fully-connected layers, but there are many different types of neural networks with different layers, and learning techniques [18].

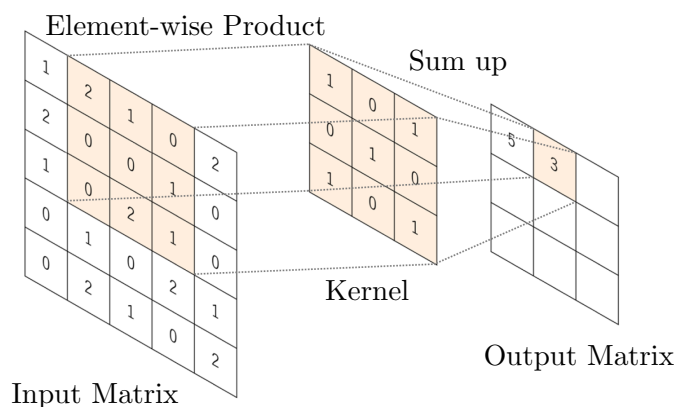


Figure 2: Convolution layer (adapted from [42]).

Conventional machine learning approaches required hand-engineered feature extractors made by domain experts to convert the raw image data into usable representations, but modern deep learning approaches also learn their feature representations by themselves. Convolutional Neural Networks (CNNs) are types of neural networks which are well suited for computer vision tasks and learn low- to high-level feature representations on their own while training. They consist of convolutional layers, which utilize the fact that image content is independent of its pixel position, which allows parameters to be shared [42, 18].

Figure 2 shows a convolution, which is done by sliding a filter, or kernel, over the image and computing a value for each pixel by summarizing over all input values within the kernel with respect to the kernel’s weights similarly to fully-connected layers. A convolutional layer can have multiple filters, whose results are stacked together. These stacks form the depth dimension of the resulting feature space. The kernel’s weights are used

for the whole image and, therefore, the amount of parameters used in a convolutional layer is lower compared to fully-connected ones. This allows to build deep networks that are easier to train. These networks usually contain pooling layers between convolution layers to reduce the feature space, for example, by only taking the maximum value of a 2×2 grid [34, 42].

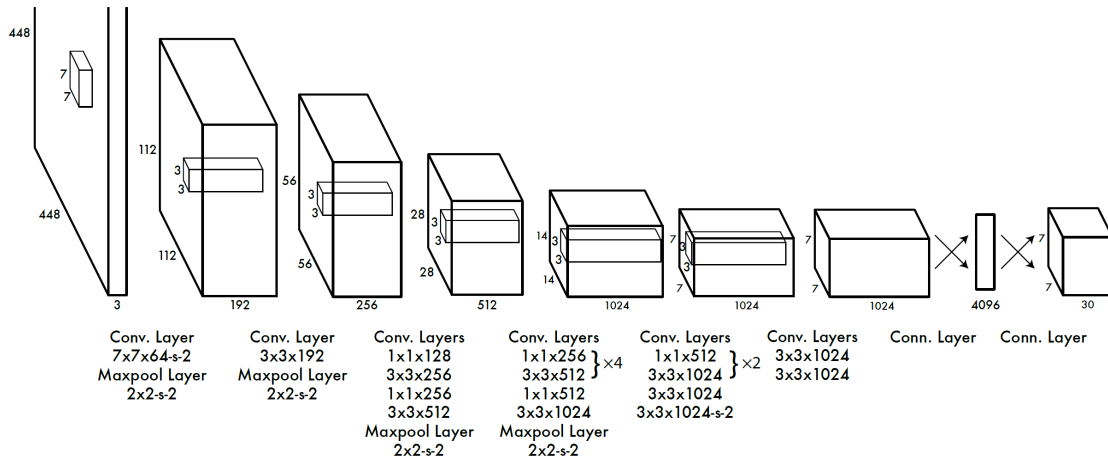


Figure 3: Architecture of YOLOv1 [34].

You Only Look Once (YOLO) [34] is a neural network for object detection, which mainly consists of convolution and pooling layers. By now, there are 4 official YOLO versions and with each, the amount of layers increased [34, 32, 33, 4]. The structure of the first official YOLO version is shown in Figure 3. It iterates between convolutional and maxpool layers and ends with two fully-connected layers. Later versions of YOLO also mainly use convolutional, pooling, and fully-connected layers, but also include some other layers like normalization layers, such as Batch Normalization [15], other types of connections like skip-connections, and regularization methods, e.g. DropOut [37, 4].

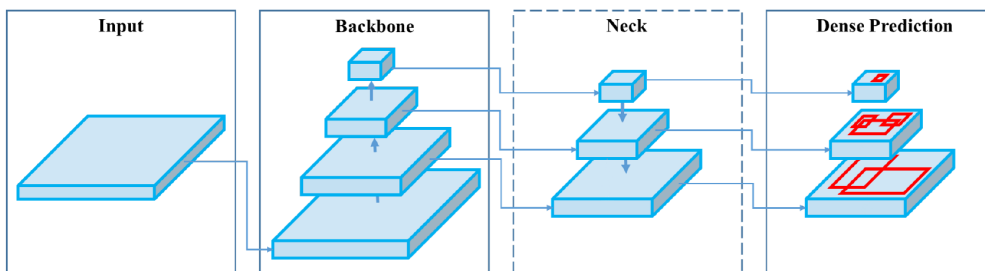


Figure 4: Structure of One-Stage Detectors (adapted from [4]).

The latest YOLO version, YOLOv4, is a state-of-the-art multi-object detector that can detect many different object classes in images. It has a low runtime in comparison to other state-of-the-art multi-object detectors and outputs bounding boxes with object classes. Bounding boxes are allowed to overlap with each other, and boxes of the same

class can appear multiple times in a single image [4].

The layers of YOLOv4 are grouped into a backbone, neck, and head and are designed to fulfill different tasks as shown in Figure 4. The input image is first processed by the backbone in which filters reduce the feature dimensions while increasing the feature depth and abstraction level with each stage. In YOLOv4, this is done with the CSPResNeXt50 [39], which contains 29 3×3 -convolution layers. The neck is used to collect feature maps from different layers and increase the receptive field of the network. As neck, YOLOv4 uses a Path Aggregation Network (PAN) [22] to aggregate parameters from different backbone levels for different detector levels and Spatial Pyramid Pooling (SPP) [10] to increase its receptive field. The last part, the head, of YOLOv4 is YOLOv3, which is used to predict object classes as well as their bounding boxes [4].

Most object detection algorithms are capable of detecting any kind of object, but objects with distinct forms or textures are usually much easier to detect and allow more sophisticated and faster algorithms to detect them [32]. Barcodes, for example, show such properties and are very useful in many applications in normal day life. Overall, barcode detection is a well studied problem with many fast algorithms as well as public data sets [44, 16].

This thesis utilizes a YOLOv4 variant to extract features from camera images as input for visual servoing, which is covered in the next section.

3.2 Visual Servoing

Autonomous robots require knowledge about their environment before moving. There are many different sensors for this purpose, such as RGB-D cameras, stereo cameras, laser sensors, GPS, and many more. The task of controlling a robot based on image features is called visual servoing (VS) [14, 6, 7, 40].

VS is used to move mobile robots or stationary arms. Such an approach starts with visual information from which predefined target features are extracted. To compare the target and current situation, which is either a pose or a set of image features, an error is defined, e.g., as the difference between the target pose and the current pose. With this information, the VS algorithm controls the robot's pose to reduce this error. VS approaches can be categorized by their type of control loop, whether they are open- or closed-loop, and their error calculations, which are either position- or image-based [14, 40].

Figure 5 shows the structure of an open-loop approach in which the current situation is given by the vision system. This step is also called "look". The current situation is then compared to the desired situation by the controller to estimate the error. A move command is sent to the robot, which should move the robot towards the desired

situation to reduce this error. For controlling the robot, only odometry responses are used without any further advise of the vision system, which is the reason why it is called an open control loop. This step is called "move" and explains why open-loop VS is also referred to as "look and move". After moving, the vision system extracts features from a current image to determine the current situation. If the robot is close enough to the desired situation, the algorithm terminates. If this is not the case, another move command is sent, which repeats until it is close enough [14, 40].

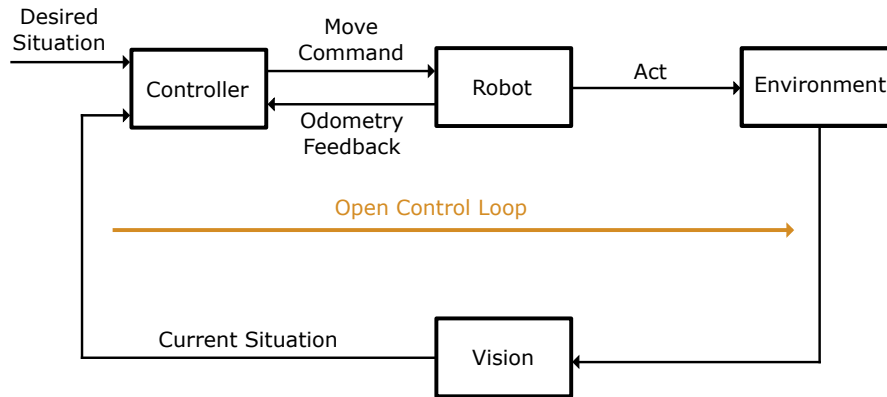


Figure 5: Structure of open-loop VS.

Figure 6 shows a closed-loop architecture, which showcases the difference between both approaches. The control loop now includes the vision system, which is directly used for the vision controller and does not rely on odometry feedback. It frequently updates the current situation and allows the vision controller to update the error, which usually results in end-effector velocities as move commands. "Look" and "move" are performed in parallel in contrast to open-loop approaches, which perform them serially. This allows to dynamically adjust to movement errors of the robot and changes in the environment as well as the target. In a static environment, using a perfectly calibrated camera and a perfectly precise robot with noise-free sensors, both approaches are equivalent, but these assumptions never hold in practice. Therefore, closed-loop architectures are more precise and dynamic with the disadvantage of higher computational costs because of their higher usage of their vision systems [14, 40].

The major difference between VS approaches in error calculations results in position-based VS (PBVS) and image-based VS (IBVS). PBVS defines the error in 6D space and, therefore, extracts the target image features into coordinate frames and compares them to the current robot pose. To estimate this pose, depth perception is needed, such as triangulation in case of stereo cameras, the depth estimate of RGB-D cameras, or a depth estimate through known feature sizes. Additionally, the end-effector position needs to be known, either through end-effector tracking or hand-eye calibration. An advantage of PBVS against IBVS is its reliance on real world distances instead of computations in image space [14, 6].

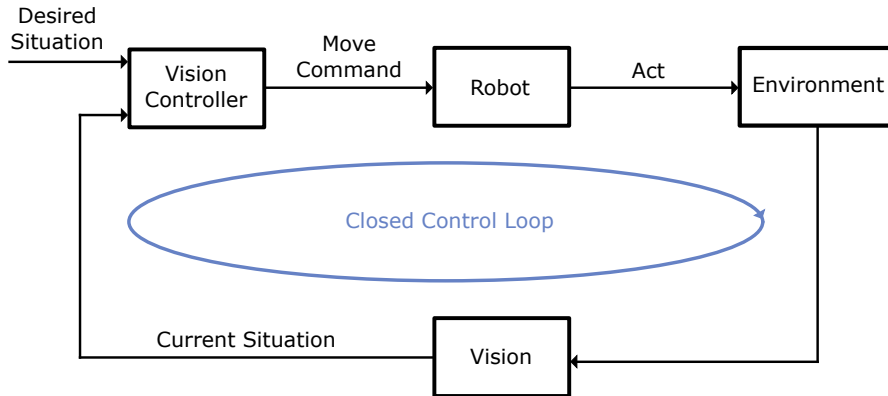


Figure 6: Structure of closed-loop VS.

IBVS defines the error directly in image space. The desired and current situation in the VS loop consist of the appearance of tracked features in image space like primitive shapes or textures, which are changing in image space as the robot moves. The advantage of this method is that there is no interpretation from image space to 3D space, and depending on the robot-camera configuration it can be less sensitive to camera calibration as well as hand-eye coordination. The move commands containing end-effector velocities are computed from image Jacobians of known features. The image or feature Jacobian, also known as interaction matrix, is used to derive changes in image space based on velocity and angular velocity of the robot as well as intrinsic parameters of the camera [14, 6].

There are hybrid approaches between IBVS and PBVS too. For example 2 1/2-D VS defines features partially in 2D and partially in 3D space to decouple rotational from translational motions [7].

The installation of the camera in comparison to the robot results in different robot-camera configurations. If the camera is mounted on a moving robot, it is called eye-in-hand and if it is not, so if the camera is stationary, the configuration is called eye-to-hand. In eye-to-hand approaches, there is a differentiation between endpoint closed-loop, in which case the end-effector is always visible in the camera, and endpoint open-loop, where it is not. The accuracy of these endpoint open-loop approaches depend on the hand-eye calibration of the robot and camera. In contrast, the endpoint closed-loop approach is more robust to hand-eye calibration, but requires end-effector tracking. IBVS requires a hand-in-eye approach or a fixed camera such that manipulator movements cause changes in the image to reliably compute the error, which usually requires an endpoint closed-loop system [14, 7].

There is a wide variety in visual features to extract, track, and detect in images, such as simple shapes like lines, circles, cylinders, special textures, or objects used for detectors. Once a feature is detected, it can be tracked using the same method or simpler ones, using the expensive first detection only for initialisation. To assign the correct response

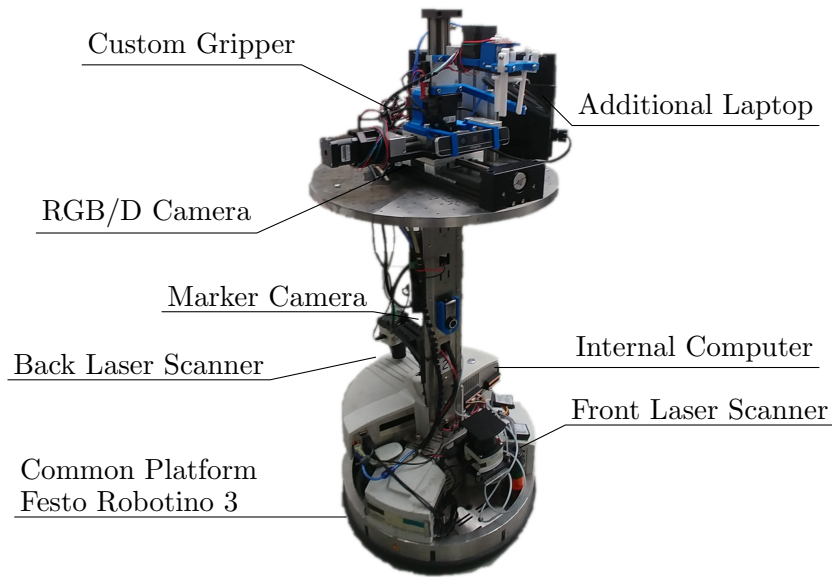


Figure 7: Carologistics' Robotino (adapted from [12]).

to the tracked feature, methods such as thresholding around the last or expected position as well as nearest neighbor fitting can be used [14, 6, 7].

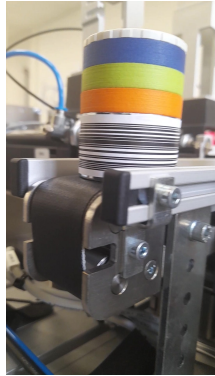
After the feature positions are extracted, the controller computes the next move command in form of a target pose or an end-effector velocity, which is then sent to a low-level controller. Additional constraints, such as keeping sight of the target, or keeping a certain distance to objects, can be included, which are considered when choosing an end-effector velocity and simultaneously reduces the error. This can also be done by the low-level controller if a target position is computed through VS instead of an end-effector velocity [14, 6].

3.3 RoboCup Logistics League

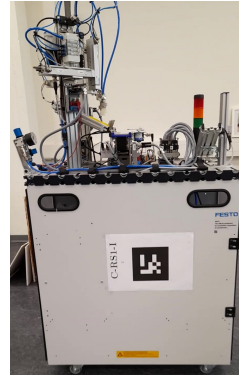
The RoboCup Logistics League (RCLL) [28] is a worldwide competition in which teams of autonomous robots transport workpieces in an industry 4.0 environment. Each match between 2 teams with each 3 autonomous robots is decided by the amount of points each team has after 17 minutes. Points are earned for delivered workpiece orders.

Figure 7 shows a Robotino of the Carologistics team consisting of a 3 DoF omnidirectional Robotino body including a laser range finder, an RGB-D camera (Intel Realsense SR300), a 3-finger gripper, and a laptop for additional computing power. The robot's main body is imprecise with a minimum translational tolerance of 2 cm, but its attached gripper is very precise [13].

An order consists of a workpiece description and a delivery time window. A workpiece description consists of a base, between 0 and 3 rings, which are stacked on top of each other, and a cap, where each part has a determined color. Figure 8a shows a finished order with 3 rings. When interacting with the machines, the so-called Modular Production System stations (MPS), a robot either sends a signal, takes a workpiece from a conveyor belt or a shelf, or puts it on a conveyor belt or an MPS slide. Each team has 7 MPS of 5 different types with different purposes like mounting a ring or delivering an order. Figure 8b shows a type of MPS, a cap station, with a conveyor belt and a slide. The main challenges of the RCLL are planning the robot's actions and MPS usage, robot navigation, and workpiece manipulation, the latter being the topic of this thesis [13, 38].



(a) Image of a finished order with a silver base, an orange first ring, a green second ring, a blue third ring, and a grey cap.



(b) Image of a cap station front with an empty conveyor belt and slide.

Figure 8: RCLL environment.

Gazebo allows to simulate these matches by simulating each robot with artificial noisy laser data and the same functionality as real robots. The simulated MPS stations fulfill the same tasks and take an average recorded time. The simulation allows to test new approaches in a safe environment [46].

4 Related Work

In this chapter, we will cover related work about VS, object detection, and whole body movement. Approaches used for the RCLL are covered first. Afterwards, approaches from other RoboCup leagues are explained and, lastly, related approaches outside of RoboCup are mentioned as well.

4.1 Object Manipulation in the RCLL

Hofmann et al. [13] explain Carologistics' current workpiece manipulation approach for the RCLL, in which the workpiece itself is never detected but the conveyor belt and MPS slide are. The robot aligns with the conveyor belt or MPS slide such that the gripper pick up workpieces from the conveyor belt or shelf, or put workpieces on the conveyor belt or MPS slide. A multi-modal 3-stage pipeline is used in which, first, the LiDAR sensor is used to determine the pose of the MPS box with a RANSAC approach that fits a line parametrization while matching the MPS boxes' width. The second stage consist of a RANSAC-based plane fitting using the previous stage as an initial guess. In the final stage, an Iterative Closed Point (ICP) based model fitting approach is used on the point cloud provided by the RGB-D camera with the previous stage's approximation as initial guess as shown in Figure 9. After this precise alignment, the robot follows a routine for picking up the workpiece or putting it down without further visual information. We use this approach as a comparison for evaluating the proposed framework.

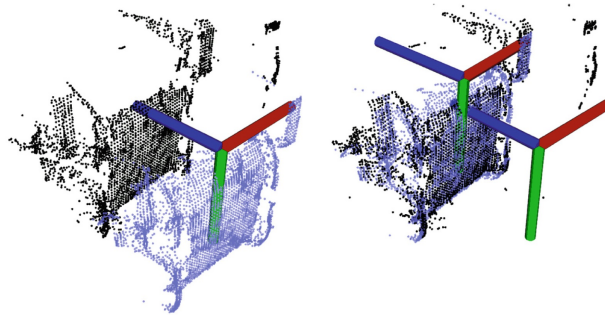


Figure 9: On the left, the saved conveyor belt model (blue) is aligned near the captured point cloud (black). The right diagram shows them precisely aligned as result of ICP [13].

The ICP takes several seconds to compute and requires good lighting conditions as well as a good point cloud model of the MPS's conveyor and MPS slide, which does not adapt to changes in practice, like a different wiring. Once the ICP terminates, the robot has a static understanding of the environment and cannot adapt to changes if, for example, a robot pushes the machine.

Ulz et al. [38] present the manipulation approach for the GRIPS team at RCLL 2018. A closed-loop control approach is used in which the robot aligns in front of the MPS by reducing 3 errors, one for the offset in x direction, one for the y direction, and one for the angular offset from the MPS middle point. The relative MPS position is approximated by clustering the main body’s laser data, followed by least squares line fitting, and is used to calculate the errors frequently. The errors are then minimized using a sliding mode controller, which approximates the sign function with a saturation function.

4.2 Other RoboCup Leagues

Lee et al. [19] were the first to implement a VS approach in the RoboCup small sized league, also called F-180 league, in 2003. The challenge consists of a 6 versus 6 soccer match played in a highly dynamic environment with small robots and an orange golf ball on the ground. IBVS and PBVS were similarly successfully implemented using the known geometry of the ball and the calibrated camera to approximate the distance between ball and camera.

Object manipulation is also an important aspect of the RoboCup@Home League, where a robot consisting of a 3-DoF mobile body and a 6-DoF arm interact with everyday items in a home environment. Mitrevsky et al. [25] use demonstration learning for motion primitives to grasp different objects such as cups. The motion primitives are represented by a second-order differential equation representing a trajectory in Cartesian space. Its parameters are learned by observing arm movements controlled by a human operator. The trajectory is robot-independent and achieved by solving inverse kinematics. The robot first detects the object using a Single Shot MultiBox Detector [23], determines its pose, chooses a grasping position, aligns itself with the object, and starts grasping it. Figure 10 shows a successful attempt using this approach. This framework is a whole-body motion framework that moves the arm whenever possible and is only moving the robot body if the manipulator Jacobian is close to a singularity.



Figure 10: The robot successfully grasps a cup using whole body motion [25].

Loncomilla et al. [24] introduce an object detection approach for multiple everyday items used in the @home league, which relies on YOLOv3 to generate proposals and continues

with a Maximal Activation of Convolutions (MAC) approach to recognize the objects. This proves the capability of YOLO to be used as a reliable object detector in such a complex competition environment with occlusions and illumination changes.

Another manipulation approach is shown by Padalkar et al. [29] for the RoboCup@Work League in which an omni-directional robot body with a 5-DoF arm is used to pick up objects on a rotating plane. SqueezeDet is used to obtain 2D bounding boxes of the objects. An RGB-D camera provides the 3D points in these bounding boxes, which are then used to create 3D bounding boxes around the objects. The centroids of the 3D boxes are tracked using the closest distance between observations and are used to determine the object trajectory. A pre-grasp planner is used to determine an end-effector position to grasp the object and the arm is moved there using a weighted damped least square inverse kinematic velocity solver. The timing of the grasp is based on the determined object trajectory.

4.3 Outside of RoboCup

Agravante et al. [1] simultaneously uses IBVS to move a humanoid robot body to keep vision at the target box and uses PBVS to move its arm to grasp the box. Additional conditions, such as field of view, and occlusion avoidance are integrated in this VS framework represented as inequalities to an acceleration-resolved, quadratic optimization problem. To keep track of the robots hand, blob detection and pose estimation are used. Furthermore, a model based tracker is used to localize the target box.

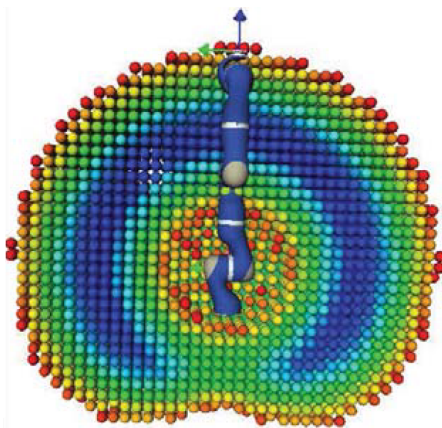


Figure 11: Reachable workspace of 7-DoF arm [43].

Zacharias et al. [43] explain the concept of reachable workspace, which defines the 3D space reachable by a robot arm. Complicated arms have reachable and unreachable spaces. The reachability map provides a discretized space around the arm indicating the reachability factor for each sphere, which is computed by solving inverse kinematics for multiple points in each sphere. Figure 11 shows such a discretized reachable workspace

of a 7-DoF arm using spheres where red represents a low, green a middle, and blue a high reachability, which is given by the percentage of reachable points within the sphere. Since this map is independent of the robot's main body, it can be computed once and used in any situation. With this knowledge, the robot can check if the target object is in its reachable workspace and if not, it moves towards the target until it is.

The reachable workspace of this thesis' gripper is clearly defined as a box. However, the proposed framework may be extended to any precise gripper with a known reachable workspace, which can be computed with the approach of Zacharias et al [43].

Liu et al. [21] present a garbage detection and grasping approach. The YOLACT framework is used to detect garbage and provides a pixel map of the object, which is used to determine a point cloud of the object using RGB-D data over the object's pixel map. A grasping pose is computed using Grasping Pose Detection on the voxelized point cloud. To reach this pose, a trajectory planner unit is used, which computes intermediate waypoints using a fifth-order polynomial law, which are then targeted by a whole-body Cartesian impedance controller.

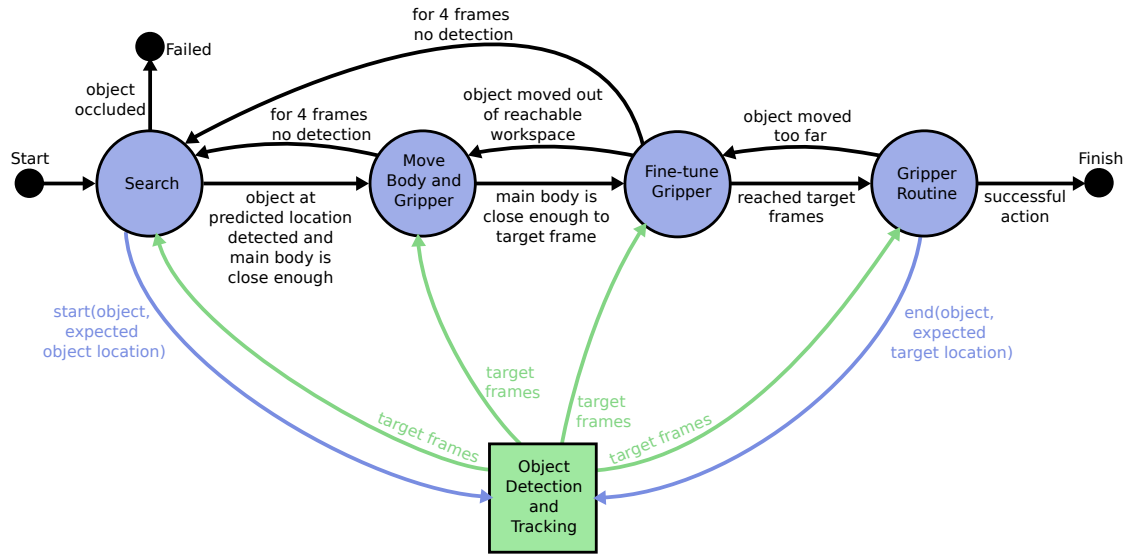


Figure 12: State machine for the proposed approach.

5 Approach

This section presents a new whole-body object manipulation framework. This approach is implemented on a robot of the Carologistics team and integrated in the Fawkes Robot Software Framework [27]. YOLOv4-tiny is trained to detect and track the target objects reliably. The goal of the approach is to navigate the robot to the target object while moving its main body and gripper simultaneously such that the object can be picked up or put down quickly. The task of reaching the target poses is defined as 2 simultaneous PBVS tasks.

The robot either needs to pick up a workpiece from the conveyor belt or a shelf, or put down the grasped workpiece on a conveyor belt or an MPS slide of an MPS.

5.1 Overview

In this thesis, we present a whole-body movement framework for robots, with an imprecise 3-DoF main body and a precise gripper for any manipulation task.

Figure 12 shows the state machine for this approach. When starting a manipulation, the target object’s class and an expected location of it is known. This information is used by the object tracker to ignore all other object classes. The object tracker estimates the distance between camera and object to compute a relative 3D object position. The target poses for the gripper and main body are computed by adding predefined offsets to this position.

We use PBVS for the robot’s target body and gripper poses to solve both VS tasks simultaneously. Throughout the procedure, the target poses are updated frequently by continuously detecting the target object and computing the target poses. Therefore, it is a closed-loop VS approach. To apply VS, the target object needs to always be in the camera view. An expected object position is known through map data and the robot needs to drive there such that the target object can be viewed. This is done in the first state, the ”Search” state, by driving to the expected position and keeping track of incoming target poses from the object detector.

If the target is detected reliably and the distance between robot and target is below a threshold, the ”Move Body and Gripper” state is reached. In this state, the main body is continuously driving towards its target pose while the gripper assumes that the body is already at its target pose and is reaching the gripper’s relative target pose under this assumption.

If the main body has reached its target pose within a given threshold, the state ”Fine-tune Gripper” is selected in which the gripper does not assume the body’s pose to be exactly the target pose anymore. It uses the current body pose and moves the gripper to its actual target pose.

If this pose is reached, the robot is in the final state, ”Gripper Routine”, in which the gripper performs a predefined sequence of actions depending on the manipulation task and after the routine finished, the object detector is stopped. The robot reached its desired goal and the manipulation action is performed successfully. If the routine failed, we assume that the object moved too far and transition back to the ”Fine-tune Gripper” state.

The transitions between states do not require additional information and since the target poses are updated frequently even while moving, the robot does not need to stop until it reached its target poses. The robot is continuously driving even while transitioning between internal navigation and VS, which is an advantage compared to previous approaches, which would always require the robot to stop for several seconds for sensor updates.

The approach is based on PBVS instead of IBVS because using IBVS for the gripper control would require the gripper to always be in the camera view while grasping, which is very limiting and unnecessary if the camera and gripper are calibrated well. Additionally, PBVS should be more precise when using such a calibrated camera and gripper, because the error is computed in 6D space instead of in its projection and no additional end-effector tracker needs to be used, which would be another source of noise. Therefore, PBVS is used for the gripper, and since the 3D position of the target is already known, there is no disadvantage in using PBVS over IBVS for the main body as well. Adding additional constraints like collision avoidance or reaching a desired angle towards the target is easier with PBVS, which makes it a better fit for the RCLL and as a general

framework, and should result in a better solution.

In the following, we describe each part of the proposed method in more detail.

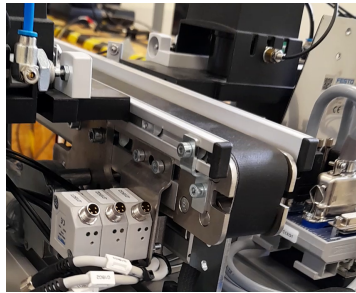
5.2 Object Detection

For PBVS, we need an object detector, which is capable of detecting every graspable object and every location an object should be placed down. The object detector for the RCLL needs to detect:

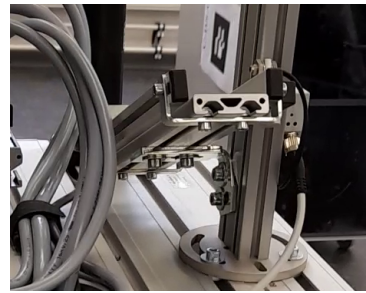
1. workpieces, as shown in Figure 13a, are the only graspable objects and are target of every picking action
2. conveyor belts, as shown in Figure 13b, are potential target locations for a grasped workpiece
3. MPS slides, as shown in Figure 13c, are the other potential target locations



(a) Red workpiece with blue ring and no cap on a conveyor belt.



(b) Front of a conveyor belt.



(c) Front view of an MPS slide.

Figure 13: Target objects needed for the VS approach for the RCLL.

5.2.1 Requirements

There are a few requirements for such an object detector. It needs to approximate a 3D target position from an image. The expected position for the target object is used to eliminate false positives. Because of this reason are false negatives way worse than false positives and, therefore, the detector should be very sensitive.

A weighted average is taken over the last 5 detected 3D positions to reduce detection variance. Past detections decrease in their weight from 40%, 30%, 15%, 10% up to 5%

over time. The target objects need to be detected and tracked reliably for this approach. In case the object is not detected, its expected position is used for the weighted average instead.

Missing detections in the "Search" state can be solved if the expected position of the target is known, since the robot can still move close to the target object. However, this approach cannot recover from target objects which are occluded while the robot is close up and will eventually fail.

5.2.2 3D Projection

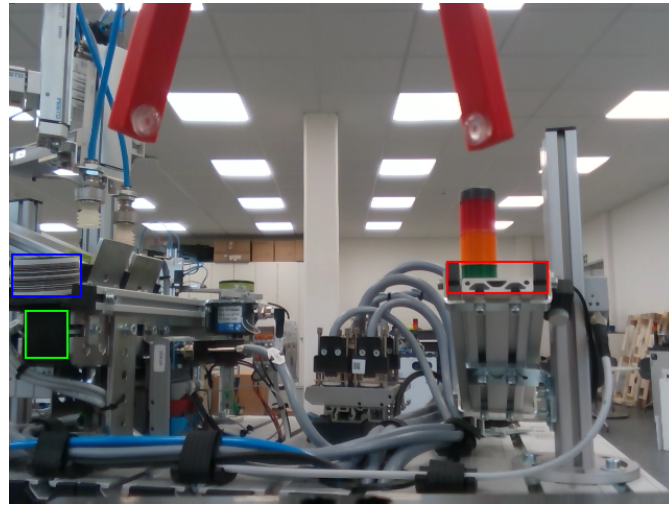


Figure 14: Image of a workpieces with a blue, a conveyor belt with a green, and an MPS slide with a red bounding box.

A calibrated camera in combination with an object detector is used to determine the target object position. Figure 14 shows the bounding boxes provided by the object detector. They fit the form of the target objects well and since the width of each workpiece, conveyor belt, and MPS slide is known, the bounding boxes can be used reliably to estimate the 3D object position using a triangulation approach, which we will cover in the following. This approach was chosen because calibrated cameras are needed for the required precision in hand-eye coordination for PBVS anyways and such object detectors are very common, fast to compute, reliable, and can be used for any type of object, which make them a good fit for this thesis anyways.

Next, we will make a derivation of how to project a bounding box into 3D space using the setup in Figure 15 in which a workpiece is projected on the image plane. Values in image space are measured in pixels while euclidean values are 3-dimensional and measured in meters.

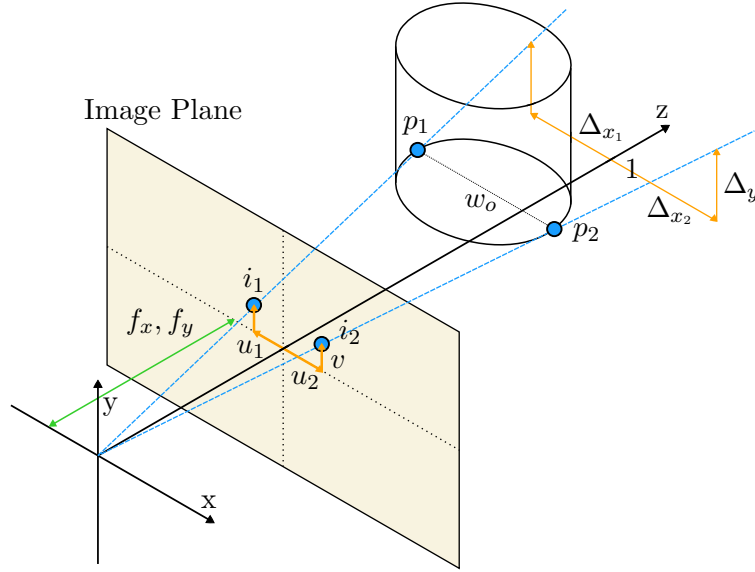


Figure 15: 3D projection of a workpiece.

Given the lower bounding box corners $i_1 = (u_1 \ v_1)^T$ and $i_2 = (u_2 \ v_2)^T$ in image space, where $v_1 = v_2 = v$ since the bounding box edges are parallel to the axis, the object width, w_o , and the focal lengths, f_x and f_y , we can derive the bounding box corner's 3D world positions $p_1 = (x_1 \ y_1 \ z_1)^T$ and $p_2 = (x_2 \ y_2 \ z_2)^T$ as follows:

The focal length measures the distance between the camera center, located at the 3D origin, and the image plane in pixels. Since pixels on the image plane are not required to be quadratic, the focal length for the x-axis, f_x , is measured in pixel width and the for the y-axis, f_y , in pixel-height.

First, we compute the direction from the camera center to the projection of the bounding box corners in x-direction, Δ_{x_1} and Δ_{x_2} , and in y-direction, Δ_y , normalized to a z-value of 1 in euclidean space for each point:

$$\begin{aligned}\Delta_{x_1} &= \frac{u_1}{f_x} \\ \Delta_{x_2} &= \frac{u_2}{f_x} \\ \Delta_y &= \frac{v}{f_y}\end{aligned}$$

In the following, we use the yaw between the workpiece and the camera, α , which can be computed using the x-direction of the projected bounding box middle point as follows:

$$\alpha = \arctan \frac{\Delta_{x_1} + \Delta_{x_2}}{2}$$

Since the bounding box edges are parallel to the axis, the following holds:

$$\begin{aligned}x_1 + \cos \alpha w_o &= x_2 \\z_1 + \sin \alpha w_o &= z_2\end{aligned}$$

We can compute the 3D positions of the workpiece corners given their direction from the camera and the distance between camera and workpiece:

$$\begin{aligned}x_1 &= \Delta_{x_1} z_1 \\z_1 &= \Delta_{x_2} z_2\end{aligned}$$

With these values, we can compute z_1 or similarly z_2 :

$$\begin{aligned}x_1 + \cos \alpha w_o &= x_2 \\ \Leftrightarrow \Delta_{x_1} z_1 + \cos \alpha w_o &= \Delta_{x_2} z_2 \\ \Leftrightarrow \Delta_{x_1} z_1 + \cos \alpha w_o &= \Delta_{x_2} (z_1 + \sin \alpha w_o) \\ \Leftrightarrow z_1 &= \frac{\cos \alpha w_o - \sin \alpha w_o \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}}\end{aligned}$$

We can compute p_1 and p_2 as well as their middle point, p_m :

$$p_m = \begin{pmatrix} x_m \\ y_m \\ z_m \end{pmatrix} = \begin{pmatrix} x_1 + \frac{\cos \alpha w_o}{2} \\ y_m \\ z_m \end{pmatrix} = \begin{pmatrix} \Delta_{x_1} z_m + \frac{\cos \alpha w_o}{2} \\ \Delta_y z_m \\ z_m \end{pmatrix} = \begin{pmatrix} \Delta_{x_1} (z_1 + \frac{\sin \alpha w_o}{2}) + \frac{\cos \alpha w_o}{2} \\ \Delta_y (z_1 + \frac{\sin \alpha w_o}{2}) \\ z_1 + \frac{\sin \alpha w_o}{2} \end{pmatrix}$$

Using z_1 , we get the following:

$$p_m = \begin{pmatrix} \Delta_{x_1} \left(\frac{\cos \alpha w_o - \sin \alpha w_o \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin \alpha w_o}{2} \right) + \frac{\cos \alpha w_o}{2} \\ \Delta_y \left(\frac{\cos \alpha w_o - \sin \alpha w_o \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin \alpha w_o}{2} \right) \\ \frac{\cos \alpha w_o - \sin \alpha w_o \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin \alpha w_o}{2} \end{pmatrix} \quad (1)$$

To track the workpiece position, we use the center of the workpiece base as target, since the bounding box middle point is changing with the amount of rings and the cap. Therefore, we additionally add half of the workpiece base height, h_o :

$$p'_m = \begin{pmatrix} x_m \\ y_m + \frac{h_o}{2} \\ z_m \end{pmatrix}$$

Since the conveyor belt and MPS slide do not vary in their size, we can directly use the y-value of the bounding box middle point for i_1 and i_2 as well as p_1 and p_2 respectively when projecting to the conveyor belt and MPS slide center.

All possible positions for workpieces, conveyor belts, and MPS slides are known. This additional knowledge is used to filter detector responses outside of the expected position within a threshold and allows to take the closest response to the expected position in case there are multiple.

5.2.3 YOLO

In the following, multiple choices for object detectors with bounding box outputs for the RCLL are discussed. In this thesis, a tiny version of YOLOv4 is used to detect workpieces, conveyor belts, and MPS slides. It is a state-of-the-art object detector, which is capable to be used in real time, can be trained to output tight bounding boxes, and is suitable to detect multiple object classes at once.

YOLOv4 is a deep network and takes too much time to compute on the robot to be used as object detector for a closed-loop VS approach, since the robot control relies on vision updates. Therefore, a faster object detector is needed. YOLOv3, which is less deep and, therefore, faster but less precise, was tested and was still too slow for real time use on the Robotino's CPU.

Usually, when referring to YOLOv4-tiny the smaller YOLOv4 variant with 2 detection layers, also called YOLO heads, is meant, but in this thesis, we used a YOLOv4-tiny with 3 detection layers and refer to it as YOLOv4-tiny. YOLOv4 has the same amount of detection layers, but consists of more layers in between and especially before the first detection layer. However, YOLOv4-tiny is fast enough to be used for this thesis and produces good results too.

Training YOLOv4-tiny started by creating a data set using smartphone images and labeling them by hand. Additionally, a YOLOv4 was trained using this data to help initializing the labels of new images. This lowered the time of the labeling process.

Furthermore, active learning was used to reduce the time to choose new images to label. A lot of possible images were captured by the camera of the robot while moving in front of the target and by performing the manipulation actions. These were labeled by YOLOv4-tiny and images with badly placed or missing bounding boxes were chosen to be used as part of the data set [9].

Augmentation was used on the labeled training set to improve robustness, variety, data size, and, with that, the performance of the trained network. For our augmentation, we used:

- a rotation by 0° , 90° , 180° , or 270° , such that we create 2 augmented images for each rotation per image in the data set
- cropping and padding randomly between 85% of their width or height up to 125% separately
- with a probability of 50% Hide and Seek augmentation, in which rectangular blocks from the image are deleted as long as the area is unlabeled
- with a probability of 50% coarse Dropout, which cuts out small circular regions

- Cut and Mix

When applying Cut and Mix, image parts with no information due to coarse Dropout, Hide and Seek, and padding are replaced with random images from the COCO data set [20], which contains a collection of many different objects of which none is a target object of this thesis. Therefore, the labels only need to be adjusted to the cropping and padding [26].

Additionally, the Darknet framework [31] was used to train YOLOv4 and YOLOv4-tiny, which randomly adjusts the image saturation, exposure, and hue, applies Gaussian noise, and mosaic augmentation in every training iteration [4]. In mosaic augmentation four images from the training set are cut and mixed together to create a new image.

5.2.4 Alternatives

Since every workpiece has a barcode around it, a barcode detection algorithm as described by Zamberletti et al. [44] could be used to detect the workpieces. This approach has the advantages of publicly available data sets and pretrained workpiece detectors as well as simple low-computational approaches while object detectors trained to detect new classes from scratch such as YOLO requires data sets for each target object and are computationally more expensive.

We did not use barcode detectors in this thesis, because they could be too sensitive and misclassify a barcode around a workpiece as two separate barcodes, since there can be a gap in between. However, its biggest problem is that the conveyor belt and MPS slide do not contain barcodes and are, therefore, not detectable with this approach, which means an additional classifiers is needed for these classes anyways.

There are multiple approaches to get a 3D position from an image, which are all feasible when using the proposed approach. A 3D model-based tracker, which uses edge detections to match objects in images to their 3D model, could be used to get a 6D object pose. This approach is most likely not reliable enough for the RCLL, because many edges of the conveyor belt can get lost on images depending on the lighting. Additionally, the wiring and other MPS parts can change between different MPS and result in different edges. Furthermore, there are similarly shaped objects to target objects like cylindrical lights, which could be mistaken for a workpiece, especially since workpieces can have multiple heights, which would require a different 3D model for each. Rings can also end up skewed on the workpiece and could result in missing detections.

Alternatively, the RGB-D camera could have been used by computing the median depth value over a YOLO bounding box and projecting it in direction of the bounding box middle point to estimate the 3D object position. This approach is not used in this thesis, because RGB-D data is less reliable at short distances and its results are worse at black

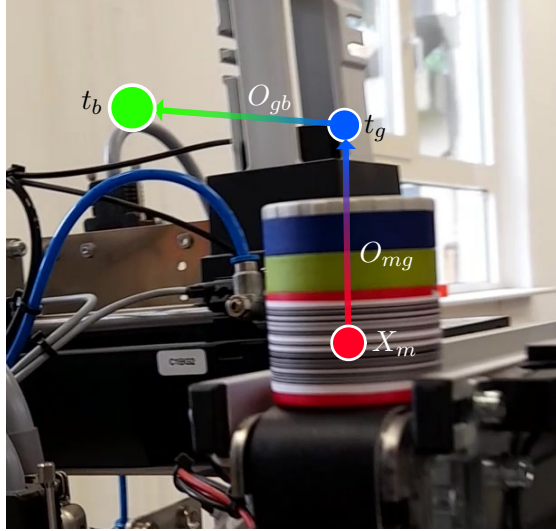


Figure 16: Workpiece with middle point x_m and target poses t_g and t_b .

or reflective materials, which is a frequently occurring issue in the RCLL [17]. The object position based on YOLO could also be used as initial estimate for a 3D model-based tracker or for an ICP, but was also refused because of its reliance on point cloud data.

5.3 Error Calculation

After the 3D target object position, represented by the object middle point, X_m , is computed, it is used to derive the target poses for the gripper, t_g , and the main body, t_b . After these target poses are reached, using PBVS for the body and gripper, the gripper routine can be safely executed.

Figure 16 shows the target poses in relation to the target object middle point, X_m , for a workpiece. X_m is defined as a pose in the special Euclidean group SE^3 and defined by:

$$X_m = (x_m \ y_m \ z_m \ 0 \ 0 \ 0)^T \in SE^3 \quad (2)$$

SE^3 is a manifold with structure $\mathbb{R}^3 \times SO^3$ and is used to define 6D poses consisting of translations and rotations in 3D space. Translations are described by \mathbb{R}^3 , e.g. as $x_m, y_m, z_m \in \mathbb{R}$, which is the position of X_m in 3D space. Rotations are described by the special orthogonal group SO^3 as Euler angles around the z-axis as yaw $\in [0^\circ, 360^\circ]$, around the y axis as pitch $\in [0^\circ, 360^\circ]$, and around the x direction as roll $\in [0^\circ, 360^\circ]$ [3]. X_m 's rotation is ignored and set to 0.

The target object middle point is measured relative to the robot's main body middle

point, which is, for convenience, defined as the origin for the error computations:

$$X_b = (0 \ 0 \ 0 \ 0 \ 0 \ 0)^T \in SE^3 \quad (3)$$

O_{mg} , the offset from X_m to t_g , is a vector pointing upwards by $z_{mg} \in \mathbb{R}$ given by:

$$O_{mg} = (0 \ 0 \ z_{mg} \ 0 \ 0 \ 0)^T \in SE^3 \quad (4)$$

The target gripper pose, t_g , is located over the workpiece and computed by:

$$t_g = X_m + O_{mg} = (x_m \ y_m \ z_m + z_{mg} \ 0 \ 0 \ 0)^T \in SE^3 \quad (5)$$

Rotations are ignored since the gripper only has 3-DoF.

The offset O_{gb} from t_g to the target body pose, t_b , sets the z-value of t_b to 0 and is placed a predefined distance, d_o , in normal direction of the MPS front, defined as yaw relative to the robot by η , away from the target object, given by:

$$O_{gb} = (d_o \cos(\eta) \ d_o \sin(\eta) \ -z_m - z_{mg} \ 0 \ 0 \ \eta + 180^\circ)^T \in SE^3 \quad (6)$$

The direction the robot body is supposed to face when reaching the target body pose is $(0 \ 0 \ \eta + 180^\circ)^T \in SO^3$, which is opposite to the MPS orientation.

The target body pose, t_b , is computed by:

$$t_b = t_g + O_{gb} = (x_m + d_o \cos(\eta) \ y_m + d_o \sin(\eta) \ 0 \ 0 \ 0 \ \eta + 180^\circ)^T \in SE^3 \quad (7)$$

The main body is always assumed to stand on the ground, which is why its z-value is 0. This was changed in Figure 16 to fit the target poses into the image. All other rotations are ignored, since the body has only one rotational dimension.

The MPS's yaw can be approximated using laser data, but in this thesis we use the MPS's orientation in the map relative to the robot's yaw using its localization data.

To determine a suitable position for the target body pose in relation to the gripper pose, the reachable workspace needs to be known. The target gripper pose should be located in an easily graspable area even with slight variations through imprecise body movement and the body pose should be placed accordingly.

With these target poses, the relative body and gripper error, e_b and e_g , can be computed. They are the target of minimization of the PBVS and are updated frequently, although a "look and move" approach could be applied as well, in which case the error would only be updated after each movement.

The current gripper pose, X_g , which is located at the middle point between the gripper fingers, is defined as:

$$X_g = (x_g \ y_g \ z_g \ 0 \ 0 \ 0)^T \in SE^3 \quad (8)$$

We compute the gripper error for each direction as difference between the current gripper and the target gripper value in the particular direction by:

$$e_g = \begin{pmatrix} e_{x,g} \\ e_{y,g} \\ e_{z,g} \\ 0 \\ 0 \\ 0 \end{pmatrix} = |t_g - X_g| = \begin{pmatrix} |x_m - x_g| \\ |y_m - y_g| \\ |(z_m + z_{mg}) - z_g| \\ 0 \\ 0 \\ 0 \end{pmatrix} \in SE^3 \quad (9)$$

Since the robot body is defined as the origin of the coordinate frame for the error calculations, the body error for each direction is the absolute value of the target body value for this direction as seen here:

$$e_b = \begin{pmatrix} e_{x,b} \\ e_{y,b} \\ 0 \\ 0 \\ 0 \\ e_{yaw,b} \end{pmatrix} = |t_b - X_b| = |t_b - 0| = |t_b| = \begin{pmatrix} |x_m + d_o \cos(\eta)| \\ |y_m + d_o \sin(\eta)| \\ 0 \\ 0 \\ 0 \\ \eta + 180^\circ \end{pmatrix} \quad (10)$$

5.4 Controlling

After an error is calculated, a low-level controller uses this data to control the main body and gripper to minimize this error. This process can be done using closed- or open-loop VS. Open-loop VS requires less computations and, therefore, less energy but relies on high accuracy in the low-level navigation and object detection. It assumes a static environment while closed-loop VS adapts to moving objects and is more precise.

If the camera is not attached to the gripper, the control loop can switch from closed-loop in state "Move Body and Gripper" to open-loop in "Fine-tune Gripper" since the object might not move anymore after the robot reached its target body pose. In this thesis, closed-loop VS is proposed for every state, since environments are usually dynamic, especially if the robot can accidentally move objects by itself.

5.4.1 Main Body Control

For the proposed approach, only rough localisation and camera data is used for all movements to show its reliability, precision, and overall capability.

In the "Search" state, we use the robot's standard navigation method to move towards the expected object location. If the object is in the camera view and the distance between

it and the robot is below a certain threshold, we transition into the "Move Body and Gripper" state, where the robot control is completely done by PBVS unless the object is not detected for multiple frames. In this case, we transition back to the "Search" state and use the internal navigation again.

When using PBVS for the body, it is an eye-in-hand approach since the camera is mounted on the main body.

The target body pose is computed to leave room for small errors by choosing a distance between the robot and MPS of 4.5 cm with a tolerance in x direction of 1 cm, 0.5 cm in y, and 0.01 rad in its yaw. If the object moved out of the grippers reachable workspace, the "Move Body and Gripper" state is selected again to move the main body to a suitable location.

The robot is accelerating and decelerating linearly and its velocity is computed for x and y direction as well as for its rotation separately.

When accelerating, the robot's main body velocity in direction $d \in \{x, y, yaw\}$, $v_{d,acc}$, is computed using time step t and an acceleration factor $f_{d,acc}$ by:

$$v_{d,acc} = t f_{d,acc} \quad (11)$$

When decelerating, the robot's velocity, $v_{d,dec}$, is computed by:

$$v_{d,dec} = e_{d,b} \frac{v_{d,max}}{dist_{d,dec} f_{d,dec}} \quad (12)$$

This equation uses the base error in direction d , $e_{d,b}$, as difference between current and target body value in this direction, maximum velocity, $v_{d,max}$, the required distance to decelerate, $dist_{d,dec}$ and a deceleration factor, $f_{d,dec}$.

To compute the resulting robot body velocity in direction d , $v_{d,b}$, the maximum and minimum velocity in direction d , $v_{d,max}$ and $v_{d,min}$, as well as $v_{d,acc}$ and $v_{d,dec}$ are used:

$$v_{d,b} = \max(v_{d,min}, \min(v_{d,max}, v_{d,acc}, v_{d,dec})) \quad (13)$$

5.4.2 Gripper Control

In the "Move Body and Gripper" state, the gripper movement is independent of the main body and assumes that the body reached its target pose already. Therefore, the gripper might reach its relative target pose while the body is still reaching for its target pose. This allows for overall faster whole-body movements and faster manipulation times. Even if the main body reached its target pose before the gripper target pose was reached, the gripper moved closer to its target pose in the meantime. Depending on the

body's precision and the chosen threshold for the acceptable body end pose, the gripper just needs to move slightly to compensate for that.

When using PBVS for the gripper, it is technically an eye-in-hand approach since the camera moves in x and y direction with the gripper. However, it is similar to an end-point open-loop eye-to-hand approach, because the camera does not move together with the gripper in the z-axis and the gripper is not always in view.

The gripper's default position is placed over the maximum workpiece height such that there are no possible collisions between workpiece and gripper, which makes additional collision avoidance unnecessary.

When moving the gripper, the camera attached to it moves with it and this can result in flickering bounding boxes which results in more noisy target poses. To account for that, the threshold for adjusting the target gripper pose while moving is higher than usual. In x direction, it increases from 5mm to 10mm, it stays in y direction at 1mm, and it increases from 1mm to 1.5mm in the z-axis.

The gripper is controlled by a stepper-motor for each direction $d \in \{x, y, z\}$.

The amount of steps required to stop, s_{stop} , [2] can be computed with:

$$s_{d,stop} = \frac{v_{d,g}^2}{2a} \quad (14)$$

It uses the current velocity in direction d , $v_{d,g}$, and an acceleration constant, a .

The timer count, c_i , for the delay between physical step i and $i + 1$ is computed with the general-purpose ramp algorithm:

$$c_{d,i} = \max\left(c_{min}, c_{d,i-1} - \frac{2c_{d,c-1}}{4n_{d,i} + 1}\right) \quad (15)$$

The equation consists of the minimum timer count, c_{min} , the previous timer count, $c_{d,i-1}$ and the step number $n_{d,i}$, which increases with i and determines whether the stepper motor is accelerating or decelerating in direction d by changing its sign to be negative if the amounts of steps needed to stop is equal or greater than the difference to the target value in steps, defined as error $e_{d,g}$ [2]:

$$n_{d,i} = \begin{cases} |n_{d,i}| & s_{d,stop} < e_{d,g} \\ -|n_{d,i}| & s_{d,stop} \geq e_{d,g} \end{cases} \quad (16)$$

The current velocity for direction d is computed by dividing the timer frequency, f , by the timer count [2]:

$$v_{d,g} = \frac{f}{c_{d,i}} \quad (17)$$

The "Gripper Routine" state is reached after main body and gripper are positioned at their respective target poses. Since the gripper is now in a predefined location relative to the object, a simple routine can be executed in which the object is manipulated. Occlusions of the target object by the gripper are expected in this state and do not result in a transition to another state.

This thesis uses a routine for grasping and a routine for putting actions. If a workpiece should be grasped, the gripper moves down, grasps, and moves slightly up. In case of a putting action, the gripper moves slightly down, opens its grasp, and moves up again. When picking up a workpiece, the routine uses the last detected workpiece position as its target when moving down. Apart from this, each distance moved by the gripper is fixed and predefined. After the routine is done, either the grasping or putting action was successfully, or a failure if the routine could not be properly executed.

5.5 Simulation

The RoboCup Logistics League simulation in Gazebo [46] allows us to simulate any setup, which makes continuous development of Robotino actions easier, faster and safer. However, it was not used for any kind of evaluation for this thesis.

To be usable for this thesis, a model of the currently used gripper with moving capability and physically-based gripping was implemented.

There is no object detector used in the simulation. Therefore, the manipulation approach was tested with the true object position and no noisy object tracker before testing the approach on a real robot.

6 Evaluation

The proposed approach is tested and evaluated on real robots in the settings from RoboCup Logistics League. The goal of the approach is a robust and fast manipulation framework for 3-DoF robot bodies with precise grippers. The main evaluation criteria are average success rate and average completion time of successful actions.

Additionally, the environmental robustness is tested by increasing the height of the MPS, offsetting parts of it and changing the lighting conditions. The object detector is also evaluated separately in object detection accuracy and speed.

6.1 Object Tracking

3922 labeled images were used for the training of the final version of YOLOv4 and YOLOv4-tiny. Using augmentation techniques explained in Chapter 5.2.3, each labeled image was used in 8 different variants, resulting in 28976 training images and 300 test images. The labeled images consist of 1147 images from a smartphone camera, 1102 from the used Realsense camera at its usual position shown in Figure 18a and 1673 from the final position shown in Figure 18c.



Figure 17: Image captured while main body was shaking.

Moving the robot can cause motion blur, especially if the robot's main body is shaking. Figure 17 shows the resulting motion blur of a shaking main body. This makes detecting the object in the image harder and placing their bounding box ambiguous and challenging even for a human. When initially training YOLOv4-tiny with smartphone images, there were images with intentional motion blur, made by moving the phone. But their motion blur is only minor compared with the level of motion blur while the main body is shaking. After including many motion blurred images from the Realsense camera in the training set, the results of YOLOv4-tiny on these kinds of images in the data set and in real life were significantly improved to a point where it is not problematic for the approach.

Choosing a camera with a low exposure time can reduce the amount of motion blur heavily and improve the performance of YOLOv4-tiny when driving.

6.1.1 Camera Positioning



(a) Camera positioned at the gripper and used for the ICP approach. (b) Fixed camera with large distance between it and target object. (c) Camera positioned at the gripper with an offset to the position in (a).

Figure 18: Camera positions used in this thesis.

The ICP approach uses the camera position shown in Figure 18a where it is placed on the gripper, such that it moves together with the gripper's x and y axis. The camera is rotated by 180° around the x axis, or roll-axis, such that the image is upside down. Its height is below the conveyor belt and, therefore, rotated by 8° around its y axis, or pitch-axis, such that the camera is tilted upwards and can see the conveyor belts and MPS slides. This is a good camera position for the ICP approach since the ICP approach only needs to detect these objects and has a reasonable distance towards them. However, for the proposed approach, this was problematic, because when placing the gripper over the workpiece or MPS slide, the camera is positioned so far forward that workpieces and slides are not visible anymore.

As a potential solution, the target gripper position can be adjusted to be a few centimeters further in front, where the object is still clearly visible. The gripper routine afterwards would need to adjust for it and move a bigger distance as an open control loop. This solution was not chosen since a main reason for the VS approach was its closed-loop aspect and reactivity.

To solve this problem, the camera position shown in Figure 18b was tried. It is a static position at the upper part of the robot with a higher distance towards objects as before.

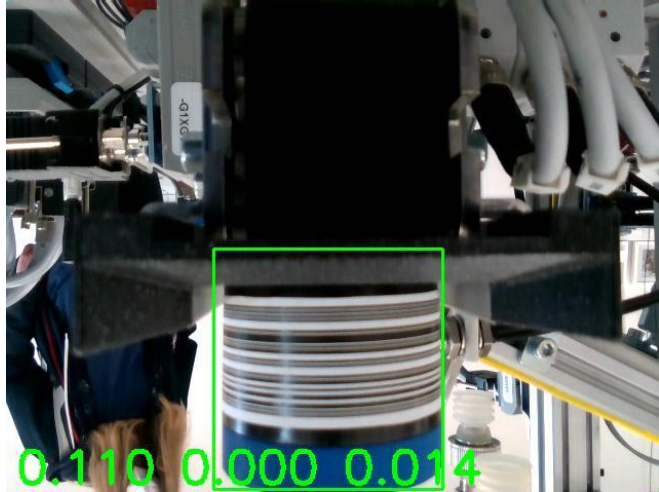
In its new position, the camera is rotated anticlockwise by 90° around its x-axis. From this position, the objects are always clearly visible.

However, two problems occurred with this change. Firstly, YOLOv4-tiny had issues placing bounding boxes as accurately as for the initial position. This is mainly caused by the training set, since the new position is significantly different from the last one and, especially, its 90° rotation results in differences between any occurrence of the target objects in the training set and in real life even though the augmentation is scaling the training images in many ways. No image was captured from the Realsense in an 90° angle since they were taken from the initial camera position. This problem can be fixed by taking new training images from the new camera position. Secondly, having a further distance to the target object results in higher errors when projecting bounding boxes into 3D positions.

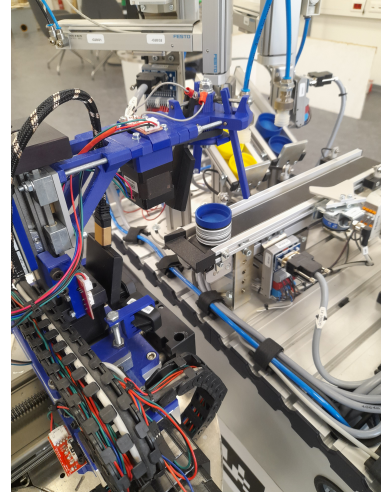
As mentioned, retraining would probably have fixed the misalignment issue in practice, but the scaling of bounding box failures remains and can happen if the robot is capturing images which differ enough from the training set. The issue can be fixed by using an image with less space around the target object for YOLOv4-tiny, which results in similar image as a closer camera. This can be done by cropping the image to the expected object position using the latest response. This is only possible to a degree since the resolution is reducing with further distances. Using a different camera with a higher focal length, and with that, a lower field-of-view from far away, would result in a similar effect. Using a camera with a lower field-of-view is preferably since the resolution is staying high. A combination of both would probably be even better, but in this thesis neither of these solutions were tested. Errors in the 3D projection are increasing with distance, too, if the camera calibration is not perfect. Therefore a camera closer to the object should result in preciser VS than one far away.

This thesis solved the camera positioning problem by mounting it at the gripper position it was initially placed, raising its height by 3.3 cm, and removing its rotation around the y axis, which results in no upwards tilt. The position can be seen in Figure 18c.

The bounding box error tolerance when using this camera position can be seen in Figure 19a, where a YOLOv4-tiny result with a slight mistake in bounding box width and height is shown. The coordinates at the bottom of the figure as well as every following labeled image with coordinates at the bottom are measured in a coordinate system where the robot's main body with a height of 0 is the origin. When using the object tracker, only the object of the target class closest to the expected position is detected and receives a bounding box. The first value, in the x-axis, measures the distance to the object, the second value, the y-value, captures the offset to the right, and the final value shows the object's height in the z-axis. The resulting alignment in Figure 19b is hardly distinguishable from a perfect alignment. In this case, the workpiece is clearly in the tolerance zone for the gripper routine to grasp it.



(a) Object tracking of workpiece with slightly wider bounding boxes from final camera position.



(b) Robot pose after finishing alignment with slightly wider bounding box from final camera position.

Figure 19: The image in (a) is taken from the camera in the situation in (b).

YOLOv4-tiny was not retrained for the new camera position while giving the result in Figure 19a. For this purpose, mainly close-ups of the target objects were captured since YOLOv4-tiny was already performing well overall, except for close distances to the target objects where it performed poorly. Problematic was especially detecting the MPS slide in close distances, where the camera would end up when aligning. Therefore, the amount of additional MPS slide images for the data set was increased compared to the data before.

After the training, YOLOv4-tiny's results near target objects were very reliable, but the performance for further distances decreased in practice. After a distance of about 60 cm, YOLOv4-tiny only detects target objects reliably if they are in the middle of the image. This probably happened because of the data set balance and its bias towards close-ups. Since the robot is positioned in this range through navigation before switching to the VS, this is no problem in practice.

Even at the minimal distance to the MPS, which is usually never reached when applying this thesis approach, the camera in the position shown in Figure 18c can see MPS slides completely, workpiece bases nearly completely, and most of the conveyor belt in this position. In close range, it cannot see rings and caps on workpieces as well as the bottom part of conveyor belts. This is not ideal, but it suits any common manipulation situation appearing in the RCLL scenario. Since the 3D projection is only using the bottom bounding box corners, the height of workpiece bounding boxes can be ignored. However, this is a limitation since the workpiece height could be used to set the gripper

target pose closer to workpieces if they have less than 3 rings mounted. The gripper routine used for this thesis assumes each workpiece to have 3 rings and places the gripper even for workpieces without rings at a high pose. The gripper needs to move this extra distance while executing its routine. When putting a workpiece down, the gripper first lifts itself over 3 potential rings before turning back for the same reason.

The positioning of the camera on the gripper has the disadvantage of requiring the gripper pose to compute the object position relative to the robot's main body. Therefore, the gripper controller in Fawkes was adapted to update the gripper pose continuously.

Another disadvantage are bounding boxes varying their sizes while moving the camera and resulting in a higher variety in 3D object positions while aligning the gripper. The gripper is adjusting to the responding target positions accordingly. These adjustments are done in split seconds with the current state of YOLOv4-tiny, but when the camera was placed in a different position, the variance in the bounding boxes could cause the gripper to oscillate. This problem can be solved by using a fixed camera position or an accurately trained YOLOv4-tiny.

6.1.2 Object Tracking Speed

We first compared Carologistics' Robotino's CPU performance with an Intel Neural Compute Stick 2 [8] and came to the conclusion that the CPU forwards neural networks faster. When choosing a suitable neural network for the object tracking, the images per second were the first deciding factor. To run VS in real time, a rate of at least 10 images per second were targeted. YOLOv4 and YOLOv3 are not fast enough on the Intel Core i5-8400H, the CPU of the Robotino. After comparing YOLOv4 and the chosen YOLOv4-tiny variant on the AMD Ryzen 5 1600, it was clear that YOLOv4 with its 0.904 images per second was clearly not fast enough. YOLOv4-tiny, however, achieves 18.904 images per second on average, which is better than our required minimum.

The object-tracker used in this thesis takes an image from the camera, runs YOLOv4-tiny on it, computes the 3D object position, and publishes the target poses. This whole process takes 0.033719 seconds on the Intel Core i5-8400H when applying VS. This would result in 29.66 images per second, but it is integrated in a sense-think-act loop and publishes 18.904 target positions per second on average in practice.

6.1.3 Evaluation Metrics

Before showing the evaluation of the object detection, we need to define a few metrics.

The Intersection over Union (IoU) [35] for two bounding boxes is the area of overlap divided by the area of union as presented in Figure 20. More precisely, IoU for comparing

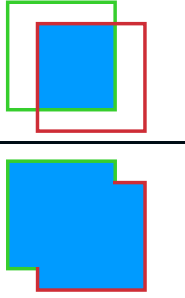
$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 20: Intersection over Union (adapted from [30]).

the similarity between two arbitrary volumes $A, B \in \mathbb{R}^n$ is defined as:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (18)$$

The precision, P , [30] of a classification is defined by:

$$P = \frac{TP}{TP + FP} \quad (19)$$

It uses the amount of correct detections or true positives, TP, and wrong detections or false positives, FP.

The recall, R , [30] of a classification as defined by:

$$R = \frac{TP}{TP + FN} \quad (20)$$

It measures the amount of true positives compared to all detections, which is the sum of all true positives and all missing detections, or false negatives, FN.

The F1-score [36] is often used to evaluate classification algorithms. It is defined by precision and recall:

$$\text{F1-score} = \frac{2 P R}{P + R} \quad (21)$$

When using a YOLO network, each bounding box has a confidence value between 0 and 1 for every class, which add up to 1 and can be seen as the predicted probability for the object to be of the particular class. The class with the highest confidence is chosen as the class of the bounding box.

When using a YOLO network, precision-recall curves (PR-curves) [30, 11] can be defined for each object class and represent the connection between precision and recall over the

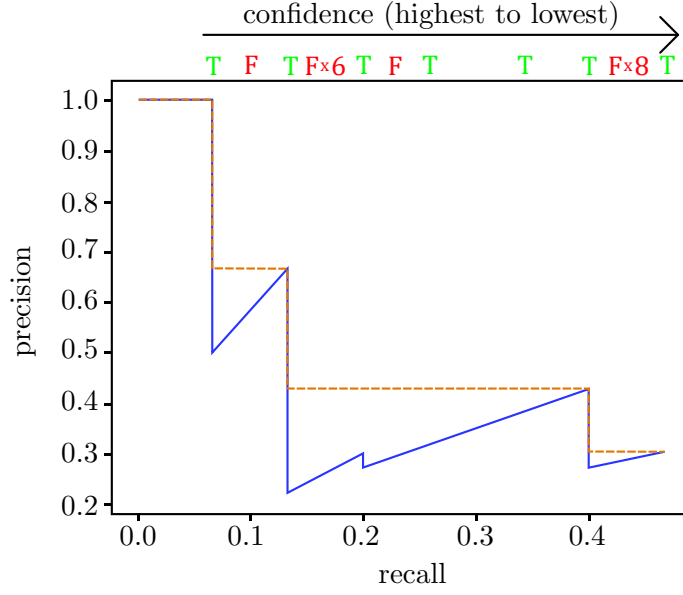


Figure 21: TP’s and FP’s are ordered from highest to lowest confidence at the top. The corresponding PR-curve for 24 labeled objects is shown below. The blue curve represents the sequence of precision and recall values for these TPs and FPs. The orange curve is the interpolated precision, which is the maximum precision for each object of same or lower recall (adapted from [30]).

given data. Figure 21 shows such a curve for an example with 7 true positives and 17 false positives. Each detection is ordered by its confidence from highest to lowest.

For each detection i , the recall up to i , R_i , which is the recall of detections 0 to i , is computed as well as its precision $P(R_i)$. The blue line in the image is the plot created by these values. The dotted orange line shows the interpolated precision, which is achieved by taking the highest precision for the same or lower level of recall [30, 11].

With such a PR-curve, we can compute the average precision (AP) [30] for one object class as the area under the interpolated precision curve:

$$AP = \sum_n (R_{n+1} - R_n) P_{ip}(R_{n+1}) \quad (22)$$

The interpolated precision for the recall up to $n + 1$, $P_{ip}(R_{n+1})$ is computed as follows:

$$P_{ip}(R_{n+1}) = \max_{\tilde{R}: \tilde{R} \geq R_{n+1}} P(\tilde{R}) \quad (23)$$

To compare the precision of multi-object detectors, the mean average precision, mAP ,

[30] over all equally weighted classes is used:

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (24)$$

AP_i is the average precision of the i th of N classes. In our case $N = 3$ applies, since our target objects are either workpieces, conveyor belts, or MPS slides.

6.1.4 Object Detection Accuracy

We used the darknet framework [31] for the YOLO evaluation, which applies non-maximum suppression with an IoU of 25% before matching bounding boxes to ground-truth labels. In this process, bounding box pairs of the same class, which have an IoU of at least 25%, are avoided by removing the one with the lower confidence.

Metric	TP	FP	FN	P	R	$F1$	AP_{wp}	AP_c	AP_s	mAP
YOLO ₅₀	559	26	11	0.96	0.98	0.97	0.9966	0.9888	0.9804	0.9886
Tiny ₅₀	520	48	50	0.92	0.91	0.91	0.9720	0.9526	0.8688	0.9311
YOLO ₇₅	529	56	41	0.90	0.93	0.92	0.9741	0.9147	0.8217	0.9035
Tiny ₇₅	423	145	147	0.74	0.74	0.74	0.7976	0.7553	0.4263	0.6597

Table 1: Evaluations for YOLOv4

To test the accuracy of YOLOv4-tiny, we compare its performance on the 300 labeled images in the test set with YOLOv4, which was trained on the same data set as YOLOv4-tiny. The results can be seen in Table 1, which shows the amount of true positives, false positives, and false negatives, the precision, recall, and F1-score (F1), as well as the average precision for workpieces, conveyor belts, and MPS slides separately and the mean average precision. Each row starts with the network name, where YOLO is YOLOv4 and Tiny is YOLOv4-tiny. The number next to the network name is its minimum IoU between a detection and the ground truth to be considered a true positive, which is given in percent.

A minimum IoU of 50%, which is used in the first two rows, is a good measurement for object trackers overall. The mAP at a minimum IoU of 50% is the most representative metric for bounding box detection accuracy in this thesis [4]. YOLOv4 performs well with only 26 false positives, 11 false negatives, a F1-score of 97%, and a mAP of 98.86% while being better in every metric than YOLOv4-tiny. This is expected, since they are similar networks, despite YOLOv4 being deeper and more expensive to compute. Considering its fast computation times, YOLOv4-tiny shows great results with only 22 false positives and 39 false negatives more than YOLOv4, a F1-score of 6% less and a mAP of 93.11%. Both networks detect workpieces best, conveyor belts second best, and MPS slides the worst. YOLOv4 has a nearly perfect AP_{wp} of 99.66% and a high

AP_c of 98.88% as well as AP_s of 98.04% while YOLOv4-tiny has a high AP_{wp} of 97.20% and a quite high AP_c of 95.26%, too. However, YOLOv4-tiny's AP_s of 86.88% shows quite a difference. This is probably caused by more images of workpieces and conveyor belts than MPS slides in the data set. Additionally, workpieces and their rings are easily distinguishable from the environment since they have distinct shapes and colors.

When comparing YOLOv4 and YOLOv4-tiny with a minimum IoU of 75%, we are only comparing detections with very precise bounding boxes. Both are performing worse in every metric than with a minimum IoU of 50% as expected. YOLOv4's mAP decreased from 98.86% to 90.35%. Those values are still high, proving that the bounding boxes placed by YOLOv4 are mostly very precise. Especially AP_{wp} stayed very high with 97.41% while AP_s decreased fairly low with 82.17%, while AP_c is between both with 91.47%. YOLOv4-tiny performs comparatively worse on a minimum IoU of 75%, which can be seen in its mAP decreasing from 93.11% to 65.97%. Especially AP_s got a lot worse by decreasing from 86.88% to 42.63% while AP_{wp} only decreasing to 79.76%, and AP_c to 75.53, which is close to each other. This shows that YOLOv4-tiny places its bounding boxes less precisely than YOLOv4, especially for the MPS slide.

False negatives of the target object result in missing detections and are usually more problematic than false positives. False positives are only a problem if their 3D projection is closer to the expected position than the actual detection of the target object, which never turned out to be a problem for YOLOv4-tiny in this thesis.

YOLOv4's accuracy is very well, but we cannot use it as object detector in real time since its way too slow. However, with only 5.75% less than YOLOv4 in the mAP at minimum IoU of 50%, YOLOv4-tiny performs good too and is easily fast enough for this thesis approach. YOLOv4-tiny's bounding boxes are less precise than YOLOv4's, especially for MPS slides, but they are precise enough for this thesis approach, which will be seen in the evaluation of the whole approach.

6.1.5 Sensing

In this thesis approach, imprecise localization is used to roughly position in front of the target object. If the robot is close enough to the target and is detecting it reliably, in this case successfully detecting the target in 3 consecutive images, the VS moves the robot and gripper purely on image data. This demonstrates the capability of the approach, but can be dangerous if a workpiece is placed too far back on the conveyor belt or shelf, since the target body pose will be placed too close to the MPS and results in the robot driving into the station. Similarly, if the object tracker is detecting the object too far behind its actual position, the robot can drive into the machine, too. In our evaluation this happened in 4% of cases and can be fixed by detecting the MPS using the laser sensors and stopping the robot's body if it drives too close to it.

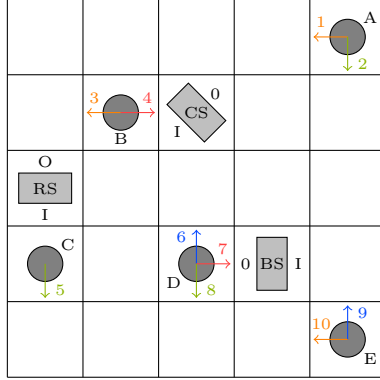


Figure 22: Field configuration for the main evaluation with a CS, an RS, a BS, and 10 starting positions for the robot.

Another problem occurs if the robot’s localization is too imprecise or if the MPS moved too far from its intended position. In this case, the expected position and the actual object position can deviate too much and we loose track of the object. A difference higher than 5 cm can already cause the robot to pick up the wrong workpiece from the shelf, but this never happened in the evaluation. This problem is no issue if laser sensors are used to detect the MPS and use this data to compute the expected object position relative to the MPS whenever the robot is in front of the target MPS.

6.2 Evaluation of VS and ICP

The main evaluation in this thesis consists of a sequence of tests in a known environment where the VS approach (VS) is compared against Carologistics’ current ICP-based approach (ICP). The setup is shown in Figure 22 and consists of a 5×5 m field, which is the same size as the field used in the RoboCup Logistics League 2021. A cap station (CS), a ring station (RS), and a base station (BS) are placed on the field. Their positions, and rotations are shown in the image as well as their input and output sides, which are marked as "I" and "O" respectively.

A starting position is defined by a 2D position, indicated by a circle in the figure with a letter between A and E, and an orientation, defined by a colored arrow pointing in the direction the robot is supposed to face. A same-colored number is placed next to the arrow which is the indicator for that particular starting pose. There are 10 starting poses in total and each is used once for every of the 10 possible manipulation targets of the machines:

- I picking up the workpiece from the left shelf spot of the CS
- II picking up the workpiece from the conveyor belt at output side of the RS

- III picking up the workpiece from the conveyor belt at input side of the BS
- IV picking up the workpiece from the middle shelf spot of the CS
- V picking up the workpiece from the conveyor belt at output side of the BS
- VI picking up the workpiece from the right shelf spot of the CS
- VII picking up the workpiece from the conveyor belt at output side of the CS
- VIII putting the workpiece on the conveyor belt at input side of the CS
- IX putting the workpiece on the conveyor belt at input side of the RS
- X putting the workpiece on the MPS slide of the RS

This results in 100 scenarios in total per approach. The time for each scenario starts with moving the robot and ends either after the action fails, a workpiece or a part of it is dropped, or successfully executing the action. An action is successful after the algorithm returns a success and if the manipulation was successful, which is the case for a picking action if the workpiece is picked up stable and the gripper is holding the workpiece only by its base and not its rings. Additionally, the amount of rings and caps of the workpiece need to stay the same.

Before performing a putting action, the robot needs to pick up a workpiece through a picking action. Therefore, the robot always starts with a picking action and followed by a putting action until there is no putting action left. In this case, we check if the workpiece is picked up properly and remove it from the gripper. A workpiece is supposed to stay on the conveyor belt or MPS slide after a successful putting action.

Only workpiece bases without caps are located on the BS. On the shelf of the CS are only workpiece bases with mounted caps, and on the output of the ring- and CS is a randomized workpiece mounted with between 0 and 3 rings and potentially a cap. The colors of each part is randomized and the resulting workpieces are equivalent between both evaluations of the approaches.

We captured 4 times for both approaches:

- total time of all phases
- positioning time, in which the robot is navigating to the machine
- alignment time, in which the robot aligns to its target
- routine time, in which the robot grasps or places the workpiece

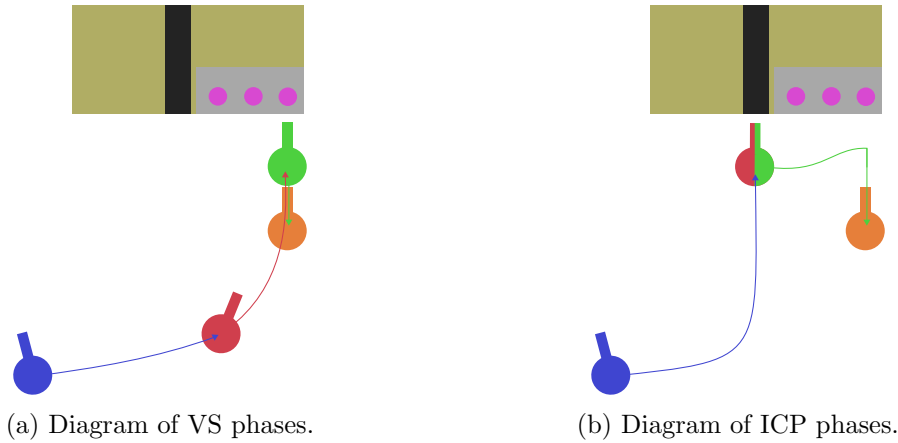


Figure 23: Top down perspective of picking a workpiece from the right shelf spot of the CS using VS (a) and ICP (b). The colored points represent the main body pose at a phase transition with a rectangle indicating the direction the robot is facing.

Figure 23a shows the phases while picking a workpiece from the right shelf spot using VS and Figure 23b shows the phases of the ICP approach on the same action. Picking a workpiece from the shelf shows the biggest differences between the approaches since ICP cannot detect workpieces. The positioning phase starts from the blue pose and ends at the red one. The red pose shows the starting position of the alignment phase, which is the same as its end pose for the ICP approach. The routine phase's starting pose is the green one and its end pose is the orange one, which ends the whole action. The total time starts with the blue pose and ends with the orange one too. Both poses are the same for both approaches, which makes this time the most comparable.

The total time is split into the positioning, alignment, and routine phase. It is the most important time for the comparison and the others are mainly for analysis. The positioning phase starts together with the total time and ends if the robot is in front of the target object, such that it switches from navigation to VS control for the VS approach. For the ICP approach, it ends after positioning the robot directly in front of the conveyor belt or MPS slide before starting the ICP algorithm. The alignment phase ends for the VS approach if the body and gripper target poses are reached. For the ICP approach, this phase ends after ICP matches its position with the conveyor belt and is aligned with it. In the routine phase, both approaches execute their routines in an open-loop control, manipulate the workpiece and drive 10 cm back. Afterwards, the routine and total time is stopped and the action was executed successfully.

The ICP approach had issues with the lasers mounted at the output side of each MPS, shown in Figure 24, because the gripper fingers are sometimes placed behind the laser before executing the routine. This resulted in the gripper dropping workpieces and breaking one of its fingers. These lasers were not mounted on the machines for the



Figure 24: Image of a conveyor belt at the output side of an MPS with a laser mounted at its left and a workpiece on top.

RoboCup Asia Pacific 2021, which is why this caused no problems before. In offline RCLL events, these lasers are mounted on the machines, but for the evaluation of the ICP approach, the laser were removed.

6.2.1 Positioning

In the positioning phase of the VS approach, the robot is moving to a position 50 cm in front of the target object while the object tracker is active. If the target object is detected in 3 consecutive images and the robot is within 20 cm to the positioning target position with a maximum difference of 0.1 radiant in rotation, the positioning phase successfully ends.

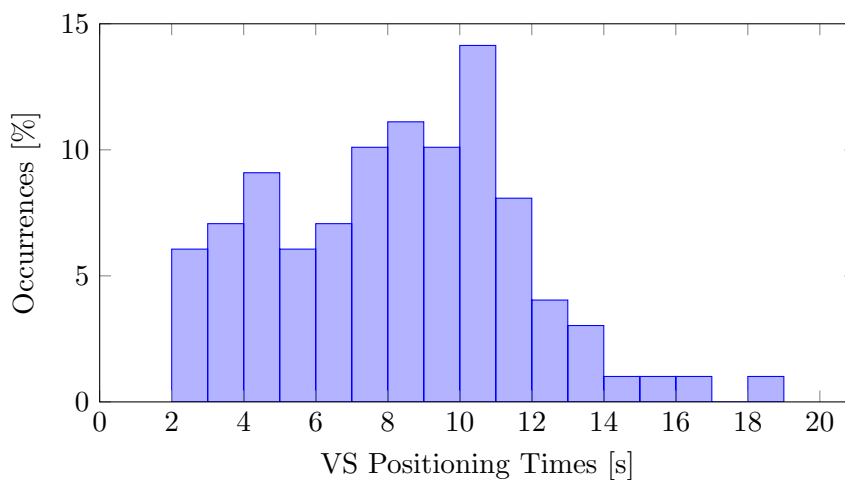


Figure 25: Positioning times of the VS approach.

In the positioning phase of the ICP approach, the robot moves to a point 75 cm in front of the target conveyor belt or MPS slide, searches the machine, and approaches it until the robot is positioned right in front of its target.

Figure 25 shows a bar chart of the VS approach’s positioning time. The time deviates a lot because of the different starting poses and target locations for each action with a standard deviation of 3.4s and a mean of 8s. This is the biggest factor in the deviation of the total times for the VS approach.

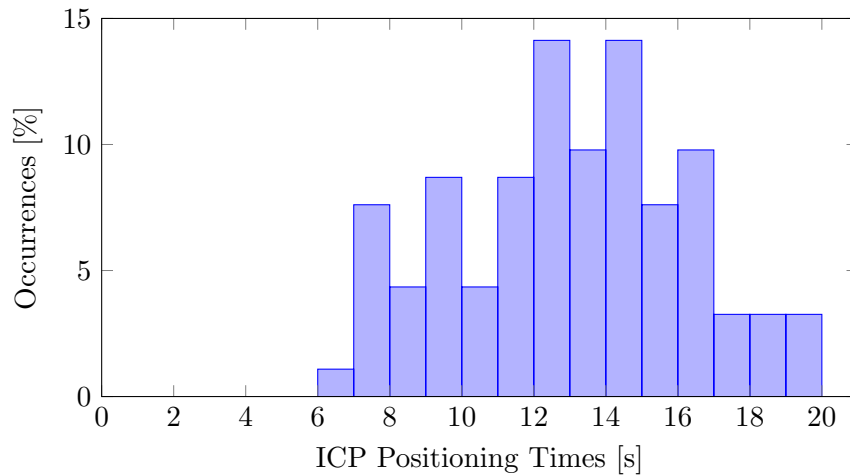


Figure 26: Positioning times of the ICP approach.

The times for this phase are plotted in Figure 26, which are slower on average than the times for the VS approach with a mean of 13s, which is 5s slower. This is caused by the VS approach’s positioning phase ending at an earlier point in time and further away from its target. The ICP approach’s positioning time’s standard deviation is 3.2s, which is smaller than the deviation of the VS approach. This might be caused by the VS approach’s more flexible approach to positioning which allows to end at different times depending on the object tracking.

6.2.2 Alignment

In the alignment phase, the proposed approach uses VS to move the robot to the target body pose directly in front of the target object, which is either a workpiece, conveyor belt, or MPS slide, while placing the gripper at the target gripper pose.

Figure 27 shows the alignment times of the VS approach with a mean value of 4.8s and a small standard deviation of 0.7s. The deviation was mainly caused by the distance between the target position and the position when starting the alignment phase, because of the high acceptance tolerance when transitioning from the positioning to the alignment

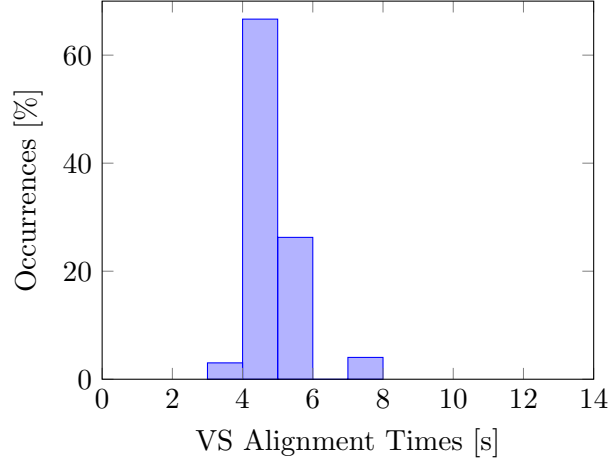


Figure 27: Alignment times of the VS approach.

phase. Additionally, the gripper usually needs to make a few adjustments near its target position until it is aligned.

Metric	<i>mean</i>	<i>std</i>	<i>min</i>	<i>max</i>
workpiece	4.7 s	0.6 s	4 s	7 s
conveyor belt	4.9 s	0.7 s	4 s	7.1 s
MPS slide	5.2 s	1.3 s	3.9 s	7.7 s

Table 2: VS alignment stats for different targets

Table 2 shows the means, standard deviations, minimum, and maximum times for each target object individually. The mean time and standard deviation is the lowest for workpieces, increases for conveyor belts, and is the highest for MPS slides. Especially, the standard deviation for MPS slides is much higher than for the other objects.

This is expected considering that less accurate bounding boxes result in more adjustments and a similar trend was seen in the object detection accuracy in the evaluation of YOLOv4-tiny in Table 1.

After changing the camera position to the final position shown in Figure 18c and before retraining, this effect was very prominent. The gripper often oscillated between two points where YOLOv4-tiny’s bounding boxes were too wide if the camera was near the target, resulting in a projected object position too close to the robot, and too thin if the camera reached the new target position. The gripper’s target position was then placed too far behind the object, where the bounding boxes were too wide and the whole process is repeated. In extreme cases, this oscillation continued until the execution was stopped manually after 30 to 60 s without progress.

After the ICP approach finishes its positioning phase, the robot is placed right before the conveyor belt or MPS slide. In the alignment phase, the robot is only moving slightly until it is aligned very precisely with the target.

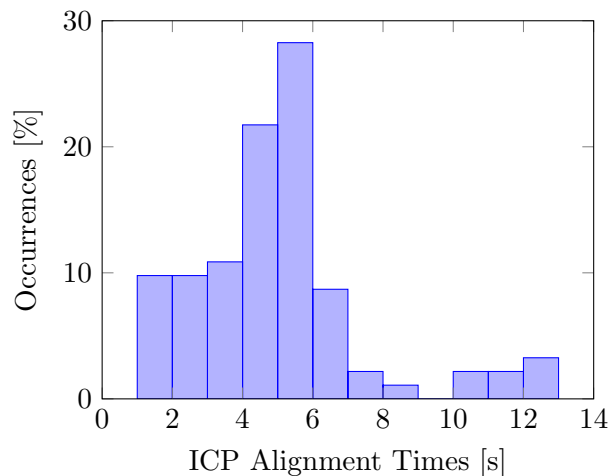


Figure 28: Alignment times of the ICP approach.

The bar chart in Figure 28 shows the resulting times of ICP’s alignment phase, which shows a higher deviation in comparison to the VS approach at a mean time of 5 s and a standard deviation of 2.5 s. The approach retries the ICP algorithm up to 2 times, but if the fitness is not high enough in any attempt, the actions fails. The high deviation in this phase’s times is caused by the amount of retries and the computation time for each run.

6.2.3 Routine

When executing this thesis routine, the main body is driving 10 cm backwards after the manipulation. Despite that, the robot is staying at target base pose and is only moving the gripper using open-loop control.

When picking up a workpiece, the last detected weighted average of the object pose is used as the target pose when moving down the gripper. This is the only adaptation any of this thesis routines does. Since the camera is not able to see the workpiece’s rings when picking it up or when placing it, it is assumed to be of maximum height with 3 rings and a cap to ensure the gripper is not colliding with it. This can be solved by keeping track of the targeted workpiece on agent level or by measuring its height while it is still in view, but since optimizing the gripper routine is not the focus of this thesis, this was not done.

Adjusting the gripper target pose closer to the actual workpiece height reduces the

distance the gripper needs to move while performing the gripper routine and reduces the time spent in this phase. Similarly, when putting a workpiece down, the gripper is moved upwards until it is above the workpiece such that they cannot collide when moving back. This may also be improved in future work.

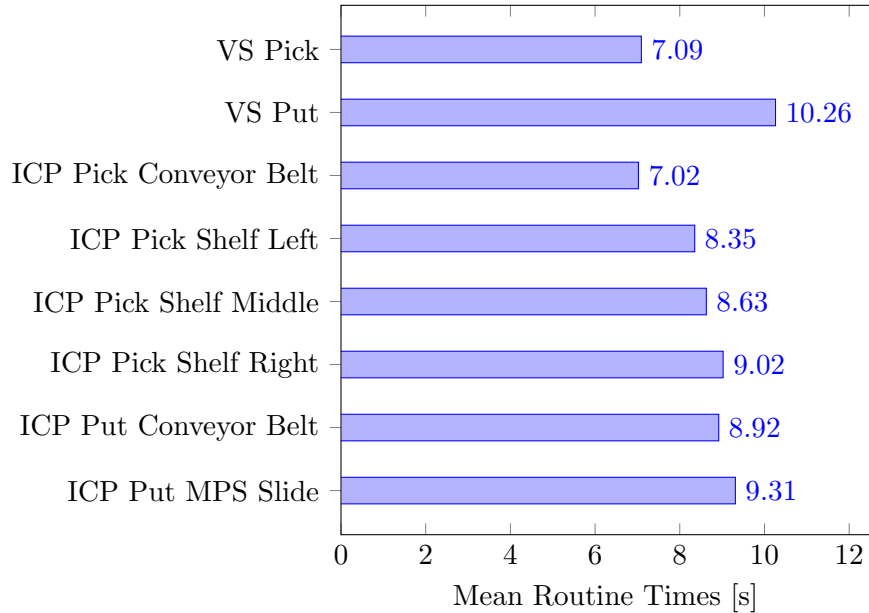


Figure 29: Mean Routine times of the VS and ICP approach.

Figure 29 shows the mean routine times for each of the approaches' routines. Both routines have a standard deviation of 0.1s, which shows there is a very low level of deviation as expected by routines.

Since evaluating both routines combined, they have a mean value of 8s and a standard deviation of 1.5s. The mean is shifted towards the picking routine's time because the amount of pick routines, manipulation targets I to VII, compared to put routines, manipulation targets VIII to X.

After the alignment phase of the ICP approach, the robot is positioned in front of the conveyor belt or MPS slide. There are 6 types of ICP routines, because when picking up a workpiece from the conveyor belt or putting a workpiece on the conveyor belt or MPS slide, the main body does not move, but when picking up a workpiece from the shelf, the body first needs to move from the conveyor belt to the position on the shelf. This results in a different picking routine for each shelf position and for the conveyor belt. Furthermore, there is a different routine for putting a workpiece on the conveyor belt and for putting one on the MPS slide. Each of these routines also shows a low level of deviation with a maximum standard deviation of 0.2s.

In contrast, there are only 2 VS routines, because every workpiece is handled equally

since we do not adjust for different workpiece heights and the body is already positioned in front of the workpiece even for shelf spots. Additionally, conveyor and MPS slide routines are executed relative to their target gripper pose and use the same routine.

Overall, the ICP approach's routine mean time is 8 s with a standard deviation of 0.9 s, which is the same mean time as for the VS approach. The VS approach's routine has the potential to be significantly faster than the ICP approach, after optimizing the routines further, since it detects workpieces and starts the routine in front of them even if they are placed on the shelf. Additionally, the routines in the ICP approach start from a gripper pose much further back, else the gripper could collide with other workpieces on the shelf while moving the main body. The routine in the VS approach starts from the target gripper pose directly above the target.

The static routines of the ICP approach can be dangerous if the environments changes slightly as seen with the lasers which the gripper could not handle. The routine for the VS approach does not need to cover a large area where the workpiece could be since it depends on the workpiece's position and is very precisely placed above it.

6.2.4 Total Time

The total time is the sum of the positioning, alignment, and routine times.

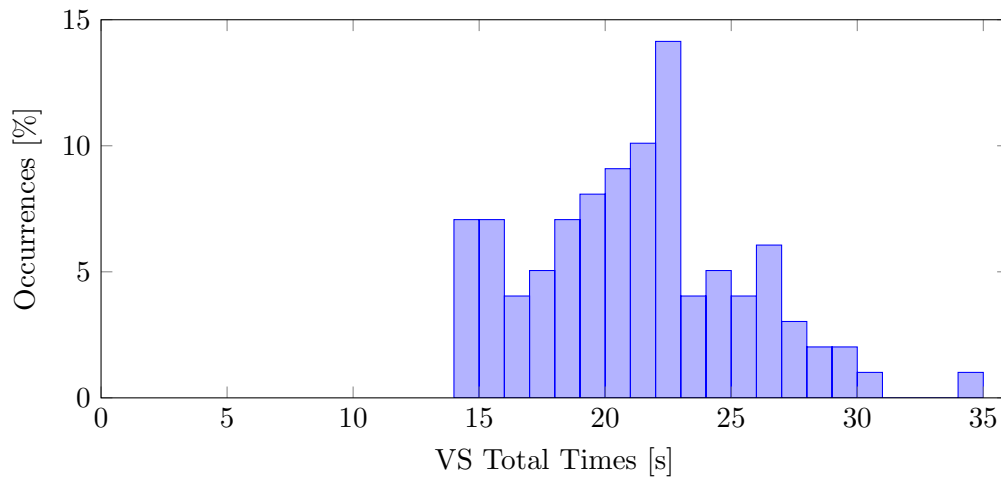


Figure 30: Total times of the VS approach.

The total times of the VS approach can be seen in Figure 30 with a mean time of 21 s, a standard deviation of 4.1 ss. The deviation was mainly caused by the positioning phase and, therefore, from the differences in the starting pose and manipulation target. Another factor is the difference in the mean times between picking and putting routines.

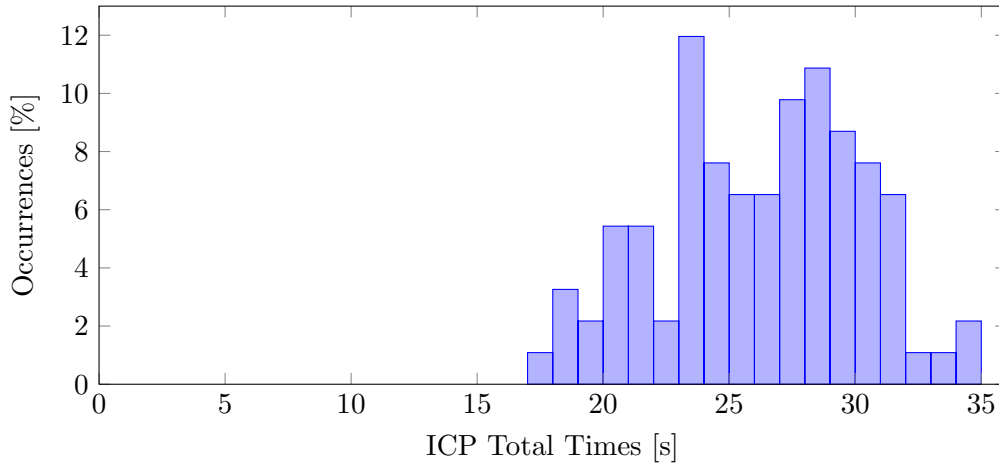


Figure 31: Total times of the ICP approach.

The total times of the ICP approach can be viewed in Figure 31. The mean total time is 26s with a standard deviation of 4s. The main deviation is mainly caused by the positioning phase and, additionally, heavily from the alignment phase. The ICP approach's mean times are worse for each phase.

The VS based approach's mean total time is about 5s faster than the VS approach, which is an 19% improvement.

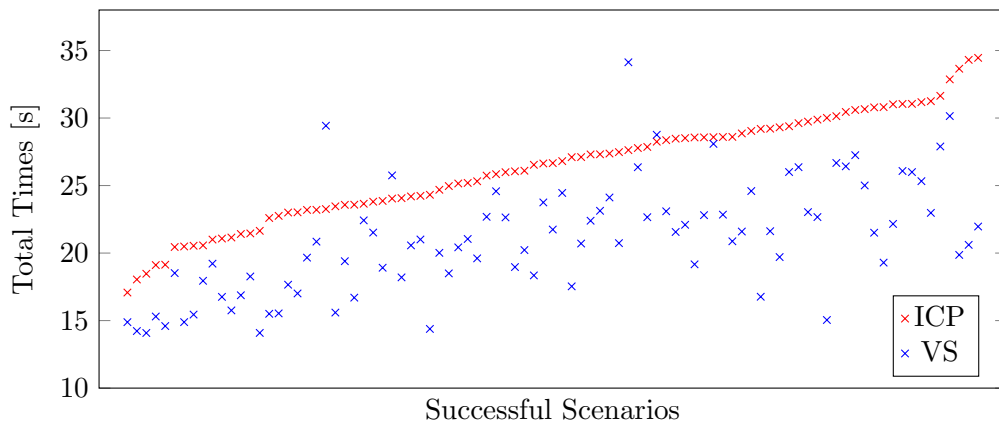


Figure 32: Total times of every attempt which was successful for both approaches ordered by ICP times colored in red and VS times colored in blue.

Figure 32 shows the total times of both approaches, the ICP approach in red and the VS approach in blue, of each scenario where both approaches were successful. The scenarios are ordered by the total times of the ICP approach. The figure shows that VS was faster in most scenarios. Only in 4 scenarios was the ICP approach faster than the VS approach.

To confirm that the VS approach is significantly faster than the ICP approach, a Welch's t-test [41] was made. The 91 total times, in which both approaches were successful, were used for the analysis. The null-hypothesis is rejected with $p < 0.0001$. This proves, that the VS approach is significantly faster than the ICP approach.

The maximum total time is an outlier which deviates a lot from the mean. Its scenario was to put a workpiece on the conveyor belt at the CS input side, manipulation target VIII, starting from point 3 as shown in Figure 22. This time was mainly caused by backwards state transitions from the "Move Body and Gripper" state to the "Search" state, which occurred because the target object was not detected for 4 consecutive images. These transitions increases this attempt's total time by about 10s. This was the only attempt where two of these appeared, which were very rare overall since only 4 backwards state transitions appeared in total while evaluating.

The other backwards state transitions were from the "Move Body and Gripper" state to the "Search" state too and resulted in about 2.5 and 5 additional seconds, but since the total times of these attempts were average despite these extra seconds, their total times did not become outliers. However, two of the three attempts were outliers in the alignment phase with times over 7s, which makes up half of the occurrences in the bar with the highest times in Figure 27. They only stand out in this phase because of its low deviation compared to the other phases.

The other outliers were mainly caused by deviations of positioning times by starting points, where the ICP approach performed better than the VS approach, deviations of alignment times when targeting the conveyor or MPS slide of the RS, and the differences in time between putting routines.

Over the set of 100 benchmark scenarios, the success rate for the VS approach is 99% and for the ICP approach 92%.

The VS approach was evaluated with the lasers, shown in Figure 24, mounted and this most likely caused the approach to drop a workpiece once, which was the only unsuccessful scenario of the approach. Before this scenario, the gripper was touching the lasers when grasping a workpiece from the conveyor belt at an MPS output side multiple times. This resulted in a miss-calibrated gripper, whose x-axis was about 1 cm too far in front. In the unsuccessful scenario, with starting pose 5 and manipulation target V, the workpiece was located at the BS's output side and the gripper was placed too far behind the workpiece, which was probably caused by the miss-calibration of the gripper. The gripper's finger were blocked by the laser and when moving the gripper upwards, the fingers were closing in such a way that the workpiece was not properly grasped and fell to the ground shortly after.

5 of ICP's 8 failed scenarios where caused by failing in the alignment phase because of too low fitness scores. In one scenario, the workpiece was not properly placed on the conveyor belt. It was placed partially on the metal rail next to the conveyor belt and

the other two scenarios failed in the positioning phase.

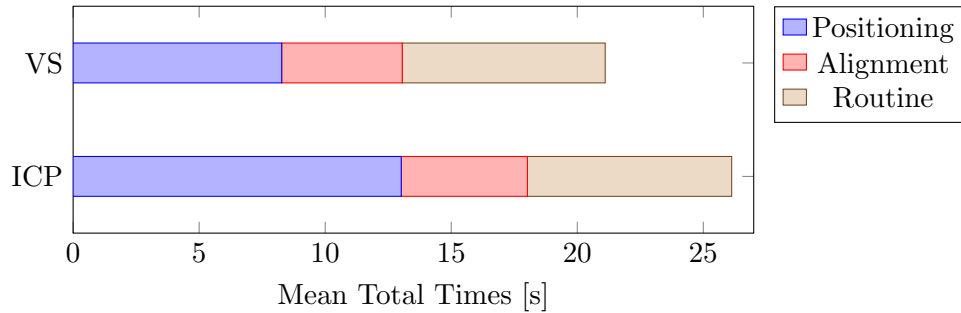


Figure 33: Mean total times of the VS and ICP approach.

The mean times of each phase of both approaches can be seen in Figure 33, where the mean positioning time is blue, mean alignment time is red, and mean routine time is brown. Even though the phases of both approaches are similar, we cannot easily compare them since their start and end robot poses differ heavily as well as their goals.

However, we can claim some conclusions. The VS approach transitions faster from positioning to alignment than the ICP approach because of its flexible nature and finishes the alignment phase in a similar time as the ICP approach finishes its positioning phase. In these transitions, the ICP approach is placed before the conveyor belt or MPS slide, starts to align itself with it and measure its relative position very precisely. In the same time on average, the VS approach had not only positioned the robot before its target, the gripper is also at its target pose. The mean alignment time for the ICP approach is close to the about 5s the approaches differ in their mean alignment time and is the mean time the VS approach is faster in total compared to the ICP approach. This can be viewed that ICP needs these approximately 5s to precisely align itself with the MPS while VS is already precisely located on its way to the target object and saves this time.

The mean routine times of both approaches are similar even though the VS approach has the advantage of being in front of the target object and having its gripper properly placed above it while the ICP approach makes this time up with more optimized routines. This indicates that the VS approach could reduce its routine time further together with its total time.

6.3 Grasping Challenge

In this section, we reproduce a challenge from the RoboCup Asia Pacific 2021 which focuses mainly on manipulation. The challenge is called the Grasping Challenge and is done on the 5×5 meter field shown in Figure 34 where a RS, CS, and BS are placed according to the figure. The MPS's orientations are clarified through their input, I, and output side, O. The robots starting positions are marked as circles. The challenge

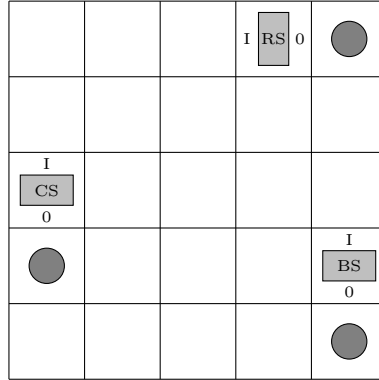


Figure 34: Starting configuration for the grasping challenge. BS is placed at M.Z12, CS1 is placed at M.Z53 and RS1 is placed at M.Z25.

starts with a robot placed in front of the output of each machine where a workpiece base is located. Each robot needs to pick the workpiece up, drive it to the input of the machine, and put it on the conveyor belt. While the robot is driving back to the output of the machine, a human supervisor is placing the workpiece again at the output. This is repeated until each workpiece was placed 3 times at the respective input and all robots returned to their starting positions.

Since only one robot was capable of VS while evaluating, the challenge was changed to be done with one robot. In the Grasping Challenge, all 3 robots are working in parallel and the time is stopped after the slowest robot finishes all actions successfully and moves to its starting position. To copy this setup with one robot as well as possible, the challenge will be done for each machine separately and the worst time is used for comparison.

When using the VS approach, the robot needed 123s for the CS, 121s for the RS, and 119s for the BS. Therefore, we evaluate the attempt with a time of 123s. For comparison, Carologistics won the Grasping Challenge with a time of 225s, which is an improvement of 45%.

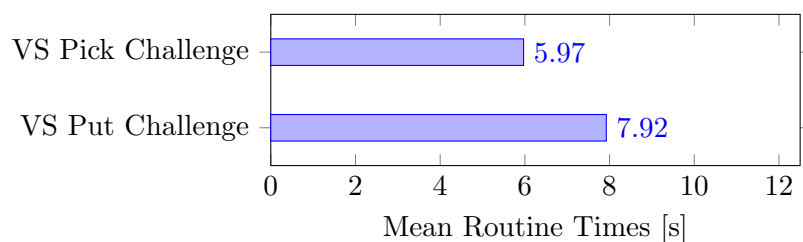


Figure 35: Mean Routine times of the VS approach for the Grasping CHallenge.

Figure 35 shows the mean time for the pick and put routine used for the Grasping Challenge, which can be compared to the mean times of the other routines in Figure

29. The mean time for the pick routine is 6 s, which is about 1 s faster than the picking routine used for the VS approach in the main evaluation. The put routine takes about 8 s, which is even 2 s faster than the putting routine of the main evaluation.

For this challenge, the routine was slightly modified to fit the height of workpieces with no rings instead of ones with 3 rings, which is safe since the challenge only uses these workpieces. This reduced the routine times significantly. The routines were additionally changed to move the robot 30 cm backwards after picking the workpiece up or putting it down instead of 10 cm to navigate around the machines more easily which slightly increased the routine times. This shows that VS can reduce the total time even further than shown in the main evaluation. These routine times have a small deviation, but the sample size is only 9 actions per routine and should be viewed with caveat.

6.4 Robustness against Environment

This thesis approach needs to be robust against various changes in its environment such as lighting changes, and mispositioned workpieces or machines.

The main evaluation was done with lasers mounted at the outputs of machines, which the VS approach could handle reasonably well without any adjustments. Additionally, while aligning the gripper to a workpiece, the workpiece can be moved continuously and the VS approach keeps track of the workpiece and adjusts the gripper to the new target position.

In the following, we will show how well this thesis approach handles machines which are elevated and how well the object tracking deals with changes in lighting. These examples are not common in an usual RoboCup environment since they are generally too extreme and YOLOv4-tiny was not trained for them.

6.4.1 MPS Placement

An MPS, which is placed a few centimeters differently than it is supposed to, is no problem for the approach as long as the object is still in view of the camera and the machine is placed within a threshold, which was 40 cm for the main evaluation.

More problematic is the differentiation between different workpieces on the shelf. Their position differ by 10 cm and small dislocations of the workpieces, CS, or the robot can cause the tracking to target the wrong workpiece. This was never the case in the evaluation since the environment was controlled and everything was perfectly placed. This problem could be fixed by using the laser sensor to detect the MPS and compute the expected target object position relative to the MPS based on this laser data.

We compared how well the ICP and the VS approach respond to an MPS which is elevated by 1, 2, and 3 cm. The input side of an RS was used for this experiment as it has a conveyor belt, and a slide and is therefore usable to pick up workpieces and put them on the conveyor belt and MPS slide.

The VS approach had no problems with a 1 cm elevation and was able to pick up workpieces and put them on the conveyor belt or MPS slide. The camera, positioned as in the main evaluation shown in Figure 18c, was not able to see workpieces or the MPS slide with an elevation of 2 or 3 cm and could, therefore, not finish the manipulation. However, it was able to see the conveyor belt and was able to put workpieces on it without problems.

When testing ICP, it matched and aligned with the conveyor belt for every height, but since its routine does not adapt to the sensed position in z direction, the routine was not completely tested. It was stopped since it would have continued as for the normal MPS height, which would have resulted in broken gripper fingers.

We were surprised by the successful matching of the ICP algorithm for every tested height, but knew about the problem with the routine. However, the algorithm can most likely be expanded to handle situations like this.

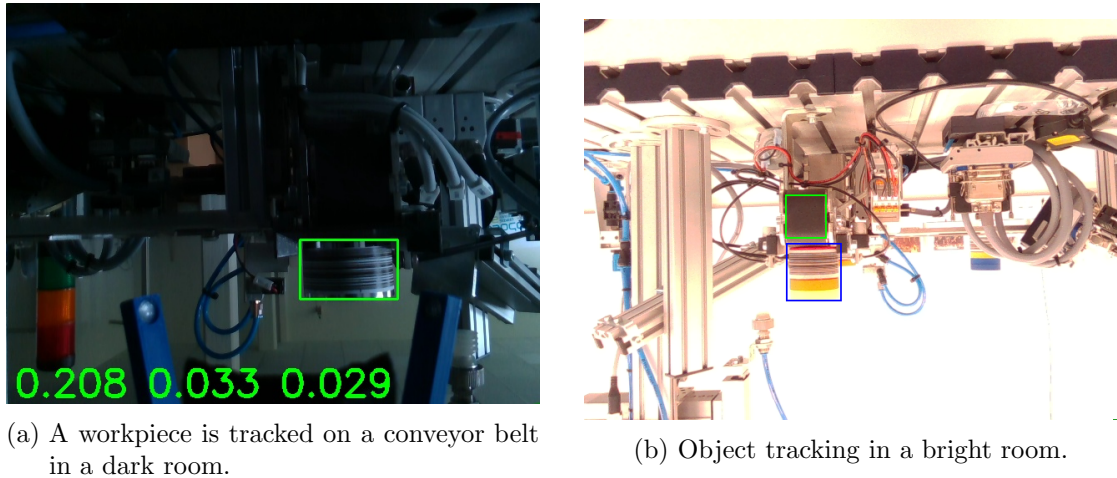
The problem with the camera of the VS approach was expected since finding a good camera position was already challenging. However, the current camera position showed some robustness by capturing all target objects with at least 1 cm of deviation in height. Putting workpieces on the conveyor worked fine and we assume, if the workpiece or MPS slide would have been in the image, the algorithm would have successfully executed the actions with them too.

This shows that the ICP approach is currently more robust against misplacements of the MPS than the VS approach. Most likely both approaches are capable to handle these situations if they are adjusted for them. The VS approach's expected position would need to be computed relative to the MPS and the camera needs to be positioned to view the target objects at all time while applying VS. Additionally, the routine of the ICP approach would need to adjust to changes in the z-axis. However, the VS approach is able to react to misplacements instantly while aligning. The ICP approach needs multiple seconds for this and is not capable to react to changes of the workpiece position since workpieces are not detected.

6.4.2 Lighting

In this section, we will show how well our YOLOv4-tiny handles lighting conditions and situations for which it was not trained for.

While the main evaluation was performed under perfect lighting conditions on a bright day, we also examined the performance under worse conditions.



(a) A workpiece is tracked on a conveyor belt in a dark room.

(b) Object tracking in a bright room.

Figure 36: Object tracking under extreme exposure levels.

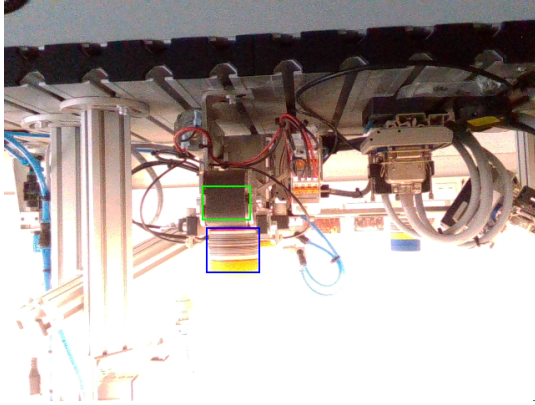
The camera used for this thesis adjusts the image to the lighting conditions well and quickly. Although, in Figure 36a, the room was very dark when the only light-source was some light from the windows, the object detector was very precise. In Figure 36b the lights over the CS were turned on while the camera was adapted to a dark room, which resulted in a bright image. However, the object tracking was very precise and detected the conveyor belt and workpiece the same as when keeping the lights on or off.

To better understand the degree the object tracking can handle and where it fails, we present two images for each situation, where the left image (a) is still handled well and the right image (b) is producing inaccurate bounding boxes or does not detect the object at all.

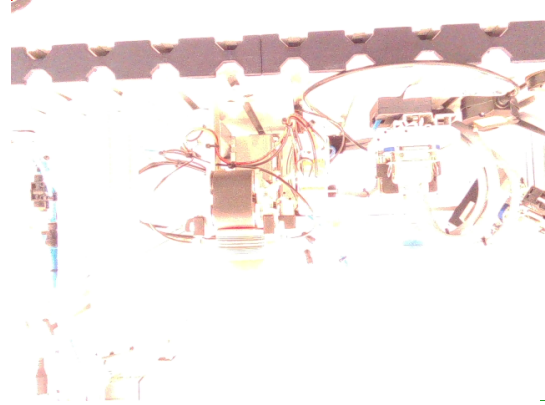
The following images are all captured from the camera position used for the main evaluation, shown in Figure 18c. Either every target object in the image is labeled or just one target object. In case every target is labeled, workpieces are marked with blue, conveyors belts with green, and MPS slides with red bounding boxes. If just one target is tracked, the image was labeled by the object tracker in real time and there is the 3D object position written on the bottom of the image.

In some images, the object position is written as "X.XXX X.XXX X.XXX" in which case the object was not detected. It is also possible that YOLOv4-tiny detected the object, but its bounding box was imprecise and rejected if its responding 3D position was not within a 40 cm threshold near the expected position.

Figure 37a shows an image of the same scene as Figure 37b, but one object tracking loop earlier. The images were captured in a dark room while the lights were turning on.



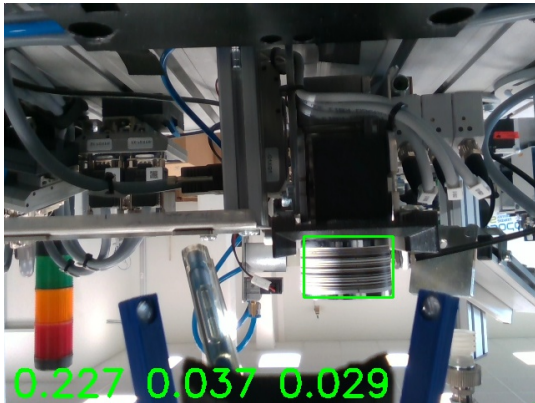
(a) A workpiece and a conveyor belt tracked in a bright room.



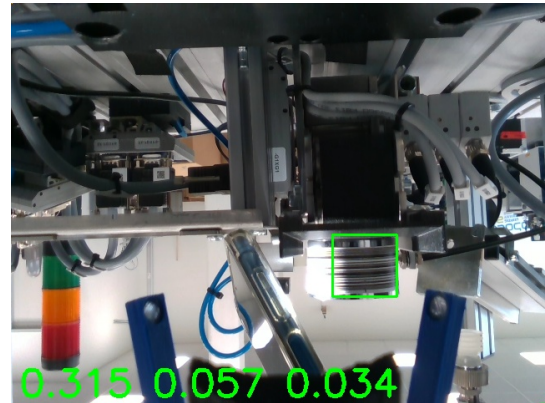
(b) A workpiece and on a conveyor belt unsuccessfully tracked in a too bright room.

Figure 37: Object tracking in bright environments.

In (a), the object tracking is still accurate and in (b), it does not detect anything. This is the only image, where the object tracking was not performing as usual, while turning the lights on. In fact, Figure 36b is the image in the following loop.



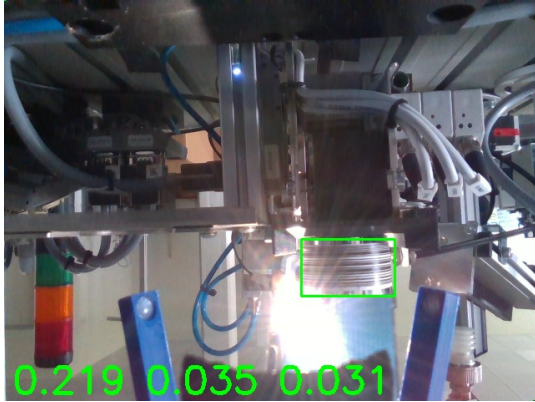
(a) Object tracking working properly with a smartphone light exposing near target object.



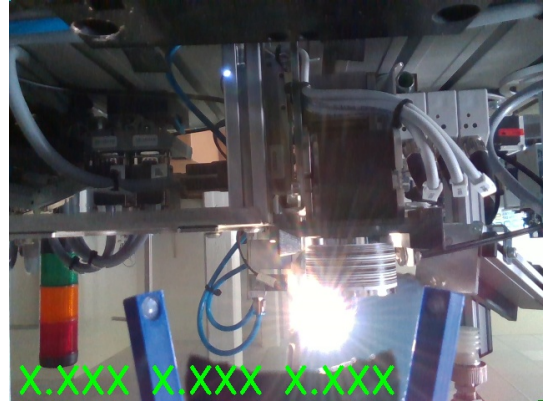
(b) Object tracker misplacing its bounding box while a smartphone light exposes it.

Figure 38: Object tracking while a smartphone light is exposing the workpiece from the side.

In the following images, a smartphone light is used to create a bright light from one direction at a close range to the workpiece. The light in Figure 38a is handled quite well and the bounding box still covers most of the workpiece. In Figure 38b a large part of the workpiece is not in the bounding box as it was most likely so bright, that YOLOv4-tiny declared it as another object or some background like a wall.



(a) Object tracking working properly with a smartphone light behind target object.



(b) Object tracking not detecting workpiece because of the smartphone light in the background.

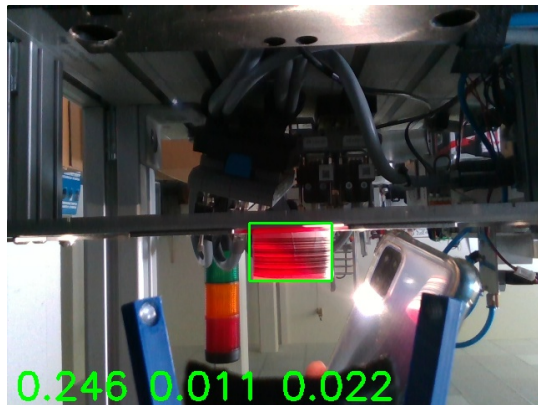
Figure 39: Object tracking while a smartphone light is exposing the workpiece from its back.

In Figure 39a and Figure 39b the smartphone light is placed behind the workpiece while lights in the room were turned off. In (a), the light looks seemingly more disturbing than in (b), but the object tracker is working fine in (a) and did not detect the workpiece in (b). The reason for that might be, that that in (b) more area of the workpiece is covered by the beam of light.

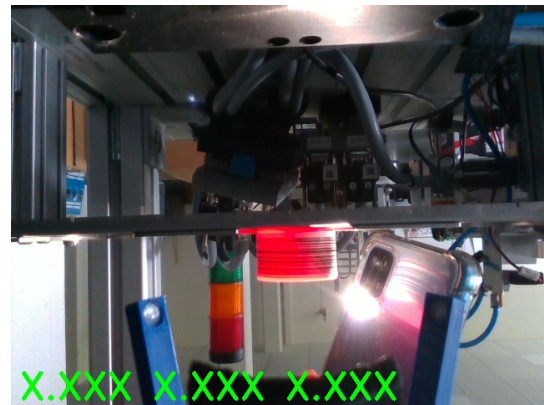
For Figure 40a and Figure 40b the scene is kept the same as in Figure 39a and 39b except for the workpiece, which was changed from a silver base to a red one. The difference in material can be seen very clearly since the workpiece is glowing red instead of reflecting the light as the silver base does.

In the final comparison, YOLOv4-tiny's robustness against human interference is tested by picking up the workpiece. In Figure 41a, the bounding box is less precise as without the finger partly blocking the vision. The image in Figure 41b was captured one object tracking loop later. In the image, the workpiece is lifted and motion blur appears. Furthermore, the workpiece was not detected.

YOLOv4-tiny has a high robustness against even extreme lighting conditions and we showed some of its limits. To improve these results, the network can be retrained, which is an important advantage of neural networks compared to traditional computer vision methods.

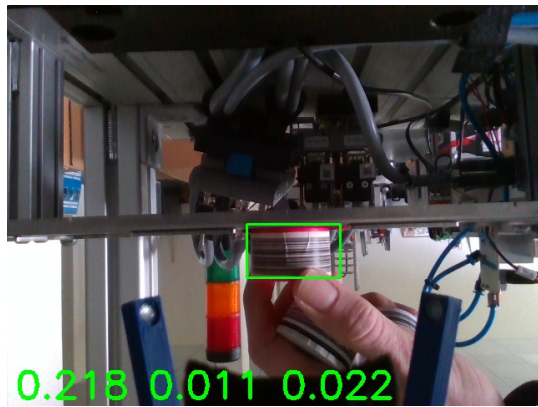


(a) Object tracking working properly with a red-glowing workpiece and a smartphone light exposing it from the side.

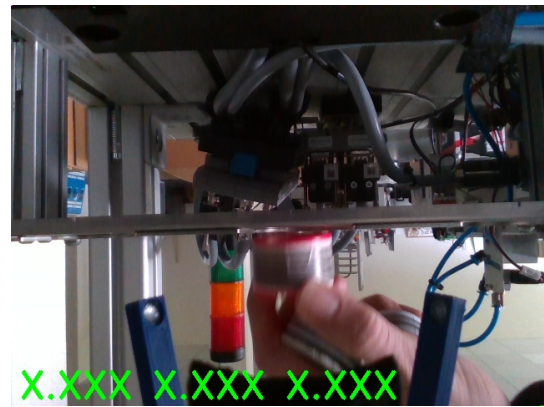


(b) Object tracking not detecting a red-glowing workpiece with a smartphone light exposing it from the side.

Figure 40: Object tracking while the smartphone light is exposing a red workpiece such that it glows.



(a) Object tracking working properly while a hand is grasping the workpiece.



(b) Object tracking is not detecting the workpiece while a hand is grasping it and motion blur occurs.

Figure 41: Object tracking while a hand is grasping the workpiece.

6.5 Future Work

In this section, we discuss possible extensions of the proposed approach that may solve some of the encountered limitations.

6.5.1 Camera

Multiple camera positions were tested and optimized for the current camera. Additionally, we discussed camera parameters, which would improve this thesis approach's performance or fix problems with certain camera positions.

Using a static camera on the main body has the advantage of more stable bounding boxes after the body reached its target pose, but a static camera needs to be placed such that it does not interfere with the gripper and is most likely to be placed further away from the target objects. However, the detection and target object position are more precise the closer the camera is to the workpiece because of calibration issues and because the area in the image with the target object is larger. However, the camera used in this thesis could not show the bottom part of the conveyor belt and the rings of the workpiece in close distances, because of its positioning and its small field-of-view for small distances. This limits the precision and functionality of the approach, e.g. by not being able to detect the workpiece height when aligning the gripper.

When using a static camera position, a camera with a higher focal length can be used. This results in a smaller field-of-view for further distances. In close distances, the field-of-view should preferably fit all target objects, which would allow to compute the workpiece height while aligning. Additionally, a higher network resolution could be used for this position, since the minimum field-of-view is bigger and the object detector needs to detect smaller target objects relative to the image space.

Additionally, the exposure time of the camera should be shorter to reduce motion blur. This should result in more precise object detection.

6.5.2 Laser Data

In this thesis, we decided to not include laser data in the object tracking to better show how well the robot moves using only images for the VS even though laser data would increase approach's robustness. However, we encourage to implement it in a future work to ensure a minimum distance between the robot and the machine by detecting the MPS using the laser sensors and regulating the speed of the robot's main body based on the distance towards the MPS such that it stops before driving into it.

Additionally, the laser data of the detected MPS can be used to increase the approach's

robustness against misplaced MPS by computing the expected object positions relative to their MPS based on the laser data.

6.5.3 Neural Network

YOLOv4-tiny turned out to perform well in terms of speed and accuracy, but the object detector could be changed to any feasible algorithm.

An implementation with 2 neural networks could be implemented in a future work as well. One is used and trained for small distances with a lower resolution to improve the computation time and performance when aligning the gripper. The other network is trained for larger distances by increasing its resolution and amount of layers while only including images with target objects with larger distances in the training set. This network is then used for the "Search" state and maybe until a certain distance for the VS. In this distance, the algorithm would switch to the network trained for small distances. A high detection rate per second is not required while the robot has a large distance to the target object and, therefore, the network for further distances could take a longer time, e.g. 200 ms, for computing the object position.

YOLOv4-tiny could be further used as the network for close distances since it proved to be capable of that.

Additionally, the image can be cropped to reduce the field-of-view to the expected target position. This needs to be done carefully, since the robot is moving and the expected position might change because of that. If parts of the target object in the image are removed, this will significantly reduce the object position accuracy. The cropping is also limited by the image resolution and will only improve the results until the certain resolution.

7 Conclusion

Interacting with objects is a common robotics problem known as object manipulation. A typical scenario is picking up an object, transporting it to a target location, and putting it down. The goal of this thesis was to make object manipulation more reliable and faster, especially for imprecise robots with precise grippers.

We propose an object manipulation framework for robots consisting of a 3-DoF omnidirectional body with a calibrated monocular camera, mounted on the body with the target object in view, and a precise gripper.

The approach is tested in the settings from RoboCup Logistics League (RCLL), which is a competition between teams of autonomous robots in which workpieces are delivered based on given orders. The robots need to pick up workpieces, transport them to certain stations, and put them down, which will be done using the proposed approach.

In this approach, workpieces as well as their destination targets, which are conveyor belts and MPS slides, are detected and tracked in 3D space using YOLOv4-tiny by assigning a bounding box around the target object and determining its 3D position using a triangulation approach without the need of stereo or RGB-D cameras, which is often required in other approaches. Target poses for the main body and the gripper are computed based on the object position and are reached simultaneously using a closed-loop PBVS task for each. The robot does not need to stop before reaching its target pose and transitions seamlessly between internal navigation and VS, which is an advantage compared to previous approaches, which would always require the robot to stop for several seconds for sensor updates. After reaching both target poses, a gripper routine is executed, which either grasps or places the workpiece.

For our evaluation, we compared the proposed approach against Carologistics' ICP approach using average success rate and average total time as main criteria. Our approach had a higher success rate and was significantly faster with an average difference of 5s. This shows, we successfully created a faster and more robust manipulation framework for Carologistic's Robotino than the ICP approach, which also reacts dynamically to changes in its environment like moving workpieces. Additionally, it only requires a monocular camera, which are usually cheaper and have less issues with reflective or black materials than RGB-D cameras.

YOLOv4-tiny not only performed well in its accuracy and speed evaluation, it is very reliable in practise, too. It was a good choice as object detector for the VS approach and can be easily expanded on if environment changes reduce its performance, since it can be trained in a few hours, which is important for use cases as the RCLL.

References

- [1] Don Joven Agravante et al. “Visual Servoing in an Optimization Framework for the Whole-Body Control of Humanoid Robots”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 608–615. DOI: 10.1109/LRA.2016.2645512.
- [2] David Austin. “Generate stepper-motor speed profiles in real time”. In: *Embedded Systems Programming* 1 (2005).
- [3] Jose Luis Blanco. “A tutorial on SE(3) transformation parameterizations and on-manifold optimization”. In: (Sept. 2010).
- [4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].
- [5] Soni Chaturvedi, Rutika N. Titre, and Neha Sondhiya. “Review of Handwritten Pattern Recognition of Digits and Special Characters Using Feed Forward Neural Network and Izhikevich Neural Model”. In: *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*. 2014, pp. 425–428. DOI: 10.1109/ICESC.2014.83.
- [6] Francois Chaumette and Seth Hutchinson. “Visual servo control. I. Basic approaches”. In: *IEEE Robotics Automation Magazine* 13.4 (2006), pp. 82–90. DOI: 10.1109/MRA.2006.250573.
- [7] Francois Chaumette and Seth Hutchinson. “Visual servo control. II. Advanced approaches [Tutorial]”. In: *IEEE Robotics Automation Magazine* 14.1 (2007), pp. 109–118. DOI: 10.1109/MRA.2007.339609.
- [8] Gianmarco Dinelli et al. “An fpga-based hardware accelerator for cnns using on-chip memories only: Design and benchmarking with intel movidius neural compute stick”. In: *International Journal of Reconfigurable Computing* 2019 (2019).
- [9] Elmar Haussmann et al. “Scalable Active Learning for Object Detection”. In: *CoRR* abs/2004.04699 (2020). arXiv: 2004.04699. URL: <https://arxiv.org/abs/2004.04699>.
- [10] Kaiming He et al. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.
- [11] Paul Henderson and Vittorio Ferrari. “End-to-end training of object class detectors for mean average precision”. In: *Asian conference on computer vision*. Springer, 2016, pp. 198–213.
- [12] Till Hofmann et al. *The Carologistics RoboCup Logistics Team 2020*. Tech. rep. Aachen, Germany, 2020, 10 Seiten. URL: <https://publications.rwth-aachen.de/record/809427>.

- [13] Till Hofmann et al. “Winning the RoboCup Logistics League with Fast Navigation, Precise Manipulation, and Robust Goal Reasoning”. In: *RoboCup 2019: Robot World Cup XXIII [Sydney, NSW, Australia, July 8, 2019]*. Ed. by Stephan K. Chalup et al. Vol. 11531. Lecture Notes in Computer Science. Springer, 2019, pp. 504–516. DOI: 10.1007/978-3-030-35699-6\41. URL: https://doi.org/10.1007/978-3-030-35699-6%5C_41.
- [14] S. Hutchinson, G.D. Hager, and P.I. Corke. “A tutorial on visual servo control”. In: *IEEE Transactions on Robotics and Automation* 12.5 (1996), pp. 651–670. DOI: 10.1109/70.538972.
- [15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].
- [16] Melinda Katona, Peter Bodnar, and László Nyúl. “Distance transform and template matching based methods for localization of barcodes and QR codes”. In: *Computer Science and Information Systems* 17 (Jan. 2019), pp. 20–20. DOI: 10.2298/CSIS181011020K.
- [17] Elise Lachat et al. “Assessment and Calibration of a RGB-D Camera (Kinect v2 Sensor) Towards a Potential Use for Close-Range 3D Modeling”. In: *Remote Sensing* 7 (Oct. 2015), pp. 13070–13097. DOI: 10.3390/rs71013070.
- [18] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.
- [19] Paul Lee et al. “On-Board Vision Using Visual-Servoing for RoboCup F-180 League Mobile Robots”. In: *RoboCup 2003: Robot Soccer World Cup VII*. Ed. by Daniel Polani et al. Vol. 3020. Lecture Notes in Computer Science. Springer, 2003, pp. 422–433. DOI: 10.1007/978-3-540-25940-4\37. URL: https://doi.org/10.1007/978-3-540-25940-4%5C_37.
- [20] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: <http://arxiv.org/abs/1405.0312>.
- [21] Jingyi Liu et al. “Garbage collection and sorting with a mobile manipulator using deep learning and whole-body control”. In: *Humanoids*. 2020.
- [22] Shu Liu et al. “Path Aggregation Network for Instance Segmentation”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8759–8768. DOI: 10.1109/CVPR.2018.00913.
- [23] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *CoRR* abs/1512.02325 (2015). arXiv: 1512.02325. URL: <http://arxiv.org/abs/1512.02325>.

- [24] Patricio Loncomilla and Javier Ruiz-del-Solar. “YoloSPoC: Recognition of Multiple Object Instances by Using Yolo-Based Proposals and Deep SPoC-Based Descriptors”. In: *RoboCup 2019: Robot World Cup XXIII [Sydney, NSW, Australia, July 8, 2019]*. Ed. by Stephan K. Chalup et al. Vol. 11531. Lecture Notes in Computer Science. Springer, 2019, pp. 154–165. DOI: 10.1007/978-3-030-35699-6_12. URL: https://doi.org/10.1007/978-3-030-35699-6%5C_12.
- [25] Alex Mitrevski et al. “”Lucy, Take the Noodle Box!”: Domestic Object Manipulation Using Movement Primitives and Whole Body Motion”. In: *RoboCup 2019: Robot World Cup XXIII [Sydney, NSW, Australia, July 8, 2019]*. Ed. by Stephan K. Chalup et al. Vol. 11531. Lecture Notes in Computer Science. Springer, 2019, pp. 189–200. DOI: 10.1007/978-3-030-35699-6_15. URL: https://doi.org/10.1007/978-3-030-35699-6%5C_15.
- [26] Humza Naveed. “Survey: Image Mixing and Deleting for Data Augmentation”. In: *CoRR* abs/2106.07085 (2021). arXiv: 2106.07085. URL: <https://arxiv.org/abs/2106.07085>.
- [27] Tim Niemueller, Sebastian Reuter, and Alexander Ferrein. “Fawkes for the RoboCup Logistics League”. In: vol. 9513. Feb. 2016, pp. 365–. ISBN: 978-3-319-29338-7. DOI: 10.1007/978-3-319-29339-4_31.
- [28] Tim Niemueller et al. “RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed”. In: *Automation, Communication and Cybernetics in Science and Engineering 2015/2016*. Ed. by Sabina Jeschke et al. Cham: Springer International Publishing, 2016, pp. 605–618. ISBN: 978-3-319-42620-4. DOI: 10.1007/978-3-319-42620-4_45. URL: https://doi.org/10.1007/978-3-319-42620-4_45.
- [29] Abhishek Padalkar et al. “b-it-bots: Our Approach for Autonomous Robotics in Industrial Environments”. In: *RoboCup 2019: Robot World Cup XXIII [Sydney, NSW, Australia, July 8, 2019]*. Ed. by Stephan K. Chalup et al. Vol. 11531. Lecture Notes in Computer Science. Springer, 2019, pp. 591–602. DOI: 10.1007/978-3-030-35699-6_48. URL: https://doi.org/10.1007/978-3-030-35699-6%5C_48.
- [30] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. “A Survey on Performance Metrics for Object-Detection Algorithms”. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.
- [31] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. <http://pjreddie.com/darknet/>. 2013–2016.
- [32] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.
- [33] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

- [34] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.
- [35] Hamid Rezatofighi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.
- [36] Yutaka Sasaki. “The truth of the F-measure”. In: *Teach Tutor Mater* (Jan. 2007).
- [37] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.
- [38] Thomas Ulz, Jakob Ludwiger, and Gerald Steinbauer. “A Robust and Flexible System Architecture for Facing the RoboCup Logistics League Challenge”. In: *RoboCup 2018: Robot World Cup XXII [Montreal, QC, Canada, June 18-22, 2018]*. Ed. by Dirk Holz et al. Vol. 11374. Lecture Notes in Computer Science. Springer, 2018, pp. 488–499. DOI: 10.1007/978-3-030-27544-0_40. URL: [https://doi.org/10.1007/978-3-030-27544-0_40](https://doi.org/10.1007/978-3-030-27544-0%5C_40).
- [39] Chien-Yao Wang et al. “CSPNet: A New Backbone that can Enhance Learning Capability of CNN”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 1571–1580. DOI: 10.1109/CVPRW50498.2020.00203.
- [40] L. Weiss, A. Sanderson, and C. Neuman. “Dynamic sensor-based control of robots with visual feedback”. In: *IEEE Journal on Robotics and Automation* 3.5 (1987), pp. 404–417. DOI: 10.1109/JRA.1987.1087115.
- [41] B. L. Welch. “The Generalization of ‘Student’s’ Problem when Several Different Population Variances are Involved”. In: *Biometrika* 34.1/2 (1947), pp. 28–35. ISSN: 00063444. URL: <http://www.jstor.org/stable/2332510> (visited on 04/28/2022).
- [42] Rikiya Yamashita et al. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4 (Aug. 1, 2018), pp. 611–629. ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. URL: <https://doi.org/10.1007/s13244-018-0639-9>.
- [43] Franziska Zacharias et al. “Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories”. In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. 2009, pp. 55–61. DOI: 10.1109/ICHR.2009.5379601.
- [44] Alessandro Zamberletti, Ignazio Gallo, and Simone Albertini. “Robust Angle Invariant 1D Barcode Detection”. In: *2013 2nd IAPR Asian Conference on Pattern Recognition*. 2013, pp. 160–164. DOI: 10.1109/ACPR.2013.17.

- [45] Zhong-Qiu Zhao et al. “Object Detection With Deep Learning: A Review”. In: *IEEE Transactions on Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.
- [46] Frederik Zwillig, Tim Niemueller, and Gerhard Lakemeyer. “Simulation for the RoboCup Logistics League with Real-World Environment Agency and Multi-level Abstraction”. In: *RoboCup 2014: Robot World Cup XVIII*. Ed. by Reinaldo A. C. Bianchi et al. Cham: Springer International Publishing, 2015, pp. 220–232. ISBN: 978-3-319-18615-3.