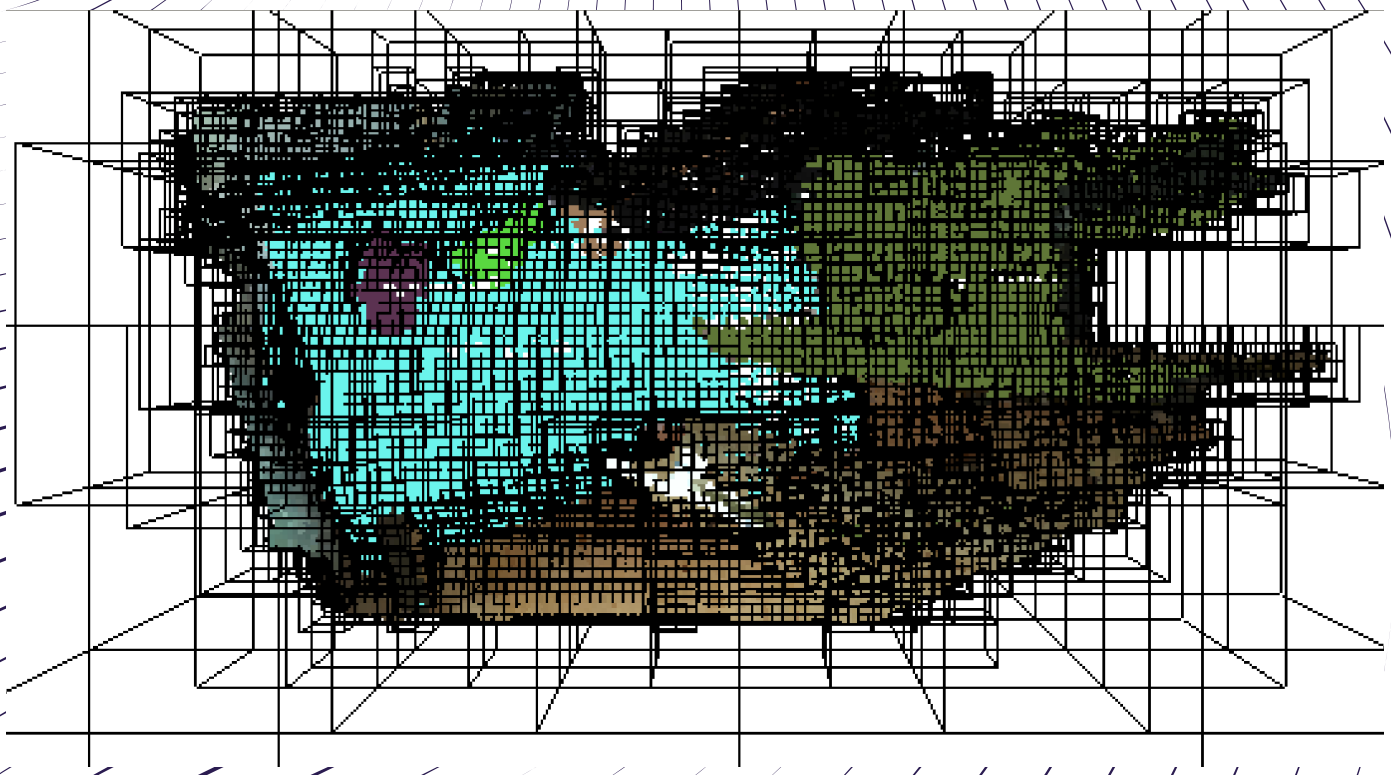
3D Volumetric and Semantic Reconstruction of a Robotic Workspace Using Deep Learning

A Proposal for a Symbiotic Human Robot Collaboration System



BREAKING NEW GROUND

AALBORG UNIVERSITY

Attributions

This report was typeset using LaTeX.

AALBORG UNIVERSITY
DENMARK

Department of Materials and Production
Department of Electronic Systems
M.Sc. Robotics

Fibigerstræde 16
9220 Aalborg Ø
https://www.mp.aau.dk
Fredrik Bajers Vej 7
9220 Aalborg Ø
https://www.es.aau.dk

Title:

3D Volumetric and Semantic Reconstruction of a Robotic Workspace Using Deep Learning

Theme:

Robotics MSc. Thesis

Project Period:

Spring 2021

Authors:

Guilherme Mateus Martins

Supervisors:

Dimitris Chrysostomou

Number of copies: 1
Number of Pages: 48
Date of completion: June 2, 2021

Abstract:

Human-Robot Interaction (HRI) has evolved at a rapid rate in the last decade. This thesis tackles the aspect of contextualising the environment surrounding the robot in the form of semantics. A scene reconstruction system is proposed, composed of an offline stage (static objects) and an online one (dynamic objects). The system works as proof of concept. Its current application is to search for small objects of interest during the online stage, utilising larger objects found during the offline stage as landmarks. The semantics of the environment are computed by utilising a two-stage object segmentation pipeline, using YOLOv4 as an object detector and DeepLabV3+ for object segmentation. Object ontologies are used to create relations between static and dynamic objects. After offline and online reconstructions have been performed, the system projects the object masks to 3D coordinates. Many qualitative and quantitative results have been reported and demonstrate the robustness of the proposed system. The proof of concept is deemed successful since the system can utilise static objects as regions of interest to detect dynamic objects based on their ontological relations and infer where specific tasks will occur.

Preface

This report is written by Guilherme Mateus Martins and contains their thesis for the Robotics Masters programme at Aalborg University.

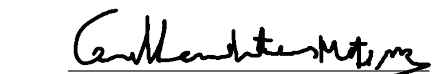
I thank my supervisor, Dimitris Chrysostomou, for his continued support during my bachelor and masters; without him, nothing of this would be possible. He made sure that the projects were always spectacular, fun, and enriching, both academically and personally.

Furthermore, I would like to thank my friends and colleagues, Daniela, Jacob, Jan, Jens, and Rune, which made my experience at Aalborg University rich, both academically and socially. Even though there were many tense moments while crunching, we always managed to have a good laugh out of the lamest things.

Also, I could not have kept sane through these last five years without the emotional support of my best friends from Denmark, Márton, Elona, Malte, and Guida. The beers we drank together will always be remembered. In addition, my best friend from Portugal, Ricardo, is one of the people most influential on my journey to Denmark; for that, thank you so much.

Moreover, I want to thank my significant other, Naima, for providing me with the emotional support I needed through and through while making us laugh even at the most difficult moments. You are the best. Thank you for putting up with my parrot Sam and me.

Most importantly, I want to thank my family for believing in me and supporting my dreams of moving 3000km away from my small home town. My grandparents, Custódia and José, which raised me as their son, and my world-class parents, Fátima and Rogério, which are always there and provide me with the love and support I need to expand my horizons, thank you, thank you... thank you. Finally, to my aunts, Ema and Bri, which raised me as a little brother and put up with so much from this (not so) little rascal, you guys mean the world to me. You are my heroes, and for that, thank you, this thesis is dedicated to you.



Guilherme Mateus Martins
<gmateu16@student.aau.dk>

Reading Guide

A Git repository for the project is made available on Bitbucket with the following link:

`https://bitbucket.org/guimateus/thesis/src/release/`

Furthermore, a demonstration of the system proposed in this thesis can be seen in the following Youtube video:

`https://www.youtube.com/watch?v=NwiqvIr0uEE`

Alternatively use the QR codes below to access the links presented above:

(a) QR code leading to the git repository.

(b) QR code leading to video demo.

Figure 1: QR codes with external links.

Finally, a list of the acronyms used throughout this report is shown below, organized alphabetically:

Acronym	**Definition**
AAU	Aalborg Universitet
AP	Average Precision
AR	Augmented Reality
AUC	Area Under Curve
CORA	Core Ontologies for Robotics and Automation
CNN	Convolutional Neural Network
CPS	Cyber-Phyiscal System
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DL	Deep Learning
FOV	Field of View
fwIoU	Frequency Weighted Intersection over Union
GUI	Graphical User Interface
GPU	Graphics Processing Unit
HMI	Human Machine Interface
HRC	Human Robot Collaboration
HRI	Human Robot Interaction
IoU	Intersection over Union
ISO	International Standard Organization
JSON	JavaScript Object Notation
mAP	Mean Average Precision
mIT	Mean Inference Time
mPA	Mean Pixel Accuracy
mIoU	Mean Intersection over Union
OWL	Web Ontology Language
PA	Pixel Accuracy
PNG	Portable Network Graphics
RAM	Random Access Memory
RGB	Red Green Blue
ROI	Region Of Interest
ROS	Robot Operating System
sHRC	Symbiotic Human Robot Collaboration
SUMO	Suggested Upper Merged Ontology
TCP	Tool Center Point
VRAM	Video Random Access Memory

Table of Contents

List of Figures

List of Tables

1 Introduction

The industry paradigm has evolved through the years, to the point in which cyber-physical systems (CPS) are now encountered in real-world applications as part of *Industry 4.0.*[1; 2]

This industrial revolution brought about an unparalleled degree of collaboration between humans and robots. Previously, people and machines coexisted with the robotic manipulators kept in cages away from humans to ensure the safety of workers. The introduction of cobots following standards and safety regulations allow robots to collaborate with humans. Such regulations provide robotic arms with safety mechanisms such as touch sensors, slower speeds and accelerations than classical manipulators, and rounded shapes. [1; 2; 3]

However, the field of *human-robot interaction* (HRI) has evolved through the years. Some robots began augmenting the previously mentioned safety strategies with safety measures of higher complexity to provide more innovative safety solutions and add redundancy to their operations. In some cases, torque force sensors are implemented to perform compliance control strategies as safety mechanisms, such as impedance and admittance control.[4] In other cases, external sensors are added, such as cameras and laser scanners; the former has seen majors advancements in the last decade with the inception of *Deep Learning* (DL) and the evolution of *Convolutional Neural Networks* (CNN).[5]

DL image processing methods allow cobots to predict human behaviour using, for example, the pose of workers, their intents, or the objects they grasp. This way, the collaborative manipulators can actively interact according to the demeanour of humans to actively avoid hurting them and collaborate on the same task. The application of such methods is causing paradigm shifts in the relationship between humans and robots. [5] Furthermore, the DL developments have aided in advancements in object semantics, which contextualizes the environment for robots. Semantics provide a higher level of understanding to the robot of its environment and can be exploited to create detailed reconstructions of the environment.[6]

Such advancements in HRI, safety, and semantics allow for a closer contextualized collaboration, which is task-driven, between workers and robots. Namely, symbiotic *human-robot collaboration* (sHRC) where humans and robots collaborate while solving tasks by making the most use of their unique capabilities while interacting naturally.[3]

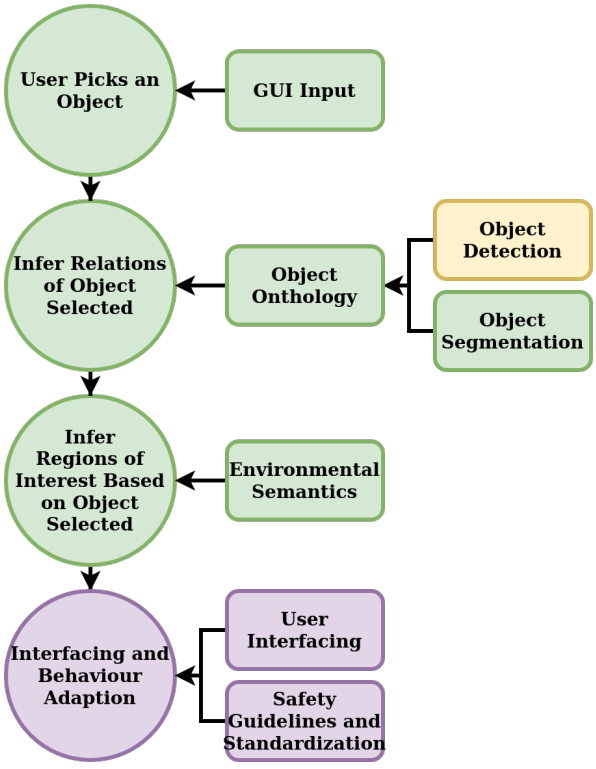
1.1 Thesis Objectives and Scope

This thesis proposes a pipeline that could be applied in a sHRC scenario. Its objective is to allow a cobot to interact and collaborate with human workers in different tasks. The text shown alongside Figure 1.1 describes the actions presented in the figure from the robot's perspective.

Circles represent a task the robot must solve in order to be able to collaborate with a human in a task. Meanwhile, the boxes are methods that can be used to solve said task. The creator of the report previously explored the yellow box. Purple boxes represent elements included as part of the literature review but are not addressed in this thesis. Finally, green boxes are explored throughout the report entirely, through a literature review and methods presentation and implementation.

The robot can observe *"Has any object been picked up by a human?"* by a multitude of inputs, such as Hand Gesture Recognition, Natural Language Processing, or Skeleton Tracking. All said methods should provide a natural input to the robot. However, as a proof of concept, it was chosen that inputs through a GUI would suffice, for reasons discussed in Section 1.2 (Scope Limitations).
When an object is chosen, the robot can infer *"What relations does this object have to its surroundings?"*, which can be achieved by associating objects to other objects or tasks through object ontologies, which is the study of objects as entities.
Then, when the properties above are known, the robot must ask itself *"Where can I find things which have relations to this object?"*. The system can use cues based on environmental semantics in order to solve this step.
Finally, the robot must interface with the human and figure out the worker's possible behaviour. The robot also must adapt its behaviour to seem non-threatening to the human and follow safety guidelines for HRC. The robot can infer *"Based on all my previous knowledge, will the human approach me? Alternatively, will they walk away?"*, or *"Should I cue the human, so they know it is safe to approach me?"*. For this step, a literature review on human-robot interfacing is explored, alongside some of the standardization necessary for HRC.

Figure 1.1: Tasks possibly contained in sHRC pipeline which the robot should perform in order to interact with humans.



1.2 Scope Limitations

Due to the COVID-19 global pandemic, physical interaction with a collaborative robot was not possible during the runtime of this project. Therefore, the last behaviour node is pushed into hypothetical future works. Hence, the aspects of the pipeline presented

in Figure 1.1 which are explored in this report are the GUI, object ontologies, and environmental semantics. These three elements should create a representation rich in the semantics of an environment when implemented into a physical robot. Moreover, the methodology presented in this report is meant to be used in industrial settings. However, the use case presented is presented with household items for the most part since those were available at the time.

The scope is also limited because to run this pipeline in a cyber-physical system, robot integration must be performed. Furthermore, utilizing a GUI, instead of more intuitive input methods, heavily limits the collaboration's symbiotic aspect.

1.3 Thesis Structure

This document is structured in the same shape as a scientific paper. The current Chapter (Introduction) presents an introduction to the problem at hand, and the scope of the thesis presented, followed by Chapter 2 (Related Works) where a literature review is performed, exploring the works previously performed which relate to the 3D environment reconstruction task explored in this report. Chapter 3 (System Architecture) proposes a layered system architecture, partly based on principles found during the literature review. Directly after, Chapter 4 (Methodology and Implementation) presents the methodology used and how it is implemented to form the system architecture presented. Chapter 5 (Evaluations) presents an experimental setup used to test the proposed system, which provides qualitative and quantitative data. Finally, Chapter 6 (Discussion and Conclusion) provides a short discussion of the results and possible future works to be done in the system and a conclusion to this report.

2 Related Works

This Chapter presents a literature review of different methods used to solve problems that relate to the one explored in this report. The first topic explored in Section 2.1 (Symbiotic Human Robot Collaboration) is the continuous approximation of humans and robots in their collaborations, presenting how different authors have increased safety measures and made collaboration between these two agents more intuitive. Then, the field of ontologies is explored in Section 2.2 (Object Ontology), which presents concepts on how the information gathered by the robot can be contextualized in relation to objects and other entities. Finally, the topic of semantics is explored in Section 2.2 (Object Ontology), which presents how robots are currently contextualizing their environment with semantic cues.

2.1 Symbiotic Human Robot Collaboration

HRC is motivated by optimizing tasks performed by humans and robots via their advantages to minimize each other's disadvantages.[3] This section explores literature regarding this topic while also considering standardization which provides safety guidelines.

For a large amount of the history of automation in factories, robots and humans have been kept separate by cages to ensure the safety of workers.[7] However, as the number of automated tasks keeps growing, so does their complexity. Human helpers, however, can help to circumvent this, though, in most cases of ordinary HRC, the human has to abide by some strict principles to make their work compatible with the flow of the automated tasks. sHRC aims to move away from this by creating interactions between humans and robots that feel natural, intuitive, immersive, require less knowledge about the robot's inner workings, and improve the symbiosis over time. [2; 3]

In [1], Nikolakis *et al.* propose a CPS which allows robots and humans to interact and collaborate safely in tasks. According to the authors, the proposed CPS gathers information regarding the conditions of the robot's workspace by acquiring data on the agents (humans and robots) in the environment. They utilize a robust communication system and components capable of changing and controlling the states of the CPS agents based on a hierarchy of systems. On the lowest level, world data is gathered and communicated to the higher hierarchy levels, processed and used for the cognition and control of physical interactions with humans. A similar hierarchy composed of five levels is also shown in [8] by Monostori *et al.* and in [9] by Kao *et al.* This hierarchy is shown in Figure 2.1 and is used as a guiding principle for the system proposed in this report.

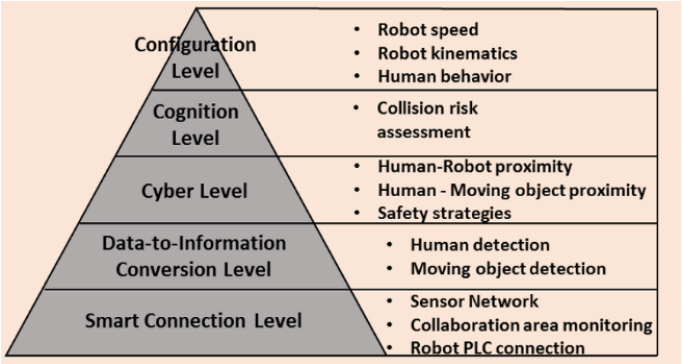
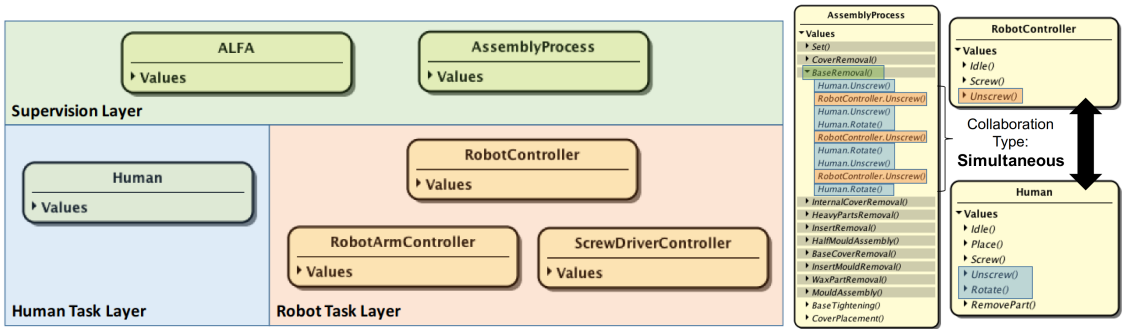


Figure 2.1: System Hierarchy proposed by Nikolakis et al in [1].

The authors of [1] use point clouds as means of detecting and tracking humans by creating convex hulls around the workers to segment them and compare the points computed to the *tool center point* (TCP) of the robot. Their safety measures are based on principles explored in [10], in which Khalid *et al.* suggested that to have safe HRC, one must use a fusion of sensors to create redundancy and comply with EN ISO standards 13849-1[11], 13849-2[12], and 13855[13]. In addition, human interactions with robots require a risk assessment to assess and reduce possible hazards for human workers, according to ISO standard 31000[14].

A different methodology is proposed in Cesta *et al.* [15], where a system is used to create a dynamic task planning framework. The system is composed of a hierarchy that supervises the actions of a human worker and adapts a manipulator's behaviour to aid the human worker to perform low-level tasks. The low-level tasks are then themselves part of a larger ecosystem of high-level tasks. The hierarchy used can be observed in Figure2.2.



(a) High level task planning. *(b) Low level task planning.*

Figure 2.2: Task planning for symbiotic human robot collaboration proposed in Cesta et al. [15].

In Liu *et al.* [16] the safety aspect of HRC is explored, where the environment surrounding the robot is contextualized using a real-time human-robot interface. This system uses a CNN for human pose estimation to avoid collisions with individuals. The sensing system of the robot is related to the robot workspace by performing a hand-eye calibration using augmented reality (AR) markers. The information gathered on the human-robot interface is used on a real-time path planner, which changes the robot's motions to assure the safety of human workers. The architecture used in this system can be seen in Figure 2.3.

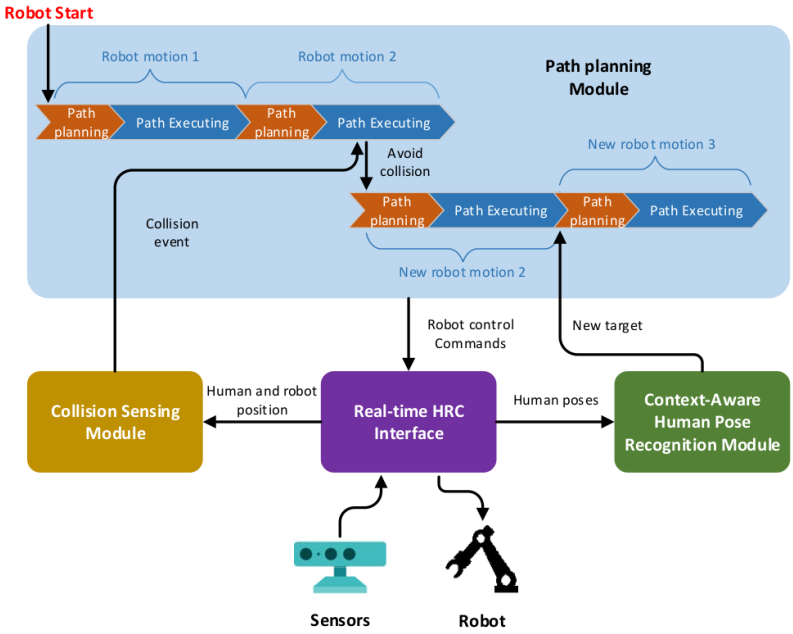


Figure 2.3: System architecture used for safe HRC, proposed in Liu et al. [16].

Advancements in sHRC allow industries that, traditionally, have not adopted automated manufacturing methods, such as the aerospace industry. In the case described by Pérez *et al.* [17], robots are being implemented alongside humans to complement each other while performing tasks. The robot is responsible for handling tasks that require high amounts of accuracy, such as placing parts in place, while the human is responsible for tasks that require a high amount of flexibility, such as screwing. Their system is composed of a control system that interacts with the inner control loop of the robot. In its turn, the control system is operated by *human machine interface* (HMI), in the shape of a tablet. The system also contains a safety system using *Lidar* and two cameras and a metrology system used to take precise measurements. The system architecture used can be seen in Figure 2.4.

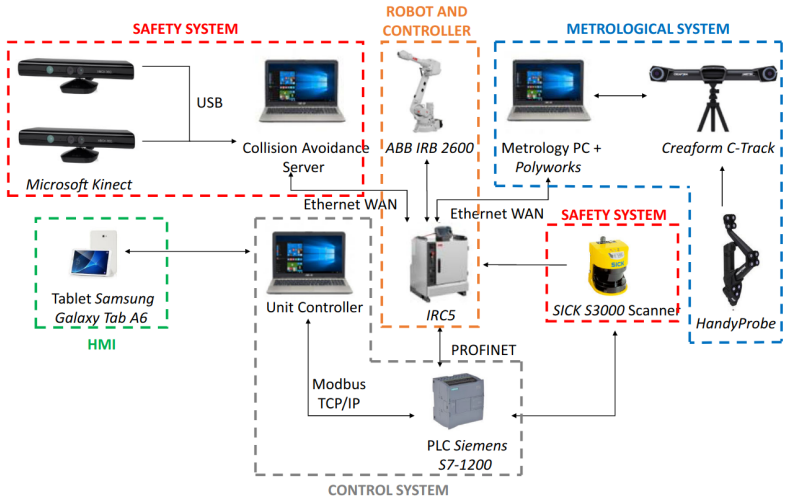


Figure 2.4: System architecture proposed by Pérez et al. [17], which is applied in a industry use case that normally would not use robots.

2.2 Object Ontology

Ontology, in general, is a philosophical field that, when applied to robotics, aims to create a bridge between human concepts and robot concepts, such that both can interact in the same scope of specific shared ideas.[18]

In [19], Jiang *et al.* make use of an ontology system for manipulation concepts. The authors manage to associate manipulation actions such as "grasp" or "hold" to specific objects such as a plastic bottle. They achieve this by creating a framework capable of reading data from camera streams, associate it with semantics, and parse them into the ontology agent using *Protégé*[20], which is an ontologies platform developed by Standford University. Moreover, *Owlready2*, a *Python* implementation of the *Web Ontology Language* (OWL), is utilized. The ontologies are used alongside a vision system in which the human executes an action, and the robot mimics the task, as seen in Figure 2.5.

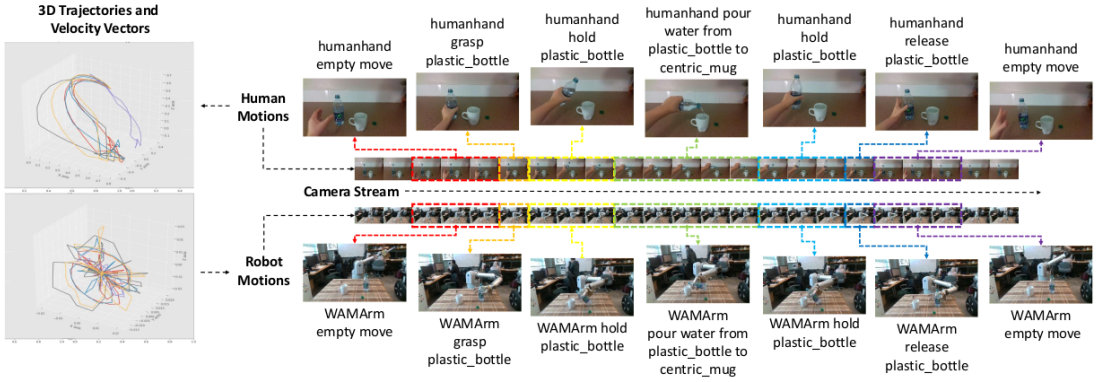


Figure 2.5: Robot mimicking human actions based on object and action ontologies, presented in [19].

In [21], Beetz *et al.* propose *RobotSherlock*, which is a computer vision framework using RGB and depth data to segment objects. The segmentation is done by utilizing the RGB feed of a camera to classify objects based on their colour and a depth feed to create hypotheses of the object positions, using *RANSAC*[22], *Euclidean cluster extraction*[23], and feature descriptors, such as *SIFT*[24] and *SHOT*[25]. *RobotSherlock* uses object ontologies (Figure 2.6) to deduct if the attributes extracted from the RGB feed match the ones expected for the hypotheses proposed using depth. Another use case for that system is to add extra attributes to an object, such as the parts which compose it, which may help robots manipulating it.

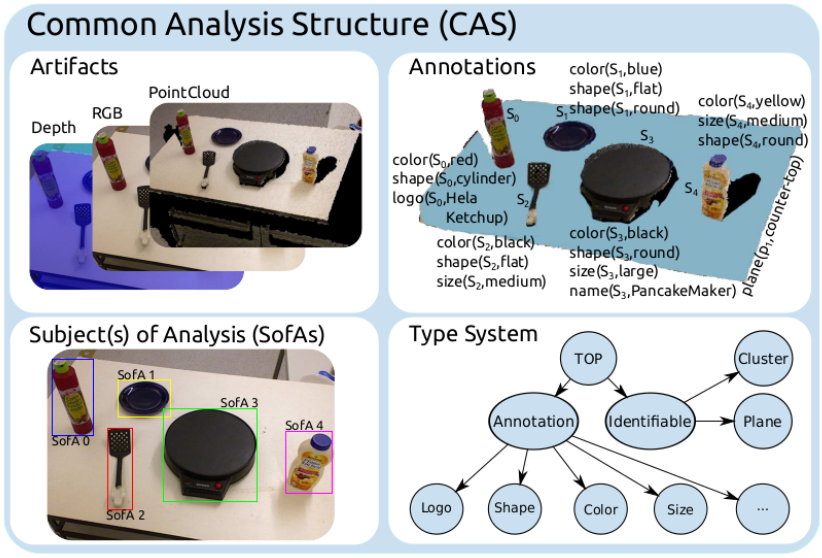


Figure 2.6: Object ontologies used by RobotSherlock in [21].

Olszewska *et al.* explore the application of ontology in autonomous robots in [26]. In the paper, two types of Ontology are used, Suggested Upper Merged Ontology (SUMO)[27] and *Core Ontologies for Robotics and Automation* (CORA)[28]. The former is used to divide entities into physical or abstract domains. In the physical domain, objects and processes are found. Objects are described as entities with spatial-temporal properties, while processes are simply temporal units. As for abstract entities, they can be represented by strings of information that are not necessarily the same but relate to the same entity. CORA expands SUMO by adding ontologies that are physical and directly interact with the tasks as tools. This is presented in Figure 2.7.

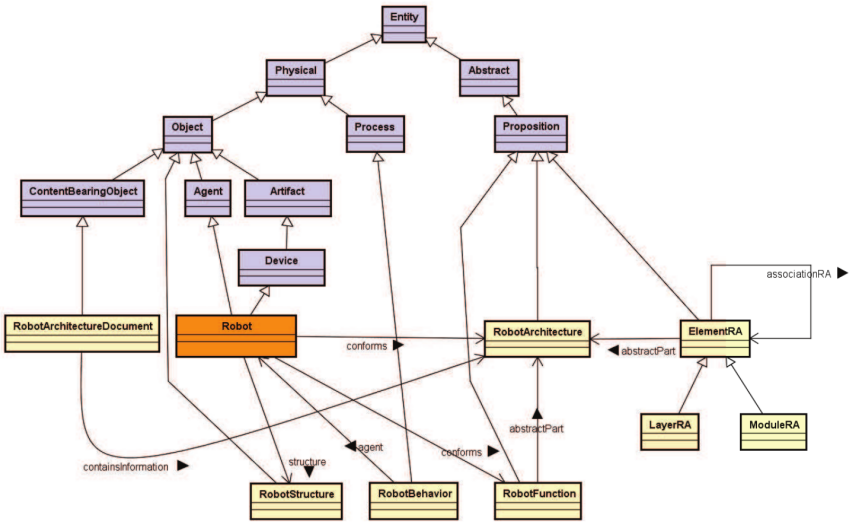


Figure 2.7: A proposal [26] of ontologies to be used in autonomous robots, where different types of entities are proposed, each with different properties. Some are physical, and others only exist in an abstract domain.

In previous work [29], Olszewska and McCluskey explored an ontology system that was responsible for creating relations between objects in dynamic environments. Using a camera to detect contours of objects which evolved over a temporal domain, the authors proposed the method *Spatio-Temporal Visual Ontology* (STVO), using a hierarchical structure proposed by Corcho *et al.* in [30]. The hierarchical structure is used to create concept trees, which expand in properties of *"things"* detected. E.g. a *"thing"* has the property *"scene"*, which in turn has the properties *"number of objects"*, *"object"*, *"spacial relations"*, and *"temporal relations"*. The deeper the tree goes, the more specific the properties become, such as *"object part"* having a specific *"part colour"*. The authors implemented their algorithm using *Protégé v4[31] and OWL*. In Li *et al.* [32], a framework to aid robots in grasping tasks is proposed, in which both object semantics and task semantics are explored. Both ontologies are implemented using OWL. However, this paper expands previous mentions of this ontology language by adding a probabilistic aspect to it using a Bayesian Network.

2.3 Environmental Reconstruction

For the most part, environmental semantics are applied in contexts of mobile robotics in the form of semantic mapping.[33; 34; 6] The principle of these mappers is to apply semantics attained by the classification of the environment itself (single cues), or by the objects present in the environment (multiple cues), into a metric map which represents the layout of spaces which a robot moves in. This report explores the possibility of applying such methods to a robotic work-cell in a manufacturing context.

The specific use case for semantics targeted in this report could be seen in a similar light as the work performed by Mantelli *et al.* [35], where a framework for semantic mapping is proposed, in which it aims to reduce the time used to search for objects in an environment. The authors achieve this by creating a metric map with semantic cues, which is divided into cells. The robot uses semantic cues to navigate specific cells using a cost function, where objects of interest may be located. The concrete example used as a use case in the paper is that the robot uses the semantic cues of doors numbers to navigate to specific rooms in the semantic map, as presented in Figure 2.8.

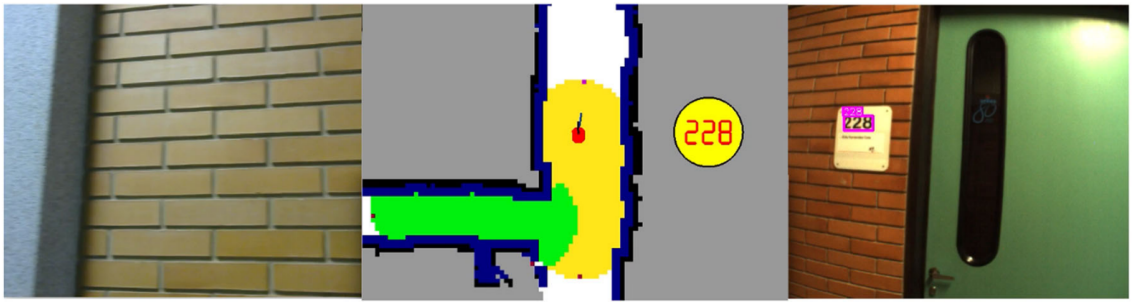


Figure 2.8: The robot in Mantelli et al. [35] gathers semantic cues based on door numbers. The cues are included in the metric map which can then be used to navigate the robot to the objective.

In some cases, the previously discussed concept of ontologies can be used to relate objects to specific places and generate semantics of environments based on those relations. Such

a concept is presented in [36] by Fernandez-Chavez *et al.* where Mask-R CNN is used for object detection, alongside odometry of the robot, OWL ontologies, and a probabilistic Bayesian framework to compute class labels for places in the robot environment. The pipeline can be seen in Figure 2.9.

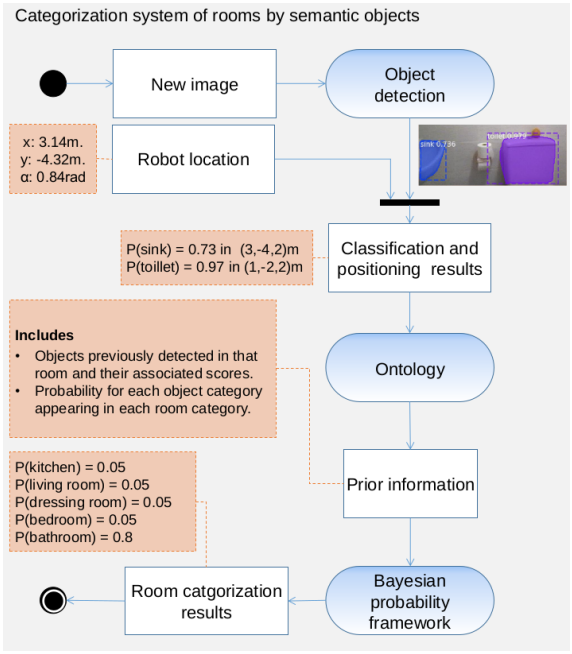


Figure 2.9: Representation of the pipeline proposed in Fernandez-Chavez et al. [36], where object relations to the environment are used to generate semantic labels for places.

In contrast, Sunderhauf *et al.* used a different strategy, also using deep learning in [6]. Instead of using object ontologies to generate environment semantics, the general context of a scene is used. The semantic labels are computed by casting the classifications of the neural network used into a point cloud of the environment, seen in Figure 2.10.

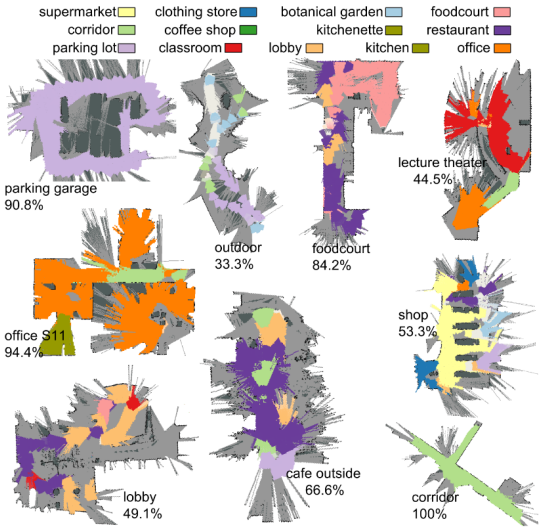


Figure 2.10: Semantics assigned to different environments in Sunderhauf et al. [6].

In [37], Grønhøj *et al.* attempted to optimize the method shown in the previous Figure by filtering the computed point cloud labels using Density-based spatial clustering of applications with noise (DBSCAN)[38] to remove outliers and optimize detections, as seen in Figure 2.11.

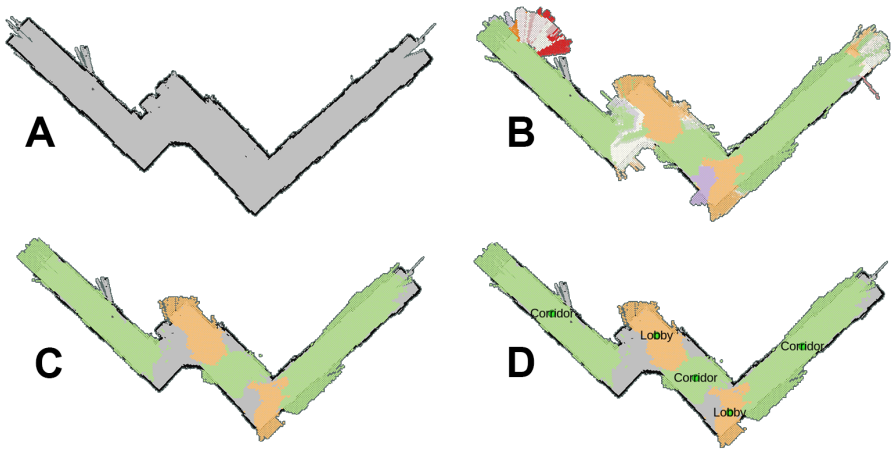
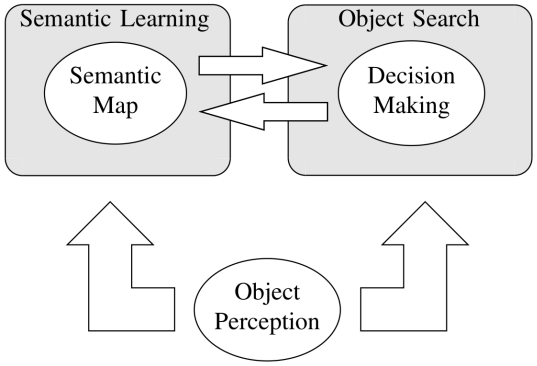
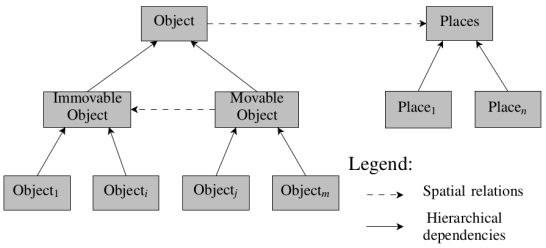


Figure 2.11: Outlier removal in Grønhøj et al. [37] using DBSCAN in the method proposed by Sunderhauf. In B, there are regions with red noise removed in C after filtering the data.

In [39], Rogers *et al.* propose the usage of a conditional random field instead of a cost function to have a robot navigating through semantics. The principle is that the probabilistic model can create relationships between rooms and objects the robot has to fetch.
Furthermore, the work [40] published by Veiga *et al.* also proposes the usage of a probabilistic framework using *Markov Decision Process* to create relations between objects and semantic categories (Figure 2.12a). In this paper, the authors used classical point cloud processing methods, creating multi-view representations of keypoints from the objects of interest to detect objects. As for the semantics, the authors created a knowledge base, which contains information regarding object relations, and a cognition engine that handles the information gathered by the object detection. The authors also make use of a structure similar to ontologies (Figure 2.12b) where some objects have immovable properties, while others can be moved.



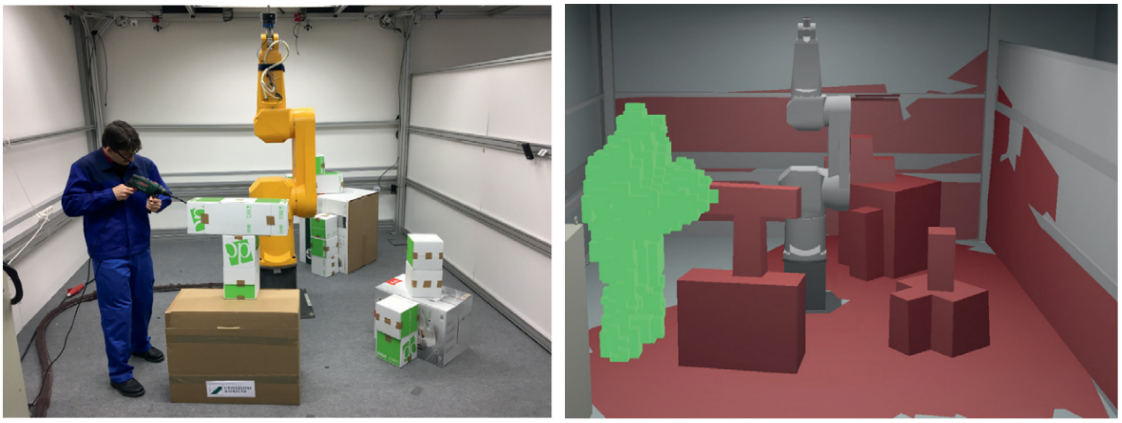
(a) Relations of the semantic map generated and objects detected.



(b) A structure similar to ontologies, where objects have properties in the sense that they are either movable or immovable. Furthermore, objects also have a relation to specific places.

Figure 2.12: Cognition of semantic map and objects detected in Veiga et al. [40].

A similar principle is applied in Werner *et al.* [41], where a volumetric map of the environment around the robot cell is generated, being partially composed of immovable entities and movable entities. The immovable entity map is generated during an offline stage, where a point cloud of objects in the environment is created. The offline map is then used alongside an online stage where an octree is generated, presenting the movable entities which must be updated in real-time. The composite 3D representation of the two stages allows industrial manipulators to plan their paths according to the environment data to avoid collisions. The resulting environment can be seen in Figure 2.13.



(a) Robot workcell physical environment. *(b) Reconstruction using point clouds and octrees.*

Figure 2.13: Physical world and 3D reconstruction based on the algorithm proposed by Werner et al. [41].

The literature review performed in this Chapter provided information about the trends in sHRC, object ontologies, and semantics. The information gathered is partially used as inspiration for the design of the architecture proposed in this thesis.

3 System Architecture

This chapter presents an architecture based on the tasks presented in Figure 1.1, back in Chapter 1 (Introduction). The design is based around the hierarchy proposed in the papers [1; 8; 9], which is presented in Figure 3.1. The purple boxes present the names of the different levels of the hierarchy. Notice that the boxes corresponding to CPS functionalities are presented in red, which means that these aspects of the system are considered out of this report's scope and should be addressed in future works.

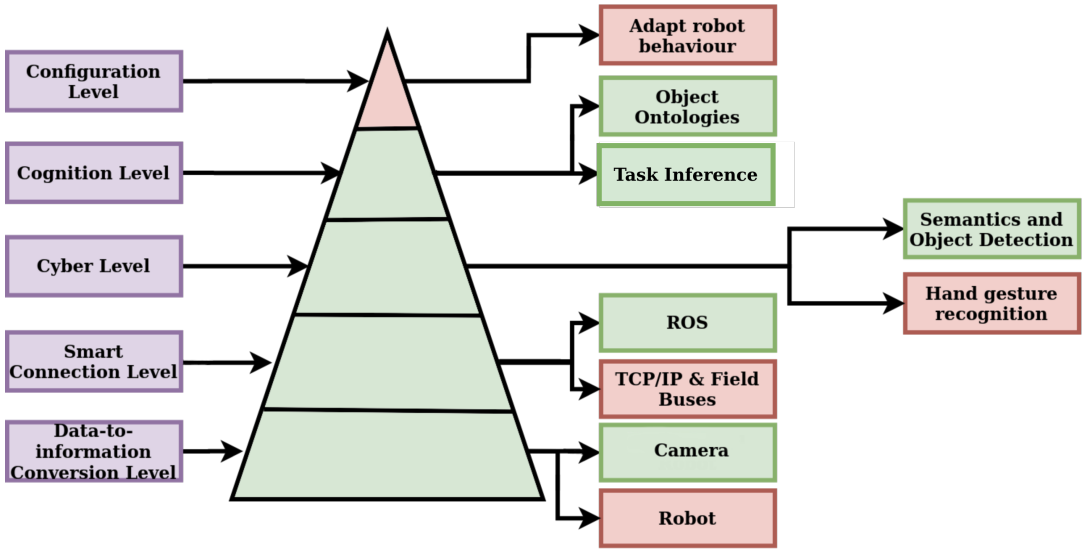


Figure 3.1: CPS system proposed, following the hierarchy design shown by Nikolakis et al., Monostori et al., and Kao et al. [1; 8; 9]. The left side shows levels of the hierarchy, and the right the aspects of the CPS proposed in this report.

3.1 Data-to-information Conversion and Smart Connection Levels

A stereo camera is used, alongside a monocular RGB camera for data gathering. The pipeline proposed in this report takes some inspiration from the work shown by Werner *et al.* [41], Sunderhauf *et al.* [6], and Mantelli *et al.* [35].

Robot Operating System (ROS) is used as middleware in the communication between the hardware and software. This framework brings the input data to higher levels of the hierarchy, where the detection and cognition algorithms are executed.

3.2 Cyber Level

The data parsed in the two previously described levels is used to run algorithms that create a 3D reconstruction of the environment. There are two separate tasks at this level that must be addressed. The first is the detection and segmentation of objects which partake in the design of the environment. The second is to create a volumetric representation of the environment.

Taking inspiration from the volumetric design proposed by Werner *et al.* [41], a two-stage 3D environment reconstruction is proposed to reconstruct the world accurately. The first stage is executed offline, where large static objects are assigned semantic labels on an RGB image, which is then stored for later usage. During the online stage, the image containing the static semantics is loaded, and a lite version of the algorithm applied during the offline stage is used. The semantics assigned during the online stage correspond to smaller dynamic objects which can be moved and manipulated in the environment while a human interacts with them.

3.3 Cognition Level

With semantics assigned to the static and dynamic objects, the usage of object ontologies is presented. This step aims to create a cognitive relationship between the environment, where the dynamic objects are found, and which tasks the objects can execute. The ontologies also allow the system to infer where precisely the task will take place in the environment.

3.4 System Architecture

Based on the levels presented above, a system architecture is proposed in Figure 3.2, composed of offline (top) and online (bottom) stages. The offline stage uses a two-stage object segmentation pipeline to compute the semantics of static objects in the environment. The masks computed by the detector are then saved in a PNG file. Meanwhile, the online stage has a similar architecture to the offline stage. The dynamic objects are detected and segmented much like the static objects. During the latter stage, the masks are cast into a 3D reconstruction node, which creates a concise representation of the environment. Furthermore, object ontologies create regions where a particular dynamic object may be found during the online stage based on their relations to specific static objects.

In the figure, blue boxes represent elements of the 2D vision, while orange ones represent 3D vision. The yellow box is the ontology node, while the purple boxes represent the communication nodes of the system.

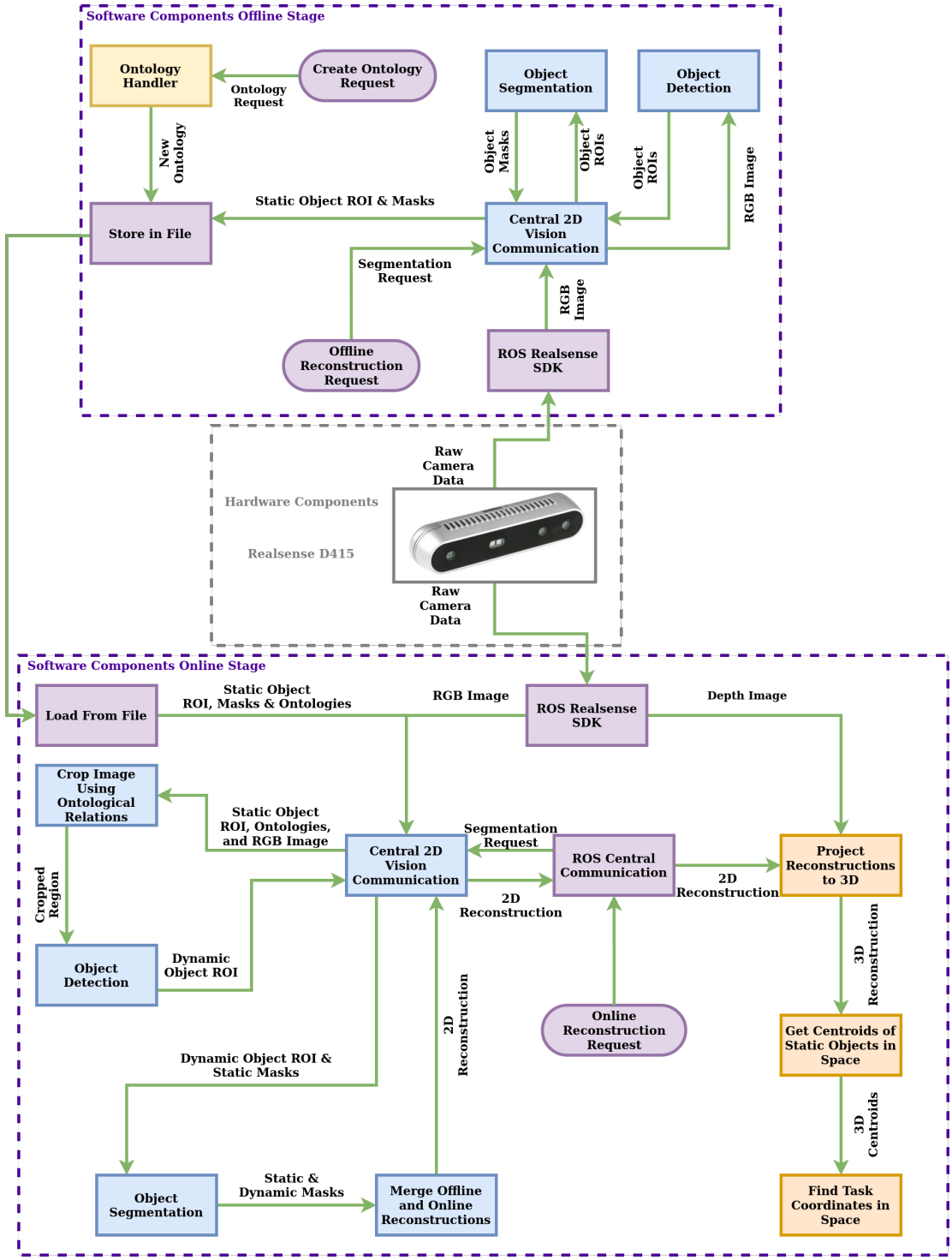


Figure 3.2: Proposed system architecture composed of offline and online stages for 3D environment reconstruction. The offline stage is responsible for the segmentation of static objects, while the online one is responsible for dynamic objects which a human can move and use to perform tasks. The system is capable of creating relations between these two types of objects by using object ontologies.

The hierarchy and architecture proposed are utilized as guidelines for the rest of the development of this project. The methods utilized are expanded on Chapter 4 (Methodology and Implementation) and tested in Chapter 5 (Evaluations).

4 Methodology and Implementation

This Chapter presents the conceptualization and implementation of the methods used in the proposed solution. Section 4.1 (A Case For Semantics) presents the thought process behind the need of contextualizing the environment through a reconstruction. The section shows the implementation of object ontologies since they provide a connection between different elements of the reconstruction, making it beneficial for a robot's cognition. Then, Section 4.2 (Volumetric Reconstruction Structures) presents different methods possible to represent the volumetric aspect of an environment reconstruction and how it is implemented in the proposed architecture. In contrast, Section 4.3 (Semantics Pipeline) demonstrates the semantics pipeline used and how it is divided into a system that handles offline and online reconstructions of the environment for static and dynamic objects. Finally, all the previously described elements are merged through a GUI, which handles their interactions and requests, which is shown in Section 4.4 (User Interfacing).

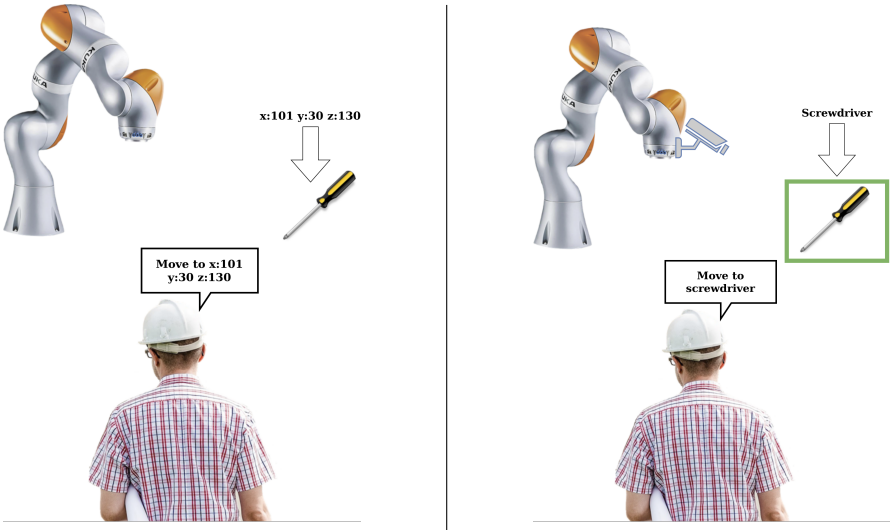
4.1 A Case For Semantics

The field of semantics in robotics has been studied in the past and in contemporary times. Robots themselves out of the box tend to be pieces of machinery with sensory inputs missing any cognition that allows them to perform intelligent decision-making in tasks. In the past, the representation of the environment surrounding a robot would be limited to volumetric data. To this day, some robots are implemented into environments with only a basic understanding of their environment, enough to generate motions and paths to complete simple tasks.

However, the relationship between humans and robots is changing and evolving, and in order to create seamless interactions between these two entities, there is a need to increase robotic systems' awareness of their surroundings in a sense that they can better interact with the world.[21] In addition, humans are incredibly reliant on semantic meanings of things and entities. We process and make sense of the world around us by assigning names to objects and concepts, facilitating interactions with other humans through linguistics. To humans, this process comes naturally, and the field of semantics in robotics aims to aid these entities to perceive the world in a grounded way, which is understandable and natural to humans. This connection can also allow workers with less technological experience and robotics knowledge to work alongside these machines.[35]

An example is presented in Figure4.1a, where a worker gets a robot to move to an object via coordinates, which can be highly unintuitive. Controlling a robot by coordinates requires a certain extent of knowledge to understand how the robot processes information

of the environment to give it the correct goals. In contrast, in Figure4.1b, the worker only needs to have an understanding of the semantics of an object to lead the robot to it using information gathered by a camera mounted on it.



(a) Case where a work must request a robot to move to an object using a three dimensional Cartesian system.

(b) Case where the robot contains a camera and understands basic object semantics to move to an object based on a user's request.

Figure 4.1: Two examples presenting an interaction using 3D data and another one using object semantics.

However, simply assigning semantics to objects via object detection is quite limited in the sense of seamless interaction. The manipulator must always be facing the direction where objects requested by the user can be found because cameras are capable of only gathering data within their field view. Alternatively, the entire workspace of the robot must be searched for the correct object, as seen in Figure4.2.

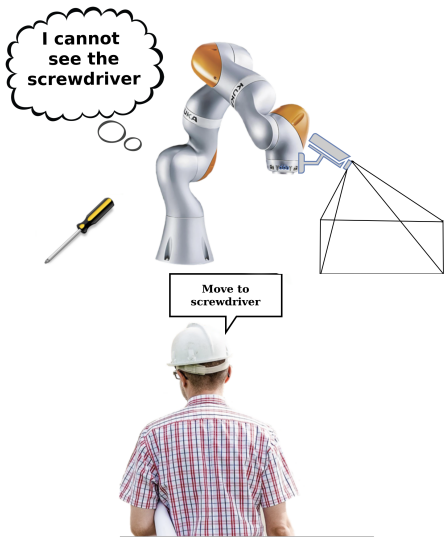


Figure 4.2: Robot facing the wrong direction of the object of interest with the field of view of the camera pointing to nothingness. To solve this the robot must either search the entire workspace, or be hardcoded to search for objects only in specific regions.

Therefore, it is proposed that by using workspace semantics, the robot should search for objects in specific regions that correspond to where the object may be found. E.g. in the hypothetical case that there are two tables in the environment, one red and one green, while the robot has the *a priori* knowledge that a screwdriver can be found on top of the red table, the robot should face said surface and look for the object of interest on that place, ignoring all the rest, as seen on Figure4.3.

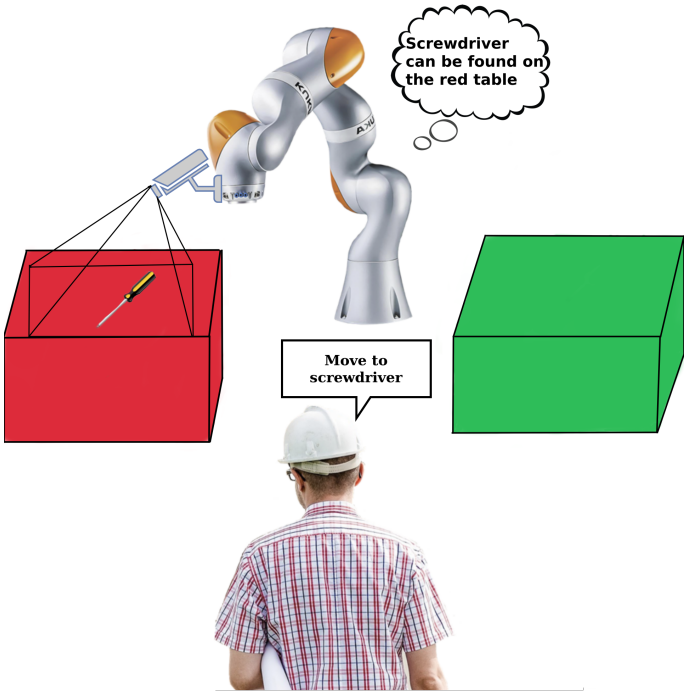


Figure 4.3: The robot is requested to move to a screwdriver, which it knows it can be found on top of a red table.

The semantic reconstruction of an environment should be performed in an offline stage and an online stage. First, the robot samples the environment for static objects (tables, chairs) and assigns labels to them, while in the second, the robot uses the static objects as landmarks to create ROIs where it searches for dynamic objects (screwdrivers, saws, pumps). However, simply assigning semantics to objects is not enough to provide the robot with an understanding of the relations between different semantic labels. Therefore, there is a need to explore relations between static objects and dynamic objects to achieve the concept presented above.

One way to explore these object relations is in the form of the previously mentioned object ontologies. An ontology can represent a thing or a concept and describes the connection between static and dynamic objects in the proposed architecture. In the case presented in Figure4.3, the dynamic object screwdriver exists within the bounds of the static object red table.

Such a relation can be visualized as it is presented in Figure4.4, where a static object has four dynamic objects related to them, and in turn, each of the four objects has properties attached to them. Properties assigned to the objects could be physical attributes or processes, such as a task in which the object is involved.

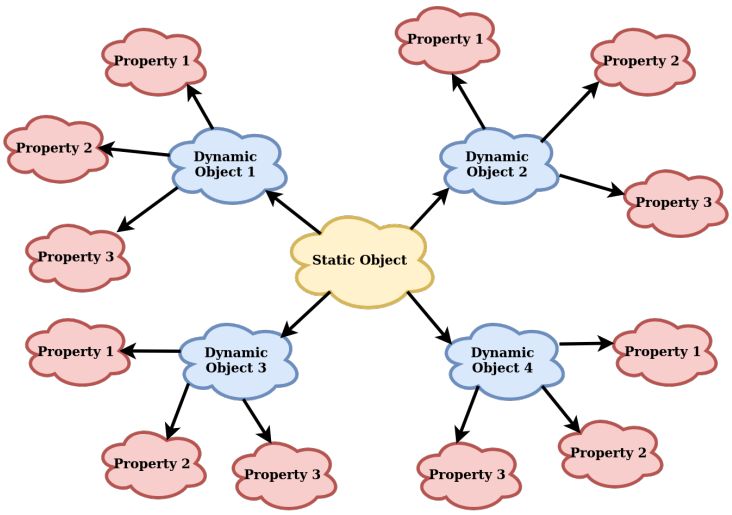


Figure 4.4: Concept of object ontologies visualized, where static objects have dynamic objects tethered to them, while they themselves have properties attached to them.

This concept is used for the implementation of ontologies in this report, which is explored below.

4.1.1 Implementing Ontologies in Reconstruction Task

In the scope covered in this report, the usage of ontologies is quite limited since it only targets the relations between static and dynamic objects and tasks in which a dynamic object can be used. Therefore, it was decided that a complex framework such as OWL would be unnecessary for the implementation. Instead, a more straightforward approach is used as proof of concept by implementing the ontologies utilizing JSON in *Python*.

An ontology handler was created to set ontological relations, based on user inputs, between static and dynamic objects, and a task that can be achieved by utilizing said dynamic objects. The ontology pipeline works by loading a JSON file into *Python* containing the structure visualized in Figure 4.5.

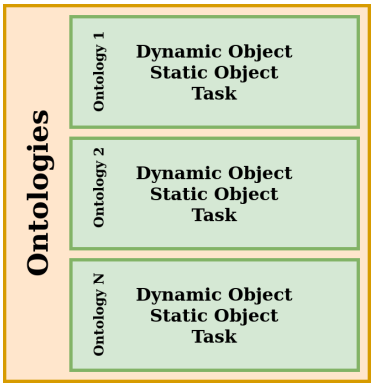


Figure 4.5: Visualization of ontology structure. The orange box represents a JSON array. The green boxes represent individual ontologies stored inside of the array. Each of the ontologies has a dynamic object, a static object, and a task.

A new relationship is created by inputting a dynamic object which had no previous relations to a static object, in which the handler appends a new element to the JSON array *Ontologies*. If the user decides to change an existing relation, the handler checks the JSON array for a known relation, replacing the existing one with the new input from the user.

The handling of new ontologies is shown in Figure4.6, in the form of a flow chart.

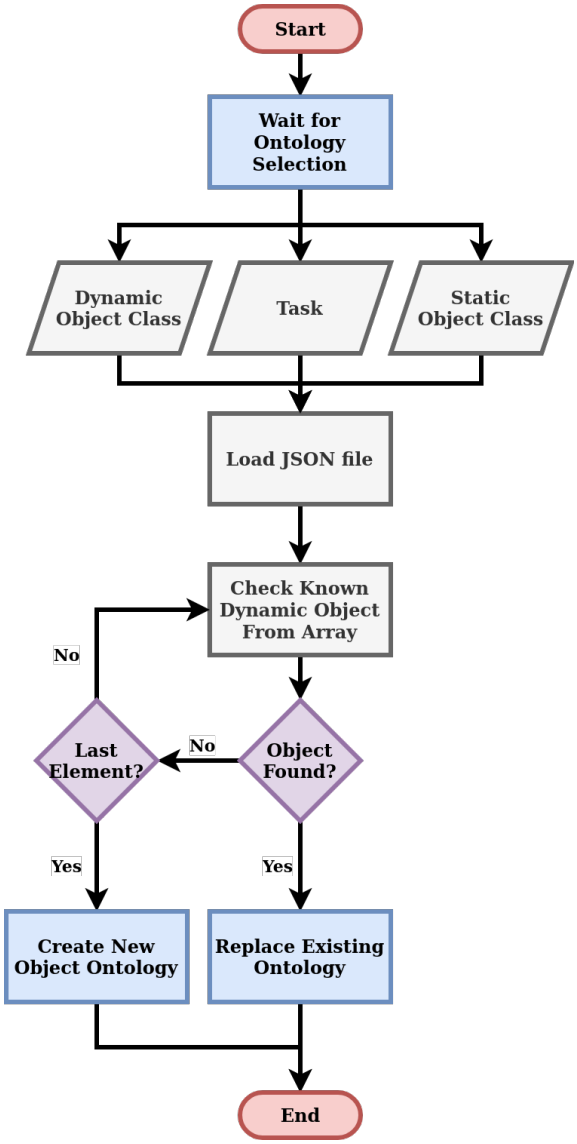


Figure 4.6: Flow chart of setting and replacing object ontologies in JSON file.

The saved file can then be loaded when there is a necessity to check relations between objects, such as when semantics are assigned to the dynamic objects, as seen in Section 4.3.2 (Implementing Pipeline in Reconstruction Task).

4.2 Volumetric Reconstruction Structures

The first step in creating a 3D semantic reconstruction of an environment is to explore the volumetric aspect since it provides the necessary foundation to distribute semantics later. RGB images are 2D representations of the Field of View (FOV) of a camera, presenting only values in height and width. In order to create a full representation of the information caught on the FOV of a camera, one must add a depth value to the width and height values. There are several methods to do this, such as sensor fusion of a range finder and an RGB camera, monocular camera depth estimation, but an RGBD stereo camera is used in this thesis.[42]

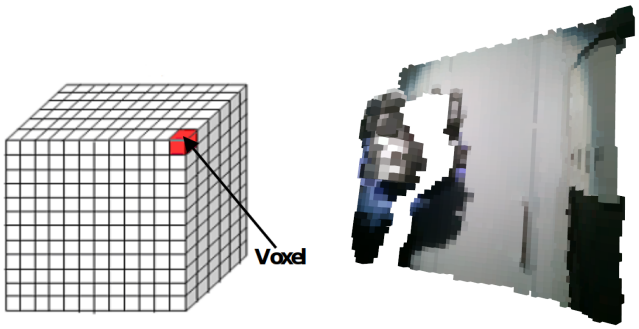
This type of sensor is composed of two cameras that are displaced from each other, allowing to calculate the disparity between the cameras' frames. The disparity can compute the distance of objects since the closer present a more considerable disparity between themselves than those further away.[42] The resulting depth image of a stereo camera can be projected into 3D data structures, such as point clouds, which can be seen in Figure 4.7.

Figure 4.7: Raw point cloud created from RGB and depth images.

Point clouds are an accurate way to represent 3D data; however, the data is not organized and non-Euclidean, since the distance between each point is ambiguous, in contrast to images in which the distance between pixels is always known. Furthermore, point clouds can contain many data points that are redundant in the representation of a scene. These redundant points must still be stored, causing an increase in the resources needed to run operations on the clouds.[43]

Since the system presented in this report aims to be lightweight, for integration in a cobot working alongside humans in a real-world setting, exploring alternatives that drain fewer resources is essential.

One of the commonly used methods to simplify point clouds is by down-scaling them using a voxel grid. This structure takes the original data and divides it into voxels, which, to some extent, are similar to image pixels in 3D, as seen in Figure 4.8b. The points contained inside of one of these cells are averaged in order to compute their centroids. Therefore, the distance between neighbouring points is the same, and the number of existing points is reduced to the number of existing voxels. A point loud down-scaled using a voxel grid can be visualized in Figure 4.8b.[43]



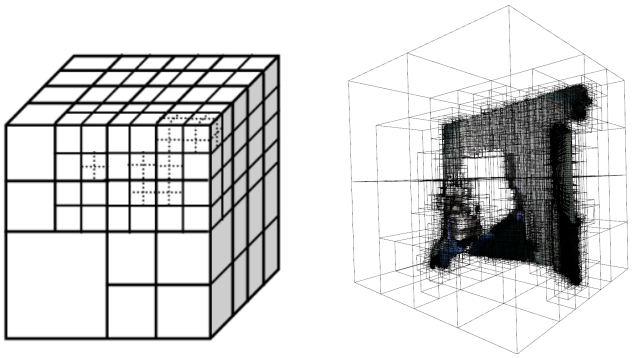
(a) Visual example on how the point cloud data is split using a voxel grid.[44]

(b) Voxel grid generated from the point cloud created from RGB and depth images.

Figure 4.8: Example of how data is split using a voxel grid and the end result on a point cloud.

Though voxel grids simplify the number of points in a point cloud, there is still much redundancy in this structure. Voxels will still represent regions without any points. A more efficient way to split the data is by using octrees.[44] This type of data structure is analogous to quadtrees used in 2D images and are a method that specifically uses three-dimensional points to split the data. The method works by splitting data alongside a central 3D point into eight child octants, which can be visualized in Figure 4.9a. Octrees provide optimized results in 3D since a single iteration of the data division into octants splits it by the x, y, and z axes.[45]

Furthermore, since this data structure is dependant on a central point to split the data, it does not split octants where no information is present, which is an advantage compared to the voxel grid's inherent problem of saving empty voxels. Another advantage of octrees is that the user can define how deep the tree should go, which aids in controlling the resources used to generate the structure. Deeper trees will show extra details; however, they consume more resources. The result of a data split using an octree can be seen in Figure 4.9b.[45]



(a) Visual example on how the point cloud data is split using a octree.[44]

(b) Octree generated from the point cloud created from RGB and depth images.

Figure 4.9: Example of how data is split using an octree and the end result on a point cloud.

4.2.1 Volumetric Mapping Method Implementation

The volumetric mapper proposed in this report takes inspiration from the one proposed by Werner *et al.* [41], which is composed of an offline and an online stage. During the offline stage, information regarding static objects is gathered in the form of an RGB frame. This information is merged in the online stage with a newly sampled RGB image containing dynamic objects placed in static objects. The implementation is performed using *Open3D*, which is a modern *Python* point cloud processing library.

The sampled data is used, alongside a depth image, to generate a point cloud, which is down-scaled into an octree since it saves precious resources for other stages of the reconstruction. In order to compute the initial point cloud, a checkerboard calibration is performed using MATLAB's *Computer Vision Toolbox*, which outputs the camera intrinsics matrix, as seen in Figure 4.10.

$$\begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 4.10: Intrinsics matrix of a camera.[46]

In this matrix f_x and f_y stand for the focal length of the x and y axis, respectively, while c_x and c_y represent the principle point of the sensor on x and y.[46]

The flow of the process to generate the volumetric representation of the environment can be seen in Figure 4.11.

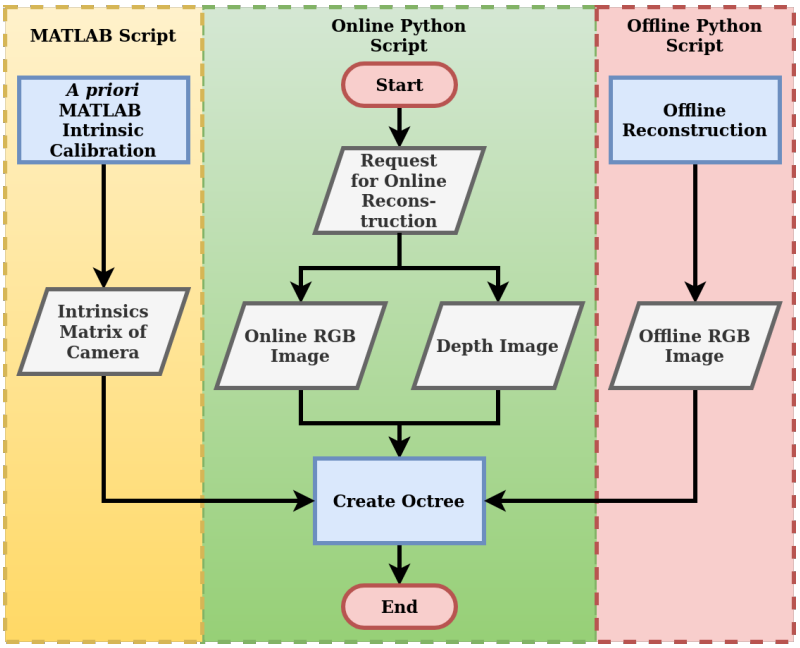


Figure 4.11: Process flow of offline volumetric reconstruction of an environment.

This implementation works as the basis to cast the semantics found in Section 4.3 (Semantics Pipeline) from 2D to 3D in order to achieve the complete reconstruction of the environment.

4.3 Semantics Pipeline

Inspired partially by the works performed by Sunderhauf *et al.* [6] and Fernandez-Chavez *et al.* [36], the semantics reconstruction in this project is performed using CNNs in order to classify the objects in the environment, both static and dynamic. For this, one must perform segmentation on objects in the environment. In the previous semester [47], the author of this report developed a two-stage detection pipeline that proposes regions of interest using an object detector, crops said regions and parses them to a semantic segmentation neural network. ***For an in-depth analysis of the methods used, refer to Chapter 5 of [47]. The methodology used in this report remained the same as the previous work, save for some improvements described below***. The principle behind this is that the detector's inference is faster than the segmentation network, optimising the time used to segment objects in the environment. In its original use case, the pipeline was implemented into a setting where screws in a diverse set of pumps had to be detected and segmented, as shown in Figure 4.12.

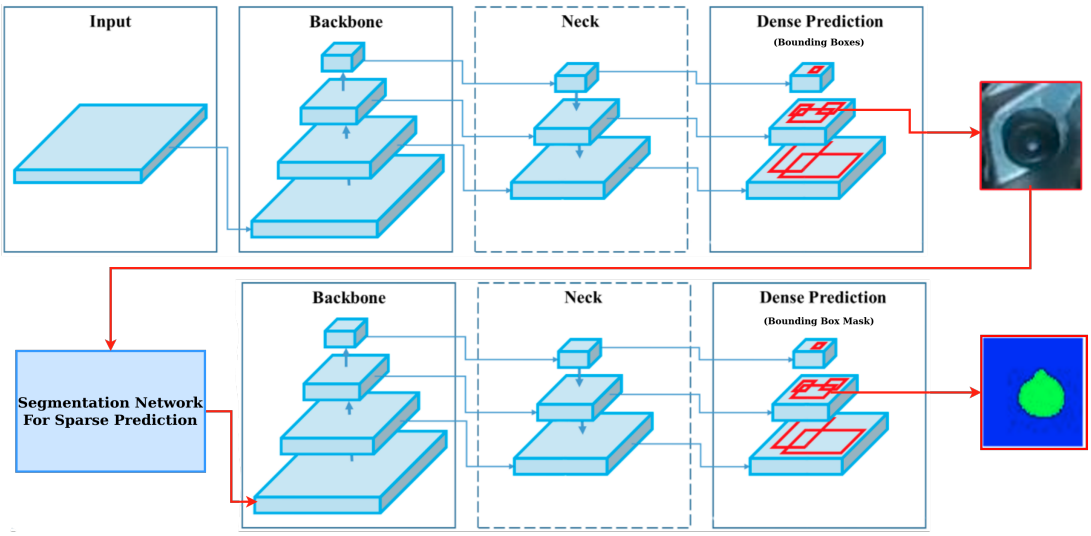


Figure 4.12: Two stage segmentation proposed on previous semester in Mateus et al. [47] for the detection and segmentation of screws. This figure is adapted from Bochkovskiy et al. [48].

The object detector used is YOLOv4 using a CSP-Darknet53 backbone proposed by Bochkovskiy *et al.* [48], which at the time the research and development of the two-stage detection pipeline took place was a state-of-the-art method in the MS COCO dataset challenge, providing a balance between speed and accuracy. It has, however, been dethroned by a faster and more precise version of YOLOv4, Scaled-YOLOv4, which can be seen in Figure 4.13. In the previous semester, there was a consideration that the accuracy of EfficientDet could prove to be beneficial while sacrificing some inference speed, however with the newest release of YOLO, it is clear that it is possible to have a more accurate detector than YOLOv4 while retaining the inference frequency.

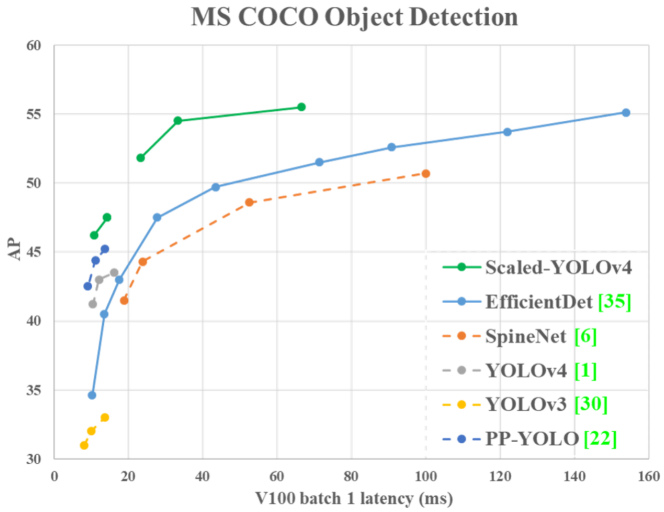


Figure 4.13: Performance of YOLOv4 compared to state-of-the-art methods. It is possible to see that Scaled-YOLOv4 scores the highest average precision in this dataset, according to the original paper's authors.[49]

The semantic segmentation is performed using DeepLabV3+[50] on a PyTorch[51] implementation [52] utilizing a ResNet-101[53] backbone, which even though the former reaches state-of-the-art results in the PASCAL VOC[54] dataset, the latter creates a bottleneck in performance, since there are methods which currently perform better as feature extractors, such as ResNeXt[55], ResNest[56], and Xception[57].

However, since this report aims to present a proof of concept for 3D environment reconstruction, it was deemed that refactoring large sections of the previously proposed pipeline was unnecessary. Instead of using resources for a marginal improvement on the predictions of object bounding boxes and masks, some more focus can be given to exploring how the semantics should interact with the object ontologies and volumetric mapping.

4.3.1 Improvements on Two-Stage Object Segmentation Pipeline

The design of the pipeline was maintained from previous work. However, some aspects of it were either lacklustre or incompatible with the use case explored in this report.

One of the most significant changes that helped stabilise the system's inference times compared to the ones achieved in the previous work was to resize the cropped regions proposed by YOLOv4 by resizing them to a standard resolution. As a consequence, all the inputs to DeepLabV3+ are uniform, having the same inference time for all the proposed regions, causing it to have a linear time complexity $\mathcal{O}(n)$. Said resizing prevents the system from having an arbitrary time complexity which could have spanned from milliseconds in best-case scenarios to several minutes in worst-case scenarios. Furthermore, each image is normalised to [-1, 1], since the network is also trained in normalised images during the training stage.

The information flow of the segmentation stage of the network can be seen in Figure 4.14, while the one for object detection has remained untouched and can be seen in Figure B.1.

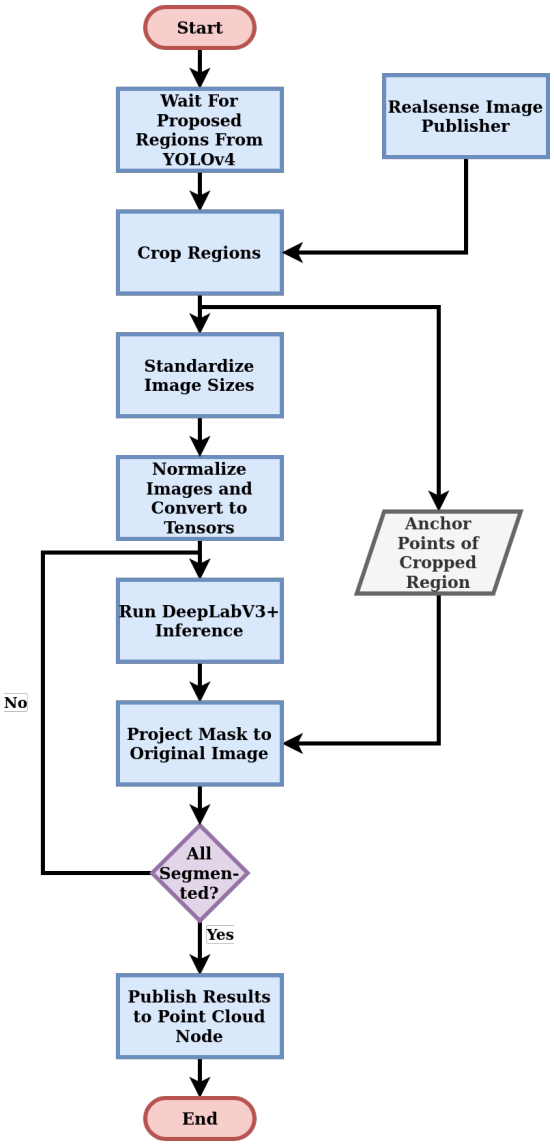


Figure 4.14: Information flow on the segmentation stage of the network with the added improvement of resizing image sizes.

The generated masks face a problem similar to one found in previous works [47], related to the fact that the images are modified to a different scale and ratio when resized, which results in the outputs having wrong proportions, as seen in Figure 4.15.

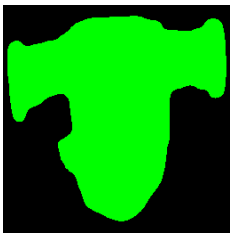


Figure 4.15: Mask output with the wrong scale and ratio.

The scale relates to the property that the image size has been changed, and therefore, the outputs from the segmentation neural network will be related to an image coordinate frame that does not represent the entire cropped region but only the resized version. The scale property can be recovered by transforming the resized coordinates into percentages and the percentages into coordinates on the original cropped region. Moreover, in the case that the aspect ratio between the cropped region and the resized image differ, scaling the images to the correct coordinate frame using the percentages of the two images will cause the outputs to be still skewed in a directly proportional way to the aspect ratio of the original cropped region. The percentile conversion does not account for different ratios on x and y. In order to circumvent this, the now scaled images must be multiplied by a ratio factor, which relates the scaled coordinates to the original cropped region correctly. This process is presented explicitly in equations 4.1 and 4.2, which were proposed in previous works.

$$x_{Cropped} = \frac{x_{resized}}{Width_{resized}} * Width_{Cropped} * Ratio \tag{4.1}$$

$$y_{Cropped} = \frac{y_{resized}}{Height_{resized}} * Height_{Cropped} * Ratio^{-1} \tag{4.2}$$

The end result from these transformations can be seen in Figure 4.16.

(a) Resulting mask from equations aforementioned. *(b) Mask overlay on object in cropped region.*

Figure 4.16: Example of mask post-processed with equations 4.1 and 4.2.

However, there is an extra step compared to the equations proposed previously since the masks must also be transformed into the coordinate frame of the original input image. In order to do so, all the coordinates of the masks are summed with the anchor point, which generates the cropped regions, effectively projecting them to the original coordinate system.

This post-processing is performed by applying equations 4.3 and 4.4 to the masks generated during the segmentation stage.

$$x_{Original} = x_{Cropped} + x_{AnchorPoint} \tag{4.3}$$

$$y_{Original} = y_{Cropped} + y_{AnchorPoint} \tag{4.4}$$

The result can be visualised in Figure 4.17, where a set of objects can be seen with their corresponding mask overlay projected on the original image.

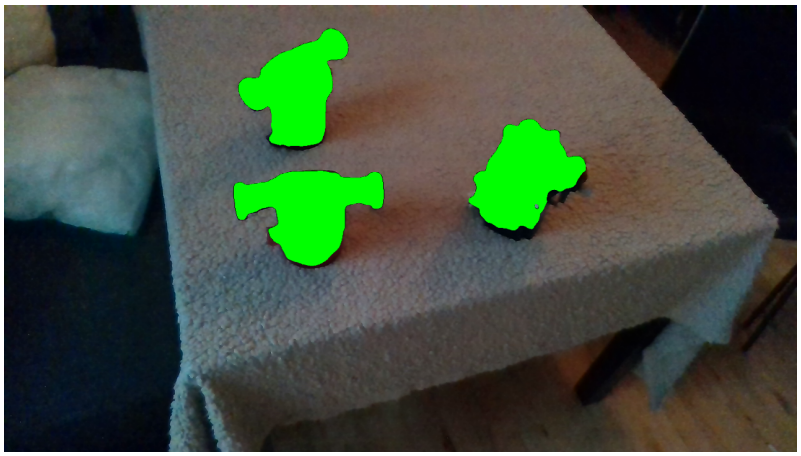


Figure 4.17: Resulting output from the post-processed masks provided by the semantic segmentation neural network.

At this stage, the pipeline is ready to be implemented into the broader system architecture.

4.3.2 Implementing Pipeline in Reconstruction Task

The two-stage segmentation pipeline is implemented as a part of the 3D environment reconstruction proposed in this report, where its task is to assign semantics to objects. As previously mentioned, the reconstruction is performed during offline and online stages. In the former, the pipeline assigns semantics to static objects which remain in the same place every time an online reconstruction is performed. The objects are classified by loading weights for YOLOv4 and DeepLabV3+, which is unloaded after the offline reconstruction is finished. The static object semantics are saved in files to be used by the online stage, where the bounding boxes of each of these objects are stored in the form of its minimum (X_{min}, Y_{min}) and maximum (X_{max}, Y_{max}) coordinates, alongside their class labels. In addition, the centroids of the masks computed are also stored. Furthermore, an RGB image with an overlay of each of the masks computed is also saved, as seen in Figure 4.18 and the information flow to generate the static masks is presented in Figure 4.19.

Figure 4.18: Computed mask during offline stage of a static object.

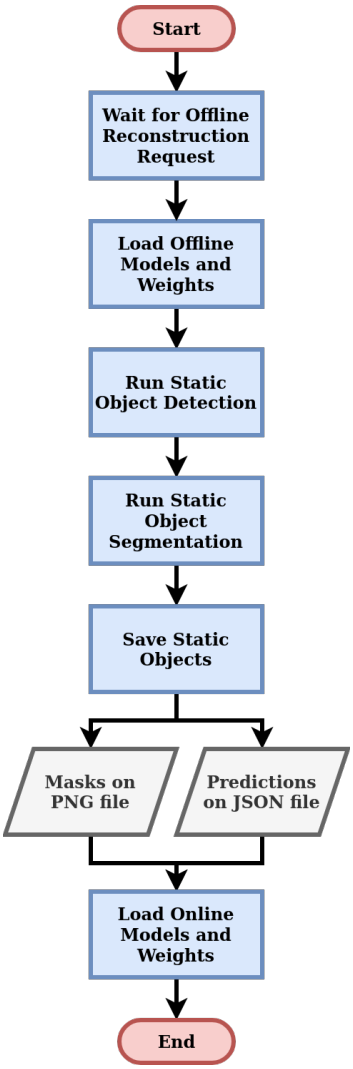


Figure 4.19: Flowchart of a request for offline reconstruction of an environment.

When a request to create an online representation is received, the detection pipeline loads the predictions describing static objects computed during the offline stage to detect and segment dynamic objects. Using the static object bounding boxes as landmarks, the system crops the input frames into regions of interest where dynamic objects may be found according to object ontologies, similarly to the concept presented in Figure 4.3. The result of the cropping based on the static objects can be seen in Figure 4.20, where a whole frame is cropped into a sub-region that corresponds to the bounding box of a table. Based on the ontologies, the system knows that pumps can be found on top of the table.

(a) Original frame used for inference during an online reconstruction.

(b) Cropped region of the frame, based on static and dynamic object relations.

Figure 4.20: Example of cropping images during online stage using the bounding box of a static object.

After this initial step, the bounding boxes are projected to the original image, and the dynamic objects are segmented. The resulting masks are then merged with the previously saved masks of the static objects in the environment, and the result can be seen in Figure 4.21.

Figure 4.21: Merged representations presenting the masks of static and dynamic objects.

The resulting merged representation is then projected to an octree, representing the distribution of objects in the environment while accounting for their distance to the camera, which can be seen in Figure 4.22. Meanwhile, the information flow during this stage of the reconstruction can be seen in Figure 4.23.

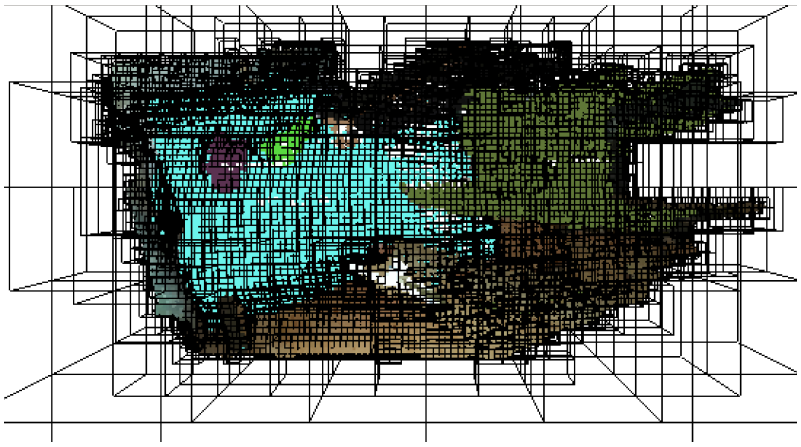


Figure 4.22: Merged representation projected into 3D space.

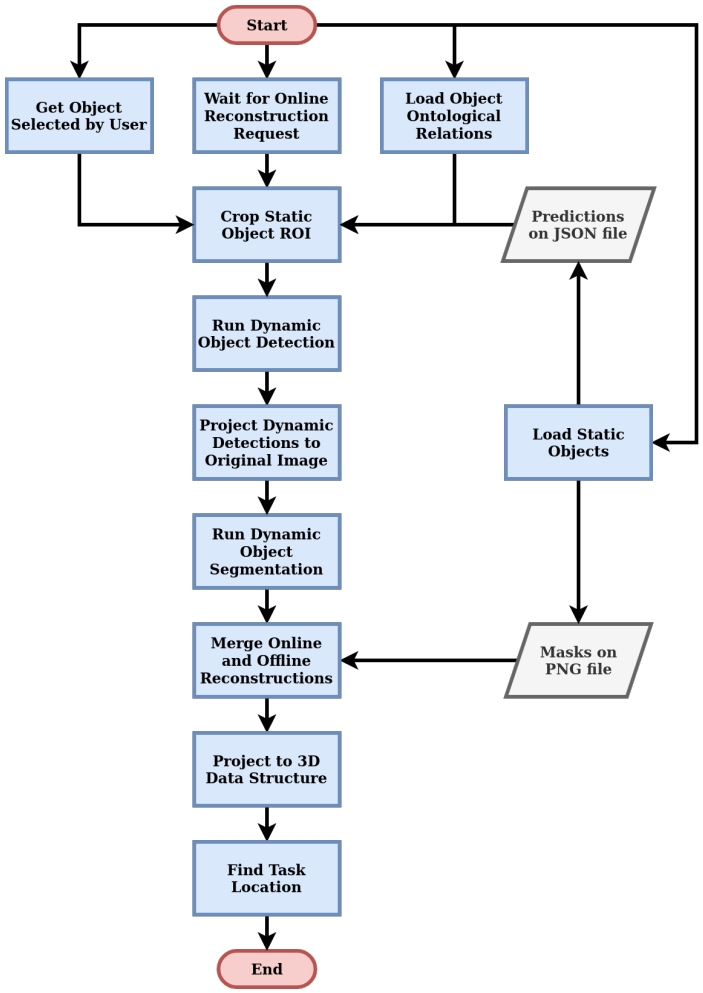


Figure 4.23: Flowchart of a request for online reconstruction of an environment, where a specific object is requested by the user.

The resulting representation is then shown to the user using a *Graphical User Interface* (GUI) presented in Section 4.4 (User Interfacing).

4.4 User Interfacing

In order to achieve an interactive and intuitive experience when utilizing the proposed pipeline in a physical robot, a Graphical User Interface (GUI) is proposed using *PySimpleGui*. The objective of GUIs is to remove unnecessary and confusing interactions of a user with the back-end of a system while providing a simple proof of concept. Especially in the case presented in this report, where it achieves to create an sHRC pipeline, users should interact with the system without having a large amount of knowledge in robotics and machinery.

The creation of the GUI was an iterative process of conceptualization and feedback by external parties. When building this node of the pipeline, the first step was to create visual feedback to the user on what was happening behind the scenes without struggling when looking for the information. In the initial design seen in Figure 4.24, three images were presented to the user, one with the objects detected, another with the masks generated, and a final one with the resulting point cloud from a masked image.

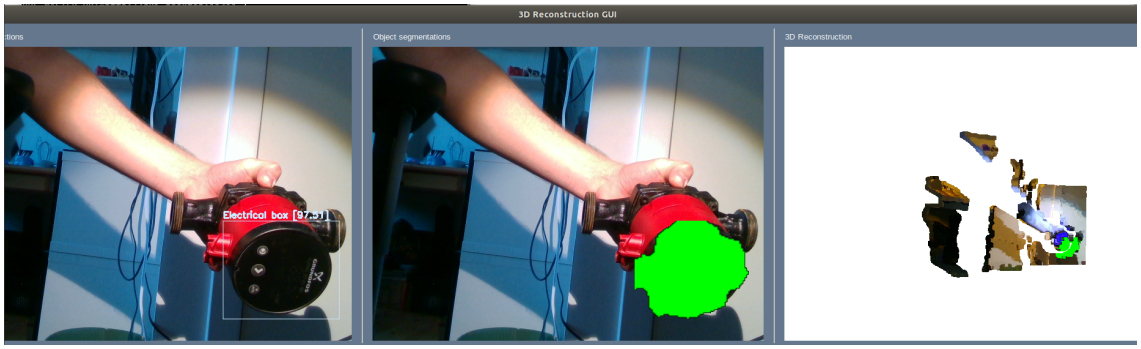


Figure 4.24: Initial iteration of the design of the GUI, presenting visual feedback to the user.

Upon some different conceptualization, it was decided that the interface should abide by some requirements:

- The interface should be simple and avoid having the user interacting directly with the back-end as much as possible;
- The interface should be capable of creating ontological relations;
- The interface should be able to request offline and online reconstructions of the environment by simulating requests from a physical robot;

For the first requirement, the visual feedback was organized into separate tabs with a description of each of the feedbacks and tool-tips on what they signify, which can be seen in Figure 4.25.

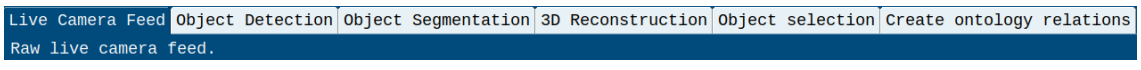


Figure 4.25: Tabs used to separate the different feedbacks and functionalities of the GUI for a more intuitive experience.

As for the second requirement, a separate tab, seen in Figure 4.26 was created in order to contain a handler in which users can select different dynamic objects and relate them to

static objects and tasks. This element of the GUI interacts indirectly with the ontologies node of the system by setting the ontological relations through a ROS service request containing the dynamic and static objects.

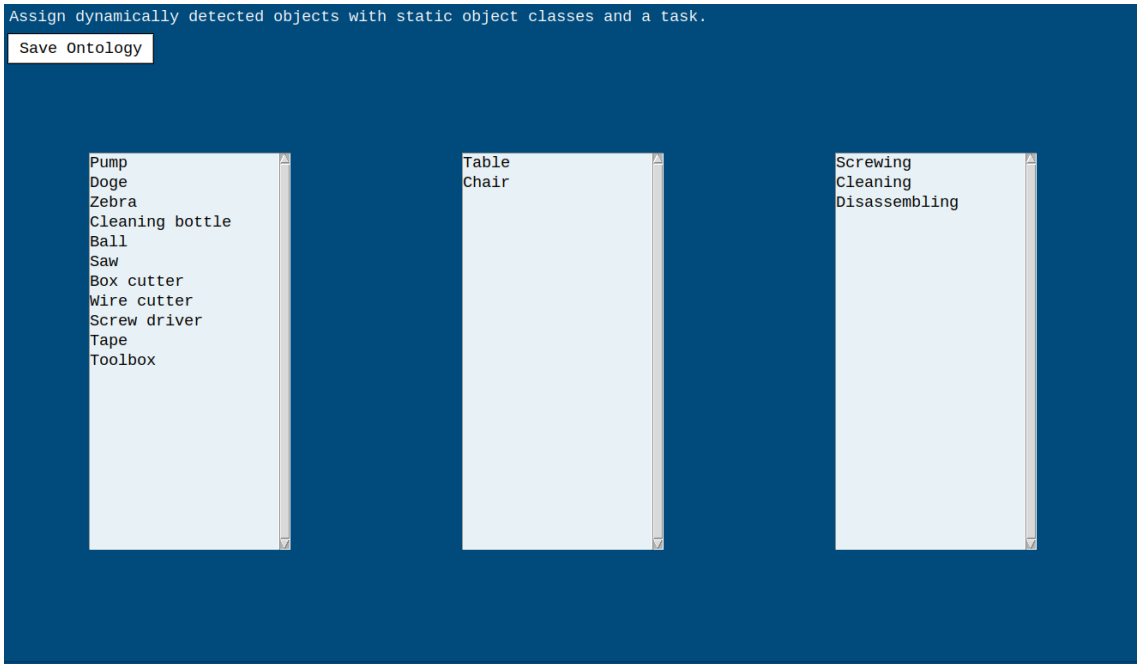


Figure 4.26: Tab used to handle user inputs regarding ontological relations of dynamic and static objects.

Furthermore, to use the ontological relations between static and dynamic objects, a tab containing buttons with images of different objects was added. When the user presses one of these buttons, the system sets the corresponding object as the object of interest for the next time an online reconstruction is performed, aiming to simulate the concept presented in Figure 4.3. The design of this tab can be seen in Figure 4.27.

Figure 4.27: Buttons used to request specific dynamic objects.

For the last requirement, two buttons to request online and offline reconstructions were created on a region of the GUI which is always visible, no matter which tab the user selects since this is the primary function of the system and should always be visible. The two buttons simulate robot requests for future-proofing by requesting the reconstructions using a ROS service with a flag to handle whether the system should look for static or dynamic objects. The buttons can be seen in Figure 4.28.

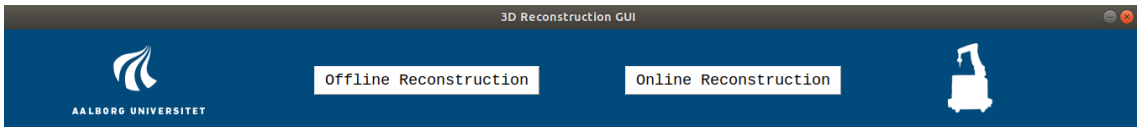


Figure 4.28: Buttons for offline and online reconstructions of the environment, located on a region of the GUI which is always visible.

The final design of the GUI can be seen in Figure 4.29 showing a 3D reconstruction of an environment.

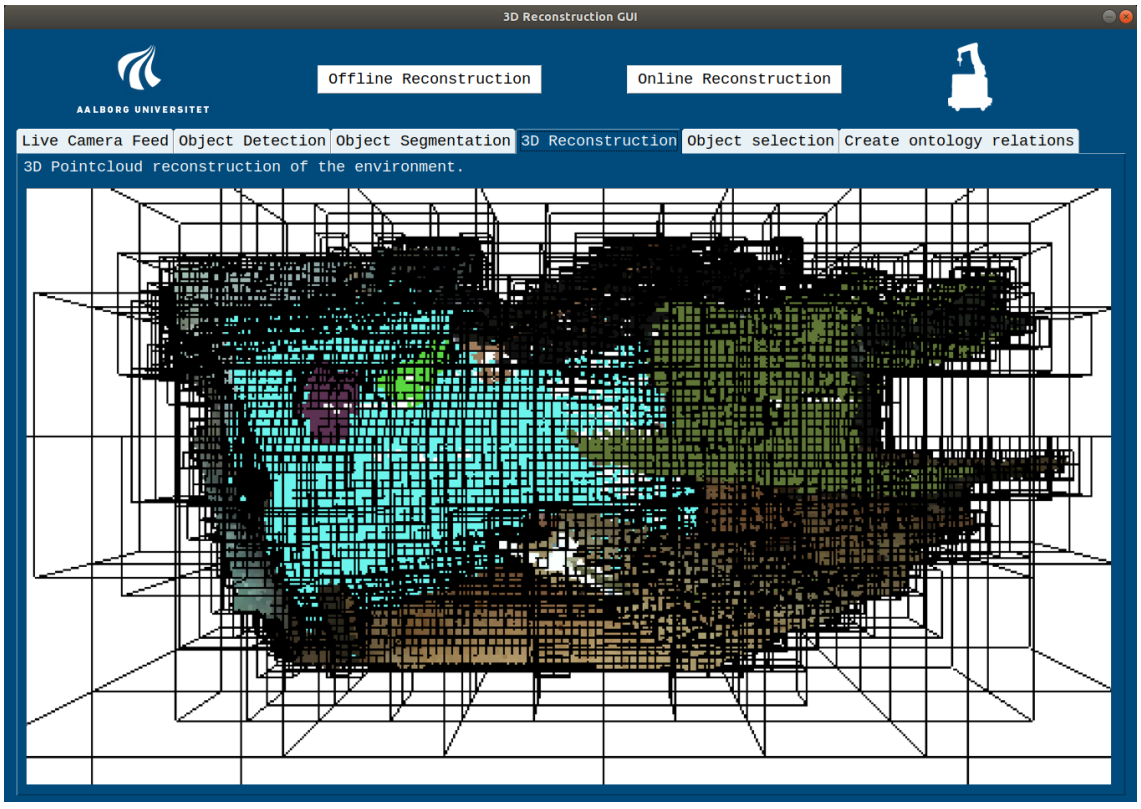


Figure 4.29: Final version of the GUI showing the 3D Reconstruction tab.

Finally, the different buttons described above cause the GUI to behave like a state machine, where each one of them causes a state change. These changes set the GUI into states that perform requests from other system nodes, such as the reconstruction pipeline, setting ontological relations, and setting a dynamic object of choice. The state machine can be observed in Figure 4.30.

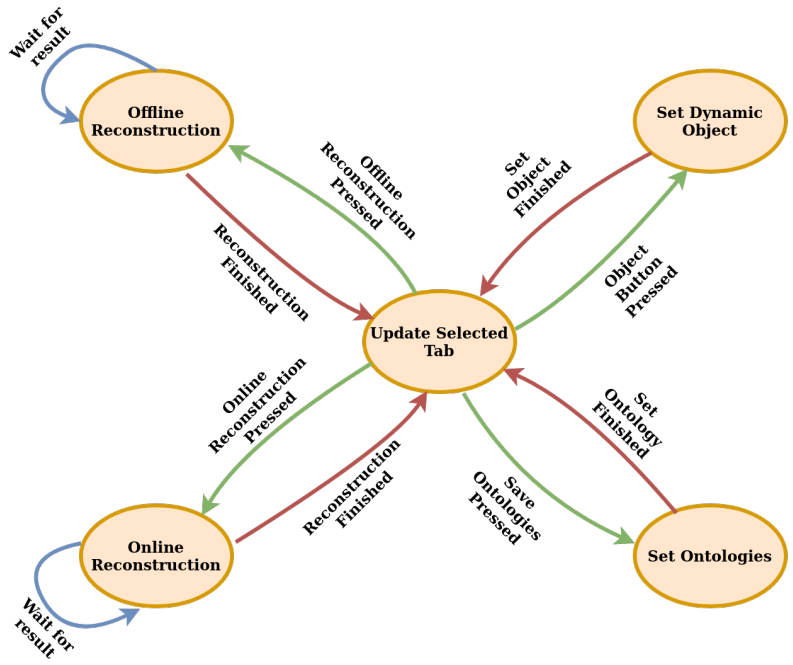


Figure 4.30: State machine modeling the behaviour of the GUI.

In short, this Chapter exposed the inner workings of the proposed system architecture. In Section 4.1 (A Case For Semantics) an object ontology handler using the JSON format was proposed, in which static and dynamic objects relations are stored. Section 4.2 (A Case For Semantics) presented different ways of representing 3D data and that octrees should provide the lightest data storage of the compared methods. Then, in Section 4.3 (Semantics Pipeline) a two-stage segmentation pipeline using DL is proposed, with a heavy focus on aspects that have improved since previous works. Finally, Section 4.4 (User Interfacing) presents a GUI which integrates all the systems and a representation of its behaviour as a state machine used to regulate the other aspects of the system. The proposed systems in this Chapter and their implementation are tested and reflected upon in Chapter 5 (Evaluations), where both qualitative and quantitative data is presented.

5 Evaluations

The system proposed in Chapter 4 (Methodology and Implementation) is tested upon in order to gather data on its performance and success of the concept proposed. Therefore, qualitative and quantitative tests are performed in Sections 5.2 (Qualitative Results) and 5.3 (Quantitative Results), utilizing the setup shown in Section 5.1 (Experimental Setup).

5.1 Experimental Setup

As previously mentioned, COVID-19 restrictions were in place during the duration of this project. Therefore, the test setup is limited compared to its full potential since all of it is realized in a domestic environment. Its current state is composed of a *Realsense* D415 Stereo Camera on a tripod, connected to a desktop computer with a GTX 970 GPU with 4GB of VRAM. The computer also contains 8GB of DDR3 RAM and utilizes an AMD FX-8320 CPU at $4.0GHz$.

5.1.1 Generating Datasets

Since the system at hand performs offline and online reconstructions, two sets of data were created. One set contains static objects and the other dynamic objects. Both these sets were saved similarly, using the *Realsense* camera and a script that stored images each time the keyboard was clicked. The outputs from this script became the data used to fine-tune YOLOv4's head, utilizing a total of 65 and 379 images for the offline and online reconstructions, respectively. Some examples of images used in these two datasets can be seen in Figure 5.1.

(a) Example of pictures used on offline dataset for object detection.

(b) Example of pictures used on online dataset for object detection.

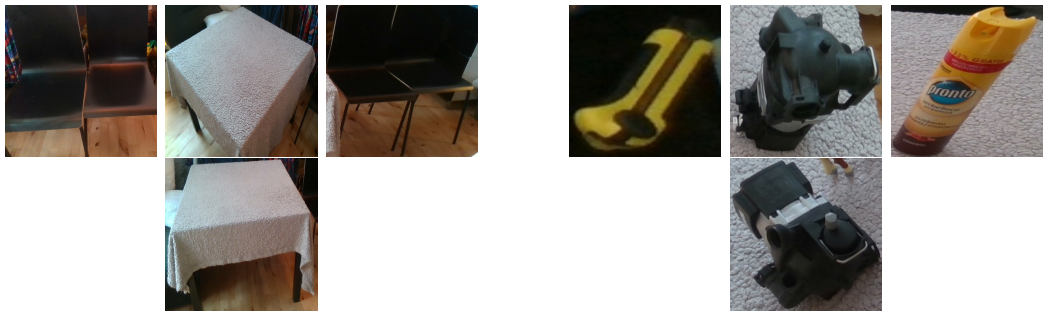
Figure 5.1: Online and offline datasets for object detection.

The classes included in the online dataset are composed of some objects possibly found in an industrial environment to create a proof of concept to perform a real-world implementation.

Meanwhile, the classes which form the offline dataset are *Table* and *Chair*. These two household objects can possibly be found in industrial environments. However, they work as placeholders for furniture, which may be more common in workspaces containing robots and in manufacturing environments.

The fine-tuning procedure in YOLOv4 was performed using hyperparameters provided by the original author of the *Darknet* implementation [48], training the offline model for a total of 6000 epochs in 2 classes and the online model for 22000 epochs in 11 classes, with an input size of 1120x1120 pixels, which were labelled with YOLO_mark [58]. The training procedure took close to 180 hours at *CLAAUDIA*[1], using one Tesla V100 GPU.

The resulting models are used in another script to generate the datasets for DeepLabV3+, utilizing the same pictures used to train YOLOv4. The weights for the reconstructions and images from the dataset are loaded using the script and cropped into smaller images containing the objects detected by YOLOv4. The resulting cropped images can be seen in Figure 5.2. In the end, a total of 99 different cropped images were generated from the initial 65 used for the offline reconstruction dataset, while 484 cropped images were created from the 379 images in the online reconstruction dataset. The images were also resized to a standardized size of 512x512 pixels.



(a) Example of pictures used on offline dataset for object segmentation.

(b) Example of pictures used on online dataset for object segmentation.

Figure 5.2: Online and offline datasets for object segmentation.

This data was labelled using *Labelme* [59] and then transformed into the Pascal VOC data format, resulting in the ground truths used in the fine-tuning of DeepLabV3+. This network was initially set to train for 500 epochs; however, it contains an early stop to save the best set of weights to avoid overfitting. The fine-tuning procedure took around two hours utilizing *CLAAUDIA* with one Tesla V100 GPU for each of the reconstructions models.

5.2 Qualitative Results

A qualitative analysis of the proposed system is performed by observing its overall behaviour when performing a reconstruction task.

[1] Cloud service at AAU for AI applications.

The system successfully assigns semantics to static objects during an offline reconstruction requested by the user utilizing the GUI. The process of loading the offline model and reloading the online model is quite extensive time-wise, clocking at 24*s* to perform this whole process. However, since this process happens during the offline reconstruction, this would not be a problem if this system was applied in a manufacturing sHRC setting. The flushing and loading procedures would not add any extra lead time to the process. In ideal conditions, it should only happen when there are changes in static elements of the environment. The result of the pipeline proposed in this report can be seen in table 5.1, under the tab *reconstruction*. Alongside the *Input Image* used to perform detections, the results from a raw implementation of *DeepLabV3+*, and the results of the *Two-stage Segmentation Pipeline* are also shown as means of comparison.

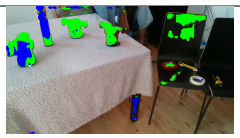
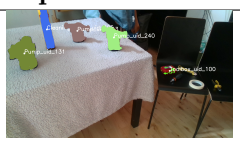
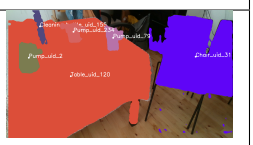
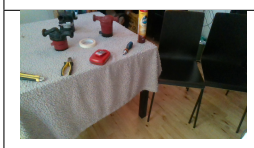
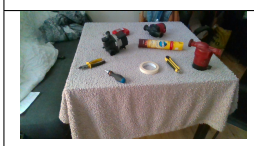
Input Image	**DeepLabV3+**	**Two-stage Segmentation Pipeline**	**Reconstruction**
			
			
			

Table 5.1: Figures showing the resulting masks using different methods on the same input image.

It is possible to observe that DeepLabV3+ when implemented by itself, seems to detect some of the objects in the environment. However, it does seem to pick up some rather large amount of noise from the background, if there is scene clutter.

As for the two-stage segmentation pipeline, most objects seem to be correctly detected if placed on top of a white surface (table). However, it seems the system has trouble picking up detections on dark objects (chairs), which could be related to the dataset used. It seems the problem worsens during the segmentation stage since the system can propose bounding boxes in these regions. However, it fails to segment the objects. It became apparent that the segmentation network is more prone to biases regarding illumination and colour differences than object detection.

The reconstruction pipeline provides a semantics rich representation of the environment between the static and dynamic objects. The connections through ontologies of the dynamic and static objects allow targeted searches in the environment. Furthermore, the system can infer the task a dynamic object can be used for and where it can be used based on coordinates of static objects through the ontologies.

5.3 Quantitative Results

Though the qualitative results provide a subjective approach to the results of the system, showing that it is capable of reconstructing the environment, it is also essential to quantify the system's performance. Doing this makes it possible to know exactly how well the object detection and segmentation behaved and how they compare to state-of-the-art methods.

5.3.1 Two-stage Segmentation Pipeline Performance in Custom Dataset

Utilising the custom made dataset presented in Section 5.1.1 (Generating Datasets), some metrics which describe the performance of the system were measured.

A test is conducted on the YOLOv4 model trained on the dynamic objects dataset. A validation dataset was created containing 25 unique images in order to compute the *Precision* (P), *Recall* (R),[60] and mean *Average Precision* (mAP)[61]. This specific dataset comprises new images, which are not included in the training and test datasets.

mAP is a metric that measures *Average Precision* (AP) of models in all the classes in which they were trained. The mAP score corresponds to the *Area Under Curve* (AUC) of the *precision-recall curve*, which is calculated with Equation 5.1.[61]

$$mAP = \frac{1}{n} \sum_{n} \left(\int_{a}^{b} P(R) dR \right)_n \tag{5.1}$$

Precision describes the percentage of *true positives* in all of the recalled instances, and recall is the percentage of *true positives* in all the existing instances. Figure 5.3 presents the concept of the two metrics, showing how they are computed inside the blue boxes.

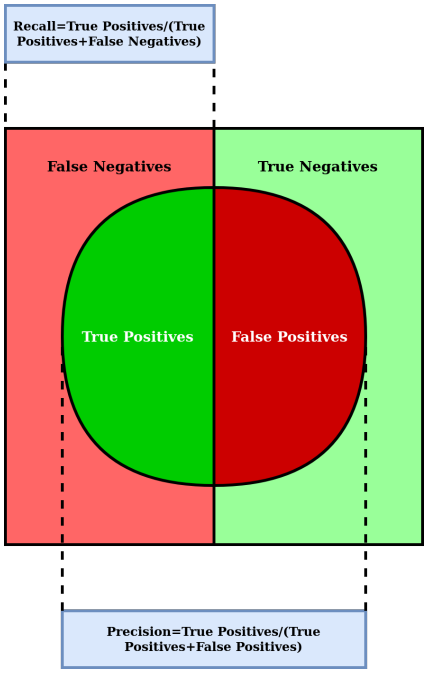


Figure 5.3: Visualization of the Precision and Recall metrics.

The mAP varies depending on the detection acceptance threshold for *Intersection over Union* (IoU). IoU represents the proportion in which a bounding box of a ground truth intersects with the model's detection. The concept of IoU can be seen in Figure 5.4.

Figure 5.4: Example of IoU, where the yellow region represents the intersection of the ground truth (red) and the detection (green). Adapted from [37].

The mAP was measured at different thresholds of the IoU. The data can be visualised in the form of a curve in Figure 5.5, and the specifics of the data are presented in Table A.3.

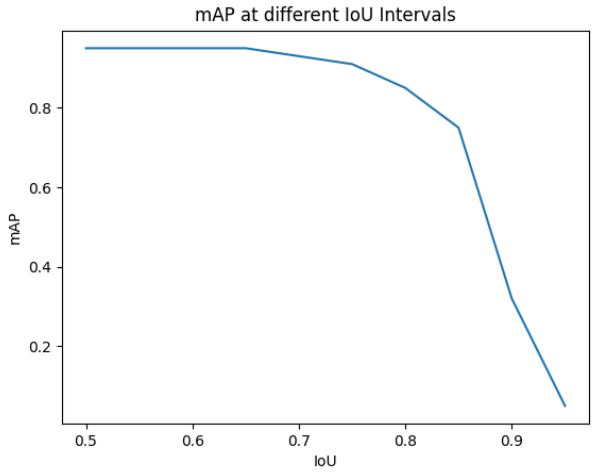


Figure 5.5: mAP at IoU=.50:.05:.95 of YOLOv4 in the proposed implementation.

The results presented above were attained using the inbuilt tool in *Darknet*. Utilising an IoU threshold @0.65, in order to have the most considerable amount of overlap between ground truths and predictions without sacrificing mAP, a *precision-recall curve* was plotted. The values used to plot the curve were acquired by tweaking the minimum confidence necessary to accept a detection as a true positive. The curve can be seen in Figure 5.6.

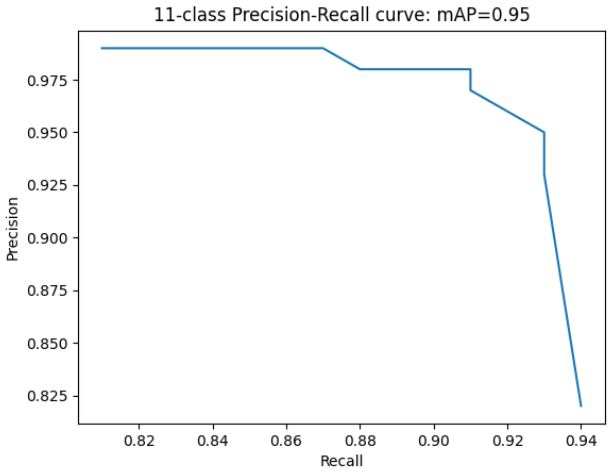


Figure 5.6: Precision-recall Curve of YOLOv4 in the proposed implementation using different confidence values.

In addition to the object detection tests presented above, a validation of the segmentation model was also done. In this case, the validation dataset was generated by cropping objects detected by the object detection model to simulate the two-stage object segmentation conditions closely.

The metrics used to measure the performance of this dataset were the same as used in Long *et al.* [62], *Pixel Accuracy* (PA), *Mean Pixel Accuracy* (mPA), *Mean IoU* (mIoU), and *Frequency Weighted IoU* (fwIoU), plus the *Mean Inference Time* (mIT). PA quantifies the percentage of pixels that are *true positives* (n_{ii}) in the total amount of pixels of the class in an image (t_i). mPA quantifies the percentage of pixels that were correctly classified in an image averaged for the number of classes (n_{cl}). The mIoU quantifies the average IoU of the model for a pixel class, where n_{ij} represents the number of *false negatives* pixels. Meanwhile, the fwIoU considers the class frequency when computing the mIoU. The results are presented in Table 5.2. The higher the value for each of the metrics, the better, except for inference time, which smaller values report better performance. The metrics can be computed using equations 5.2-5.5, which are shown also in Long *et al.* [62].

$$PA = \sum_i \frac{n_{ii}}{t_i} \tag{5.2}$$

$$mPA = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i} \tag{5.3}$$

$$mIoU = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - nii} \tag{5.4}$$

$$fwIoU = (\sum_k t_k)^{-1} \sum_i \frac{t_i n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{5.5}$$

Input Resolution (pixels)	**PA (%)**	**mPA (%)**	**mIoU (%)**	**fwIoU (%)**	**mIT (s)**
64x64	50.95	11.10	5.88	27.00	2.40
128x128	50.28	10.95	8.26	37.92	1.61
256x256	50.00	11.00	9.92	45.40	6.70
512x512	49.65	11.04	10.27	46.84	27.59

Table 5.2: Metrics used in Long et al. [62] to measure performance of object segmentation at different resolutions.

It seems there is a relationship between a resolution increase and a higher value in both of the IoU metrics. This performance improvement comes at the cost of higher inference times, except for the increase from 64x64 pixels to 128x128 pixels. Furthermore, it seems that the pixel accuracy value has remained chiefly stagnant between the different resolutions.

Though the values above allow quantifying the system's performance in the custom dataset, it is essential to compare the model's performance to pre-existing CNN architectures. Therefore, a set of extra quantitative tests was performed in the next section.

5.3.2 Two-stage Segmentation Pipeline Performance in COCO Dataset

In order to be able to evaluate the capabilities of the two-stage segmentation pipeline honestly, models trained on the COCO dataset were used. Having such models allows a direct comparison to state-of-the-art methods. The validation dataset provided on the COCO challenge website can measure different metrics for object detection and segmentation.

Method	**AP^{box} (%)**
GCNet[63]	51.80
ResNeSt-200[56]	52.50
SpineNet-190[64]	52.60
EfficientDet-D7[65]	54.40
Copy-paste[66]	55.9
Swin-L[67]	58.0
YOLOv4 in proposed implementation	**42.30**

Table 5.3: Bounding box mAP of different state-of-the-art network designs, compared to the proposed implementation based on YOLOv4 at the resolution of 960 x 960 pixels.

The same dataset was used to evaluate the performance of the segmentation of object masks. The metrics utilised for this comparison is the mIoU. The results can be visualised in Table 5.4, compared to other state-of-the-art methods.

Method	**mIoU (%)**
DSSPN[68]	38.90
SVCNet[69]	39.60
EMANet[70]	39.90
SPYGR[71]	39.90
OCR[72]	39.50
DANet[73]	39.70
CAA[74]	41.20
DeepLabV3+[50][52]	0.90
DeepLabV3+ in proposed implementation	**3.85**

Table 5.4: Mask mIoU in COCO-stuff dataset for segmentation in different state-of-the-art methods, compared to the proposed implementation based on DeepLabV3+.

5.3.3 Inference Time on Two-stage Segmentation Pipeline

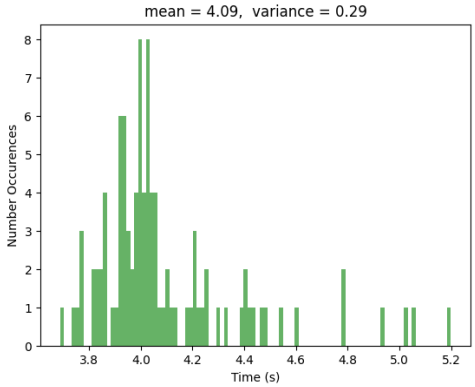
The purpose of the two-stage segmentation pipeline is to reduce the inference time of the segmentation of objects since only ROIs proposed by YOLOv4 on the original input image are segmented.

The testing procedure used to measure this metric comprises four trials, each composed of 100 iterations. In each iteration, dynamic objects are detected and segmented since these objects are time-critical compared to static objects. Three of the trials were done using the method proposed by the author [47], while the other one uses a raw implementation of DeepLabV3+ in which a single trial is conducted since adding or removing objects does not affect inference time.

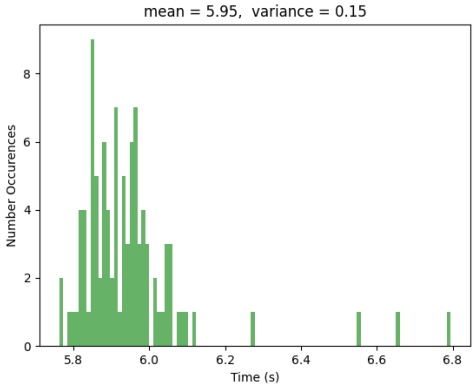
For the tests using the two-stage segmentation pipeline, the input resolution 960x960 pixels is used for YOLOv4, and 256x256 pixels is used for DeepLabV3+. Such an input resolution for the segmentation network keeps it very close to the size of the cropped region from the original image, which should create fair testing conditions. Therefore, the raw implementation of DeepLabV3+ is set to use the native resolution of the camera as input at 1280x720 pixels.

During the trials on the proposed pipeline, one extra object was added at each of them. The data gathered over the 100 iterations of each of the trials can be seen in Figures 5.7a to 5.7c in the form of histograms.

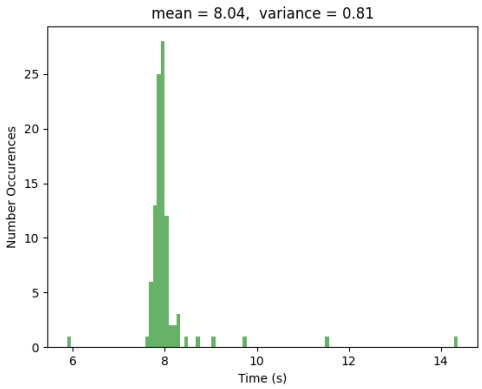
During the trial conducted on the raw implementation of DeepLabV3+, the same object number was placed in the camera's FOV, and inference was performed 100 times. However, the difference in the results was negligent. Therefore, only one of the trials is presented. The resulting data from this test can be seen in Figure 5.7d in the form of a histogram. In the Figure, μ stands for the mean of the distribution, while σ stands for the variance.



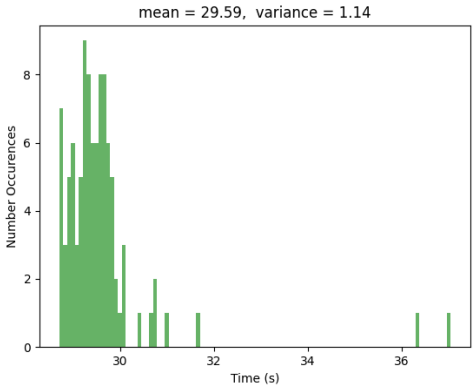
(a) Histogram of first trial. $\mu = 4.09s$ $\sigma = 0.29s$



(b) Histogram of second trial. $\mu = 6.58s$ $\sigma = 0.15s$



(c) Histogram of third trial. $\mu = 8.80s$ $\sigma = 0.96s$



(d) Histogram of DeepLabV3+ implementation. $\mu = 30.73s$ $\sigma = 1.96s$

Figure 5.7: Histograms presenting the inference times of three trials of the newly proposed two-stage segmentation pipeline (a, b, c) and raw DeepLabV3+ implementation (d).

The distributions and their means were plotted in a box-whiskers plot to visualise better how the data is scattered. It is possible to observe that some of the data does not fall within the boxes nor the whiskers limits. However, it is possible to see the linear relationship mentioned in Section 4.3 (Semantics Pipeline), through the means of three distributions, which grows with each object added to the test.

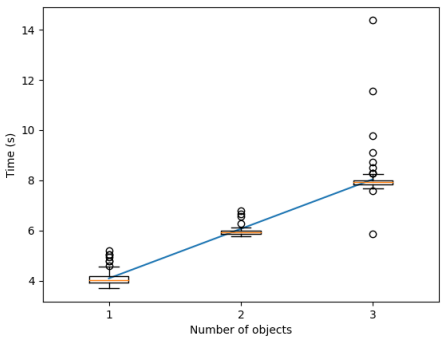
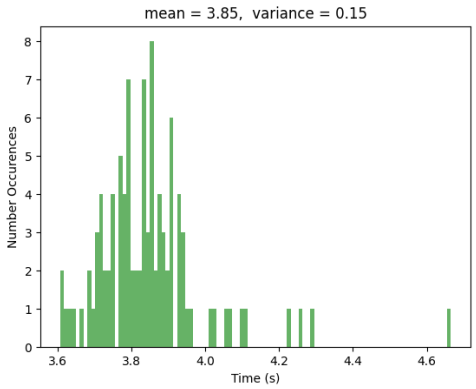
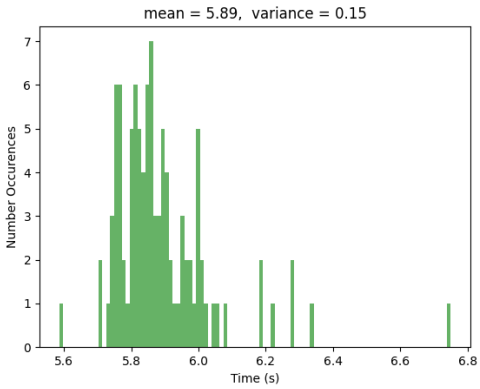


Figure 5.8: Box-whiskers plot presenting the relationship between the number of objects added to the scene and the inference time.

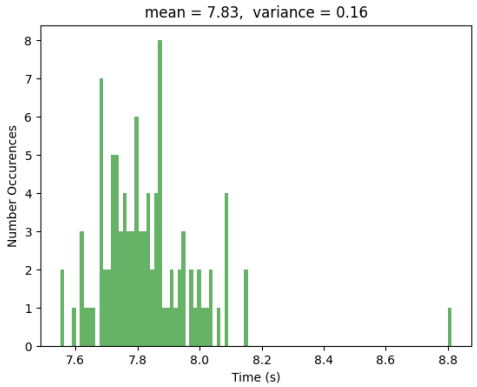
Furthermore, a set of trials was conducted using the ontological relations between static and dynamic objects. The process was similar to the previous test. Each trial ran for a total of 100 iterations, and the same amount of objects was used. The results are presented in Figure 5.9 in the form of histograms. In the Figure, μ stands for the mean of the distribution, while σ stands for the variance.



(a) Histogram of first trial. $\mu = 3.85s$ $\sigma = 0.15s$



(b) Histogram of second trial. $\mu = 5.89s$ $\sigma = 0.15s$



(c) Histogram of third trial. $\mu = 7.83s$ $\sigma = 0.16s$

Figure 5.9: Histograms presenting the inference times of three trials of the newly proposed two-stage segmentation pipeline (a, b, c) using object ontologies to infer on ROIs.

In general, the performance of the system is satisfactory as a proof of concept. The results obtained in this Chapter are reflected upon in Chapter 6 (Discussion and Conclusion) where the successes, shortcomings, and possible improvements of the system can be found.

6 Discussion and Conclusion

This chapter reflects on the performance of the proposed two-stage segmentation pipeline and the 3D reconstruction of the environment with object ontologies. Some of the discussion topics relate to the successes and issues during development and how this knowledge can guide future developments of the proposed system.

A novel pipeline to assign semantics to an environment was created throughout this project. Such a system can aid workers with little knowledge of robotics to collaborate alongside cobots. The development during this thesis managed to reach the proof of concept and complete the scope proposed in Chapter 1 (Introduction). For the most part, the system produced satisfying qualitative results, especially accounting for the domestic environment where it was tested and the short inference times compared to a raw implementation of DeepLabV3+. In its current stage, integrating the system into a cyber-physical system depends on its integration into a physical cobot. The system can also achieve a small degree of cognition by considering relations between objects and places where tasks are performed through object semantics, ontologies, and the projection of data into a 3D reconstruction.

Currently, the system is restricted to only having the *Realsense* camera as its hardware. Therefore, it is complicated to create transformations of the camera's pose in the environment, which led to the restriction where the camera only generates 3D reconstructions of what it is currently seeing. In order to add this system to a cyber-physical system, there is a need to place the camera on a robot and perform a hand-eye calibration in order to be able to map the entire environment surrounding the robot's workspace. The necessity for integration is a minor issue, which could be considered part of the natural progression of this project's long-term development.

The system manages to run on an outdated computer, which is pretty remarkable since the CNNs utilised are modern designs while achieving the previously mentioned short inference times. However, to fully apply this system in a real-world setting, there is a need to address a memory leak in the object segmentation neural network. Currently, the *PyTorch* implementation used for DeepLabV3+ runs on the CPU side since the GTX 970 used does not have enough VRAM to run both models in parallel. The memory leak is induced every time the user calls for an offline reconstruction where the online weights should be flushed. Furthermore, once the static objects are found, the offline weights should also be flushed to make room for the online model yet again. However, PyTorch does not contain any method to flush memory from the RAM, only from CUDA, in the GPU. Therefore, each time the weights are loaded, they are never flushed, increasing the amount of memory used at each iteration. Such a problem could be easily solved by utilising a newer GPU, where

the VRAM constraint is not a problem, allowing both models to run in parallel.

Furthermore, the proposed implementation for object segmentation has increased performance in the mIoU metric in the COCO-Stuff dataset, compared to the one achieved by the raw implementation of DeepLabV3+. Therefore, it seems that utilising the implementation by [52] leads to a bottleneck on the proposed method since the values achieved during testing do not reflect the ones stated in the implementation [52] nor in the original paper [50]. The performance is lacklustre and falls behind many state-of-the-art methods. Therefore, an obvious step in future works would be to move away from this implementation to another one with better support and results.

During the testing of the system, it was possible to observe that there is currently a significant problem regarding the environment's lighting conditions. Most datasets were taken when the natural was abundant, which causes a decrease in the system's accuracy when the weather is overcast. In this specific case, it is impossible to control this variable because the experiments were realised in a domestic environment where controlled light conditions cannot be achieved. In the end, some of the data, which is part of the custom dataset, ended up having low quality, as seen in Figure 6.1. However, the system still managed to produce satisfying qualitative results, leading to the belief that even in non-optimal lighting conditions, the system still has some merit. Furthermore, during previous works [47], the system produced high-quality detections with a controlled light source. Therefore, this problem is not so much an architecture flaw but an easily solvable environmental issue since it is expected that manufacturing environments have controlled environmental lighting.

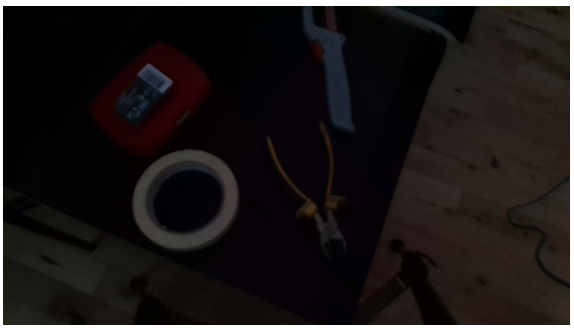


Figure 6.1: Problematic frame part of the dynamic objects dataset.

In general, the qualitative data acquired shows that the proof of concept works, and it can provide a reconstruction of the environment, compared to simply using a method such as DeepLabV3+ or the two-stage segmentation pipeline proposed. Furthermore, there is a need to develop a more intuitive way of users interacting with the system without using a GUI, since, at its current stage, it requires *a priori* knowledge of the system. A workaround for such an issue could be achieved by informing the robot that they will pick up an object or perform a task through natural language processing or gesture recognition.

Regarding the quantitative data, the two-stage object segmentation pipeline performs better on the COCO-stuff dataset than the raw implementation of DeepLabV3+, with an increase of 2.95% in the mIoU metric, even though both perform worse than the state-of-the-art methods. On the other hand, the performance of YOLOv4 was much

closer to the expected value at 42.30% in the mAP metric, performing still slightly worse than other state-of-the-art methods. However, since the objective of the thesis is not to create a state-of-the-art segmentation architecture, the results are deemed acceptable for proof of concept, especially since the inference using the pipeline is faster than using a raw implementation of the segmentation network. Furthermore, by utilising the object ontologies and the offline reconstruction of the environment, the inference time of the proposed segmentation pipeline marginally goes down. It is also possible to observe in the data that there seems to be a smaller number of outliers in the method using the reconstruction to propose ROIs than utilising the proposed pipeline on entire images.

One interesting aspect to also possibly explore is to move towards having both of the models running on the same framework. The interactions of Darknet and Pytorch with each other create, at times, complex situations which can quickly become convoluted when attempting to perform tests or refactor sections of the code. Furthermore, there is a chance that the code would run faster by using methods that are native to a specific framework instead of having cross-communication between different frameworks.

In conclusion, this thesis proposes a 3D environment reconstruction pipeline that can be applied in sHRC scenarios between a cyber-physical system and a human worker. Even though the project's scope was quite limited in terms of environment, the system could recreate an environment's semantics and use them to alongside ontological relations. There is much potential in the concept itself for collaboration cases, and it would be interesting to explore interactions with humans in the long run with the system integrated into a physical cobot.

Bibliography

[1] Nikolaos Nikolakis, Vasilis Maratos, and Sotiris Makris. A cyber physical system (cps) approach for safe human-robot collaboration in a shared workplace. *Robotics and Computer-Integrated Manufacturing 56*, 2019.

[2] Xi Wang, Zsolt Kemény, József Váncza, and Lihui Wang. Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. *CIRP Annals 66, no. 1*, 2017.

[3] L. Wang, R. Gao, J. Váncza, J. Krüger, X. V Wang, S. Makris, and S. Chryssolouris. Symbiotic human-robot collaborative assembly. *CIRP Annals 68, no. 2*, 2019.

[4] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse. Collaborative manufacturing with physical human–robot interaction. *Robotics and Computer-Integrated Manufacturing 40*, 2016.

[5] Hongyi Liu, Tongtong Fang, Tianyu Zhou, Yuquan Wang, and Lihui Wang. Deep learning-based multimodal control interface for human-robot collaboration. *51st CIRP Conference on Manufacturing Systems, 72*, 2018. URL `https://doi.org/10.1016/j.procir.2018.03.224`.

[6] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford. Place categorization and semantic mapping on a mobile robot. pages 5729–5736, 2016.

[7] U.S. Department of Labor. *OSHA Technical Manual, Industrial Robots and Robot System Safety.* Occupational Safety and Health Administration.

[8] Laszlo Monostori, Botond Kádár, Thomas Bauernhansl, Shinsuke Kondoh, Soundar Kumara, Gunther Reinhart, Olaf Sauer, Günther Schuh, Wilfried Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals - Manufacturing Technology*, 65:621–641, 08 2016.

[9] Hung-An Kao, Wenjing Jin, David Siegel, and Jay Lee. A cyber physical interface for automation systems—methodology and examples. *Machines*, 3(2):93–106, 2015. URL `https://www.mdpi.com/2075-1702/3/2/93`.

[10] Azfar Khalid, Pierre Kirisci, Zied Ghrairi, Jürgen Pannek, and Klaus-Dieter Thoben. Safety requirements in collaborative human robot cyber physical systems. *Dynamics in Logistics - Proceedings of LDIC*, 2016. URL `https://www.researchgate.net/publication/295912158_Safety_Requirements_in_Collaborative_Human_Robot_Cyber_Physical_Systems`.

[11] International Organization for Standardization. Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design. Standard, International Organization for Standardization, Geneva, Switzerland, 12 2015.

[12] International Organization for Standardization. Safety of machinery — Safety-related parts of control systems — Part 2: Validation. Standard, International Organization for Standardization, Geneva, Switzerland, 10 2012.

[13] International Organization for Standardization. Safety of machinery — Positioning of safeguards with respect to the approach speeds of parts of the human body. Standard, International Organization for Standardization, Geneva, Switzerland, 05 2010.

[14] International Organization for Standardization. Risk management. Standard, International Organization for Standardization, Geneva, Switzerland, 02 2018.

[15] Amedeo Cesta, Andrea Orlandini, Giulio Bernardi, and Alessandro Umbrico. Towards a planning-based framework for symbiotic human-robot collaboration. *In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, page 1–8, 2016. URL `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7733585`.

[16] Hongyi Liu, Yuquan Wang, Wei Ji, and Lihui Wang. A context-aware safety system for human-robot collaboration. *Procedia Manufacturing 17*, page 238–245, 2018. URL `https://doi.org/10.1016/j.promfg.2018.10.042`.

[17] Luis Pérez, Silvia Rodríguez-Jiménez, Nuria Rodríguez, Rubén Usamentiaga, Daniel F. García, and Lihui Wang. Symbiotic human–robot collaborative approach for increased productivity and enhanced safety in the aerospace manufacturing industry. *The International Journal of Advanced Manufacturing Technology*, page 851–863, 2020. URL `https://link.springer.com/article/10.1007/s00170-019-04638-6`.

[18] B. Bayat, J. Bermejo-Alonso, J. Carbonera, T. Facchinetti, S. Fiorini, P. Goncalves, V.A.M. Jorge, M. Habib, A. Khamis, K. Melo, B. Nguyen, J.I. Olszewska, L. Paull, E. Prestes, V. Ragavan, S. Saeedi, R. Sanz, M. Seto, B. Spencer, A. Vosughi, and H. Li. Requirements for building an ontology for autonomous robots. *Industrial Robot, Vol. 43 No. 5*, 2016.

[19] Chen Jiang, Masood Dehghan, and Martin Jagersand. Understanding contexts inside robot and human manipulation tasks through a vision-language model and ontology system in a video stream, 2020.

[20] Mark A. Musen. The protégé project: A look back and a look forward, 2015.

[21] Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer, and Zoltan-Csaba Márton. Robosherlock: Unstructured information processing for robot perception. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[22] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[23] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments, 2009.

[24] D. G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1150–1157, 1999.

[25] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. *Proc. ECCV*, 6313, 2010. ISBN 978-3-642-15558-1.

[26] J. I. Olszewska, M. Barreto, J. Bermejo-Alonso, J. Carbonera, A. Chibani, S. Fiorini, P. Goncalves, and et al. Ontology for autonomous robotics. *26th IEEE International Symposium on Robot and Human Interactive Communication*, 2017.

[27] Ian Niles and Adam Pease. Towards a standard upper ontology. *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*, 2001.

[28] Liam Paull, Gaetan Severac, Guilherme V. Raffo, Julian Mauricio Angel, Harold Boley, Phillip J Durst, Wendell Gray, and et al. Towards an ontology for autonomous robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[29] Joanna Olszewska and T. Mccluskey. Ontology-coupled active contours for dynamic video scene understanding. *INES 2011 - 15th International Conference on Intelligent Engineering Systems, Proceedings*, pages 369–374, 06 2011. ISBN 978-1-4244-8954-1.

[30] Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez. Methodologies, tools and languages for building ontologies. where is their meeting point? *Data & Knowledge Engineering 46, no. 1*, pages 41–64, 07 2003.

[31] Matthew Horridge. A practical guide to building owl ontologies using protégé 4 and co-ode tools edition 1.3, 2011. URL `http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf`.

[32] Cici Li and Guohui Tian. Transferring the semantic constraints in human manipulationbehaviors to robots. *Applied Intelligence*, 2020. URL `https://doi.org/10.1007/s10489-019-01580-8`.

[33] Ioannis Kostavelis and Antonios Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015. ISSN 0921-8890. URL `Addsourcelinkifyouhaveit`.

[34] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. pages 487–495, 2014.

[35] Mathias Mantelli, Diego Pittol, Renan Maffei, Jim Torresen, Edson Prestes, and Mariana Kolberg. Semantic active visual search system based on text information for large and unknown environments. 2021. URL https://doi.org/10.1007/s10846-020-01298-7.

[36] David Fernandez-Chaves, Jose-Raul Ruiz-Sarmiento, Nicolai Petkov, and Javier Gonzalez-Jimenez. From object detection to room categorization in robotics. *In Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, 2020. URL https://doi.org/10.1145/3378184.3378230.

[37] Rune Grønhøj, Jan Kjær Jørgensen, Guilherme Mateus, and Jacob Krunderup Sørensen. Semantic place categorisation and object detection using cnns. Technical report, Aalborg Universitet, 2019.

[38] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[39] John G. Rogers and Henrik I. Christensen. Robot planning with a semantic map. 2013. URL https://doi.org/10.1109/ICRA.2013.6630879.

[40] Tiago S. Veiga, Pedro Miraldo, Rodrigo Ventura, and Pedro U. Lima. Efficient object search for mobile robots in dynamic environments: Semantic map as an input for the decision maker. 2016. URL https://doi.org/10.1109/IROS.2016.7759426.

[41] Tobias Werner, Maximilian Sand, and Dominik Henrich. Sparse and precise reconstruction of static obstacles for real-time path planning in human-robot workspaces. 2018. URL https://ieeexplore.ieee.org/document/8470629.

[42] Chris Holmberg Bahnsen. Advanced robotic perception, stereo vision: Two-view geometry in practice. https://www.moodle.aau.dk/course/view.php?id=31235#section-6, 2019. Accessed: 27-05-2021.

[43] Rikke Gade. Advanced robotic perception, pose estimation and 3d. https://www.moodle.aau.dk/course/view.php?id=31235#section-11, 2019. Accessed: 23-03-2021.

[44] Soulaiman Hazzat, Abderrahim Saaidi, and Khalid Satori. Multi-view passive 3d reconstruction: Comparison and evaluation of three techniques and a new method for 3d object reconstruction. *International Conference on Next Generation Networks and Services, NGNS*, pages 194–201, 12 2014.

[45] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. 2006. URL https://diglib.eg.org/xmlui/bitstream/handle/10.2312/SPBG.SPBG06.111-120/111-120.pdf?sequence=1.

[46] Chris Holmberg Bahnsen. Advanced robotic perception, camera calibration quick brush-up. https://www.moodle.aau.dk/course/view.php?id=31235#section-12, 2019. Accessed: 26-05-2020.

[47] Guilherme Mateus and Jacob Sørensen. An impedance control driven robotic disassembly platform using deep-learning for the detection of components. Technical report, Aalborg University, Aalborg, DK, Semester Project, 2020.

[48] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[49] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *International Conference on Next Generation Networks and Services, NGNS*, 02 2021. URL `http://arxiv.org/abs/2011.08036`.

[50] Liang-Chieh Chen, Zhu Yukun, Papandreou George, Schroff Florian, and Adam Hartwig. Encoder-decoder with atrous separable convolution for semantic image segmentation. Aug 2018. URL `\url{http://arxiv.org/abs/1802.02611}`.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[52] jfzhang95. pytorch-deeplab-xception, 2017. URL `https://github.com/jfzhang95/pytorch-deeplab-xception`.

[53] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. Dec 2015. URL `\url{http://arxiv.org/abs/1512.03385}`.

[54] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, 02 2010. ISSN 0920-5691. URL `https://doi.org/10.1007/s11263-009-0275-4`.

[55] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. 2017.

[56] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, and Yue Sun. Resnest: Split-attention networks. Apr 2020. URL `\url{http://arxiv.org/abs/2004.08955}`.

[57] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[58] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolomark. `https://github.com/AlexeyAB/Yolo_mark`, 2018. Accessed: 03/12/2020.

[59] Kentaro Wada. labelme: Image Polygonal Annotation with Python. `https://github.com/wkentaro/labelme`, 2016.

[60] David Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Mach. Learn. Technol. 2*, 2008. URL https://www.researchgate.net/publication/228529307_Evaluation_From_Precision_Recall_and_F-Factor_to_ROC_Informedness_Markedness_Correlation.

[61] Steven M. Beitzel, Eric C. Jensen, and Ophir Frieder. Map. *In Encyclopedia of Database Systems, edited by LING LIU and M. TAMER ÖZSU*, page 1691–1692, 2009. URL https://doi.org/10.1007/978-0-387-39940-9_492.

[62] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *The International Journal of Advanced Manufacturing Technology*, 2015. URL http://arxiv.org/abs/1411.4038.

[63] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. 2019. URL http://arxiv.org/abs/1904.11492.

[64] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL https://doi.org/10.1109/CVPR42600.2020.01161.

[65] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[66] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. 2020. URL http://arxiv.org/abs/2012.07177.

[67] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021. URL http://arxiv.org/abs/2103.14030.

[68] X. Liang, H. Zhou, and E. Xing. Dynamic-structured semantic prop-agation network. *CVPR*, 2018. URL https://arxiv.org/abs/1803.06067.

[69] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/papers/Ding_Semantic_Correlation_Promoted_Shape-Variant_Context_for_Segmentation_CVPR_2019_paper.pdf.

[70] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. 2019. URL http://arxiv.org/abs/1907.13426.

[71] Xia Li, Yibo Yang, Qijie Zhao, Tiancheng Shen, Zhouchen Lin, and Hong Liu. Spatial pyramid based graph reasoning for semantic segmentation. 2020. URL http://arxiv.org/abs/2003.10211.

[72] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation. 2021. URL http://arxiv.org/abs/1909.11065.

[73] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. 2019. URL http://arxiv.org/abs/1809.02983.

[74] Ye Huang, Wenjing Jia, Xiangjian He, Liu Liu, Yuxin Li, and Dacheng Tao. Channelized axial attention for semantic segmentation. 2021. URL http://arxiv.org/abs/2101.07434.

A Quantitative Experiments Data

A.1 Values of mAP at Different IoU Thresholds in Custom Dataset

IoU Threshold	mAP
@0.95	0.51
@0.90	0.32
@0.85	0.75
@0.80	0.85
@0.75	0.91
@0.70	0.93
@0.65	0.95
@0.60	0.95
@0.55	0.95
@0.50	0.95
@0.45	0.95
@0.40	0.95
@0.35	0.95
@0.30	0.95
@0.25	0.95
@0.20	0.95
@0.15	0.95
@0.10	0.95
@0.05	0.95

Table A.1: mAP values of the model in different IoU metrics.

A.2 Values of Precision-Recall at Different Confidence Thresholds in Custom Dataset

Confidence Threshold	Precision	Recall
@0.95	0.99	0.87
@0.90	0.99	0.87
@0.85	0.98	0.88
@0.80	0.98	0.88
@0.75	0.98	0.89
@0.70	0.98	0.89
@0.65	0.98	0.89
@0.60	0.98	0.90
@0.55	0.98	0.90
@0.50	0.98	0.91
@0.45	0.98	0.91
@0.40	0.98	0.91
@0.35	0.98	0.91
@0.30	0.98	0.91
@0.25	0.98	0.91
@0.20	0.97	0.91
@0.15	0.96	0.92
@0.10	0.95	0.93
@0.05	0.93	0.93

Table A.2: Precision-recall data at different confidence thresholds.

A.3 Values of mAP at Different IoU Thresholds in COCO Mini-val Dataset

IoU Threshold	mAP
@0.95	0.01
@0.90	0.12
@0.85	0.26
@0.80	0.37
@0.75	0.45
@0.70	0.52
@0.65	0.57
@0.60	0.61
@0.55	0.65
@0.50	0.67

Table A.3: mAP values of the model at IoU=.5:.05:.95

B Figure From Previous Works

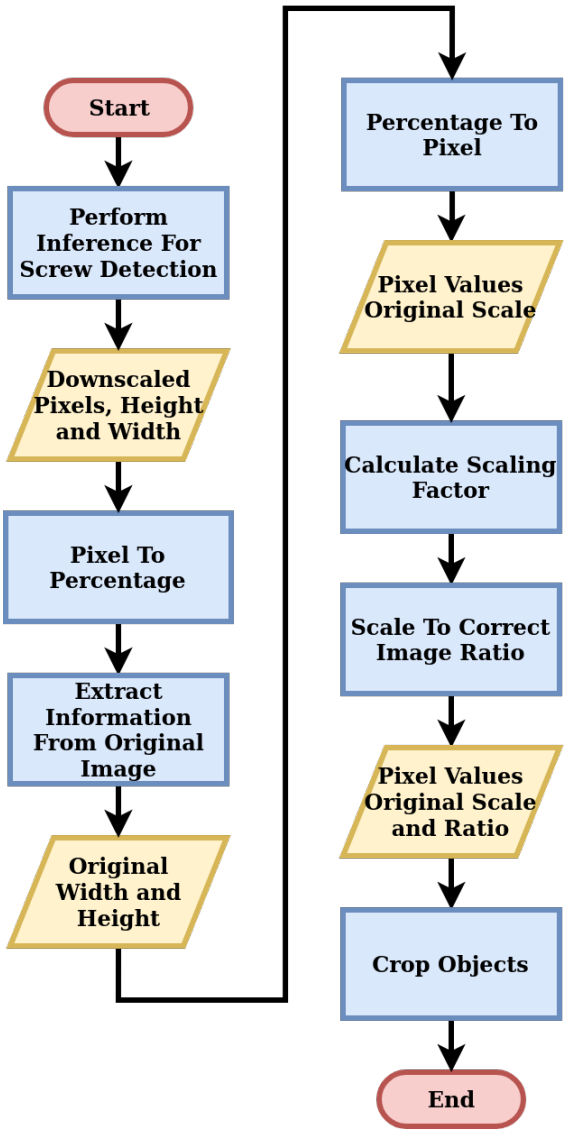


Figure B.1: Flowchart presenting the information flow when re-scaling the bounding boxes of the detected objects to the original image scale and correcting the ratio, while also cropping the objects.[47]