Horizon 2020
European Union funding
for Research & Innovation

An Intelligent Visual Analysis Scheme for Automatic Disassembly Processes in the Recycling Industry

Dissertation
for the award of the degree

"Doctor rerum naturalium"
(Dr. rer. nat.)

of the Georg-August University of Göttingen

within the doctoral program "PCS"
of the Georg-August University School of Science (GAUSS)

submitted by
Erenus Yildiz

from Istanbul, Turkey
Göttingen, 2021

Thesis Committee

Prof. Dr. Florentin Wörgötter
Third Institute of Physics/Biophysics, Georg-August University of Göttingen

Prof. Dr. Babette Dellen
Mathematik und Technik/Mathematics in Life Sciences, Koblenz University of Applied Sciences

Dr. Minija Tamosiunaite
Third Institute of Physics/Biophysics, Georg-August University of Göttingen

Members of the Examination Board

Prof. Dr. Florentin Wörgötter
Third Institute of Physics/Biophysics, Georg-August University of Göttingen

Prof. Dr. Babette Dellen
Mathematik und Technik/Mathematics in Life Sciences, Koblenz University of Applied Sciences

Prof. Dr. Jens Grabowski
Institute of Computer Science/Software Engineering for Distributed Systems, Georg-August University of Göttingen

Prof. Dr. Ramin Yahyapour
Institute of Computer Science/Managing Director, Georg-August University of Göttingen

Prof. Dr. Alexander Ecker
Institute of Computer Science/Neural Data Science, Georg-August University of Göttingen

Prof. Dr. Carsten Damm
Institute of Computer Science/Theoretical Computer Science and Algorithmic Methods, Georg-August University of Göttingen

Date of Oral Examination: 20.04.2021

Acknowledgments

This work has been produced as a part of friendly cooperation of roboticists from various parts of the world partially during the Covid-19 outbreak, and thus, first and foremost, I would like to thank the people and the academia of the participating countries, and organizations/sponsors who supplied this project. In particular, I would like to thank my supervisor throughout the study, Prof. Dr. Florentin Wörgötter, and our neuroscience group as well as the fellow roboticists from the cooperating universities in the project. My sincere thanks to all. My dear colleague Tomas Kulvicus and his efforts in the spin-off work of the project must be specifically acknowledged, as his supervision and experience led to a successful implementation. Next, I would also like to express my gratitude to the undergraduate students of the neuroscience group of the biophysics department at the Georg-August University of Göttingen who contributed to the project as part of their own theses work.

My sincere thanks go to my friends, many from totally different study backgrounds yet working towards parallel goals in life, and with whom I spent a few enjoyable years in the city of Göttingen between 2017 and 2021. One of these friends, namely Dean Comyn, has helped me with proofreading of this thesis, for which I could not thank enough.

Last but not least, I would like to express my utmost gratitude and appreciation to my family who has always motivated me to achieve accomplishments throughout my study life, including this Ph.D. thesis

Declaration

"I hereby declare that:

1. the opportunity to work on the aforementioned doctoral thesis project was not arranged commercially. Especially, I did not engage any organization which searches for doctoral thesis supervisors or which will entirely or partly carry out my examination duties against payment;

2. I have only accepted and will only accept the assistance of third parties in so far as it is scientifically justifiable and acceptable in regards to the examination regulations. Especially, all parts of the dissertation will be written by myself; I have not accepted and will not accept impermissible help from other parties neither for money nor for free;

3. I will observe the regulations of the Georg-August-University Göttingen for ensuring good scientific practice;

4. I have not applied for corresponding doctoral degree procedures at any other university in Germany or abroad; the submitted doctoral thesis or parts thereof were not used in another doctoral degree procedure.

I am aware that incorrect information precludes the admission to doctoral studies or will later on lead to the discontinuation of the doctoral degree procedures or to the revocation of the doctoral degree."

Göttingen, 27.02.2021
Place, Date

Erenus Yildiz

Abstract

This thesis aims to develop and deploy a visually intelligent disassembly scheme to automate the recycling routines for end-of-life products. The recent developments in artificial intelligence and computer vision are yet to be utilized in the E-Waste recycling industry, a shortcoming this thesis addresses. We ask to what extent and in what ways state-of-the-art deep learning methods could constitute an intelligent and generalizing scheme that can be used to disassemble commonly found computer parts such as hard drives and graphical processing units, that are known to contain valuable metals.

Using relevant metrics to evaluate the accuracy and performance of individual components and the entire system altogether, we empirically show that methods based on deep learning and computer vision are well suited for estimating the state of disassembly and inferring the required visual parameters for possible action executions.

The significance of this study is that it introduces an industry-oriented scheme that only requires off-the-shelf sensors to operate, and can be repurposed to work with new products. The work has also been part of Horizon 2020 project "IMAGINE", aimed to develop a fully automated disassembly robot to be used in recycling plants. Therefore, the results obtained and presented in this thesis have also been used in the IMAGINE project.

Contents

List of Figures

7 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

8 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

List of Tables

1 Introduction

1.1 Thesis Scope

In the century we are living in, a few concepts such as *sustainability*, *circular economy* and *re-usability* have become increasingly important. In fact, these concepts sometime play a determining role in politics of many states in the world. A very clear example to this can be given as the Millennium Development Goals Report 2015 [Mot+15] accepted in the UN Summit in New York. Items such as "Popularization of sustainable production and consumption models" and "Revival of global efforts and partnerships to ensure sustainable development" are listed and discussed under environmental sustainability. A lot of states today already report that they have been working towards integrating these concepts into national development policies, while some others are just starting to do so. It appears the world we are living in is slowly heading towards adopting these policies all around. In short, our planet - the common ground for all the states - will breathe easier due to us acknowledging we humans are an integral part of it, rather than the absolute owner of it.

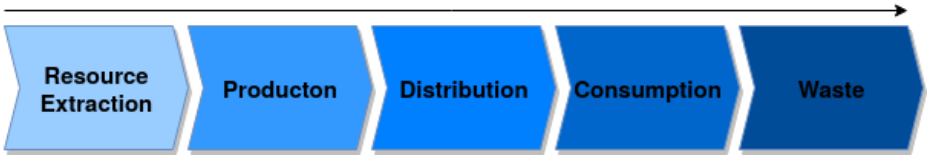

Figure 1.1: Major operational blocks and their relations under linear economy.

Sustainability and circular economy are tightly coupled concepts. Although many may not know the difference between a linear economy and a circular one, the differences are easy to grasp. As we know, in 1698 Thomas Savery invented the steam engine and it changed everything. This invention triggered the industrial revolution, which transformed our ability to manufacture things. Back then, raw materials and energy seemed infinite,

and labour was readily available. For the first time in history, we *mass produced* goods. Since then, the rapid pace of technological advancement has continued. The resulting innovations led to the fact that now many have access to products from all over the world at affordable prices. These products have brought many of us the material comfort previous generations could not have imagined. However, the current system is recently acknowledged to be not an advantageous one. The primary reason for that is the way we run our economy. Basically, we extract resources from the ground to manufacture products, which we use, and, when we no longer need them, throw them away. In other words, we have been following the *take-make-waste* routine. This is called linear economy, as illustrated in Fig. 1.1. Acknowledged by the aforementioned UN report, eventually, we must transform all the elements of the take-make-waste routine. This includes how we administer resources, how we manufacture and use products, and what we do with the materials subsequently. Only then can humans create a flourishing economy that benefits everyone and respects the limits of our planet. Shifting this routine involves everyone and everything: businesses, governments, and individuals; our cities, our products, our jobs and our consumption. By designing out waste and pollution, keeping products and materials in use, and regenerating natural systems we can reinvent everything. This is called a *circular economy* (illustrated in Fig. 1.2).

This model of economy revolves around the very concept of recycling. Many developed countries today are incorporating some of the aspects of circular economy (e.g., glass waste is used to reproduce glass, paper waste is used to reproduce paper). Ideally, the same is expected to apply to other waste categories, such as electronic waste or e-waste. Before going any further, one must define the term of e-waste. Encyclopedia Britannica[15] defines the term as follows:

"E-Waste is various forms of electric and electronic equipment that have ceased to be of value to their users or no longer satisfy their original purpose. Electronic waste (e-waste) products have exhausted their utility value through either redundancy, replacement, or breakage and include both 'white goods' such as refrigerators, washing machines, and microwaves and 'brown goods' such as televisions, radios, computers, and cell phones.

15 Electronic waste, Encyclopedia Britannica, 26 May 2016. Accessed 23 February 2021 https://www.britannica.com/technology/electronic-waste

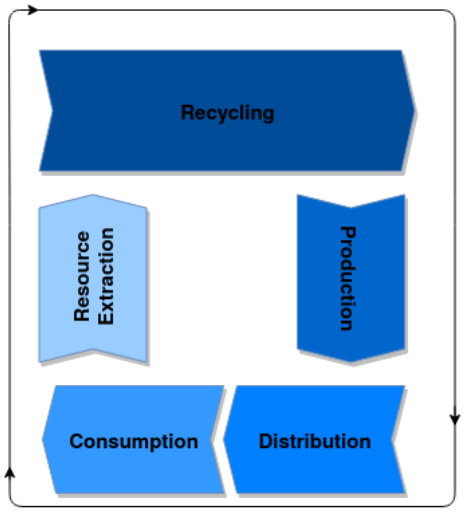

Figure 1.2: Major operational blocks and their relations under circular economy.

Given that the information and technology revolution has exponentially increased the use of new electronic equipment, it has also produced growing volumes of obsolete products; e-waste is one of the fastest-growing waste streams. Although e-waste contains complex combinations of highly toxic substances that pose a danger to health and the environment, many of the products also contain recoverable precious materials, making it a different kind of waste compared with traditional municipal waste" (Encyclopedia Britannica, 2016, "Electronic waste").

Electronic products of the last decades (e.g., mobile phones, computers and televisions) contain heavy metals such as Mercury or Beryllium. These elements, if exposed directly to humans, have the potential to cause cancer [Yan+20]. One of the places on Earth where this potential is clearly noticeable is the city of Agbogloshie, Ghana. The city is referred to as a "digital dumping ground". Currently, up to 80 tons of e-waste per month [16], from places like the USA, UK, EU and Australia, is legally and illegally

16 Morgan, T. (2014, September 18). Agbogbloshie: Welcome to the world's digital dumping ground (part 1). Retrieved February 23, 2021, from https://www.earthtouchnews.com/conservation/human-impact/agbogbloshie-welcome-to-the-worlds-digital-dumping-ground-part-1/

transported and dumped there. The hazardous impact is amplified by the fact that there are little-to-no skilled recycling personnel working in the city's dumpsites, and even more so with child labor in the picture. The cancerous elements leaking into the soil and afterwards into the drinking water makes everything terribly worse for the locals. In other words, e-waste dumping is causing an environmental and health crisis in Africa.

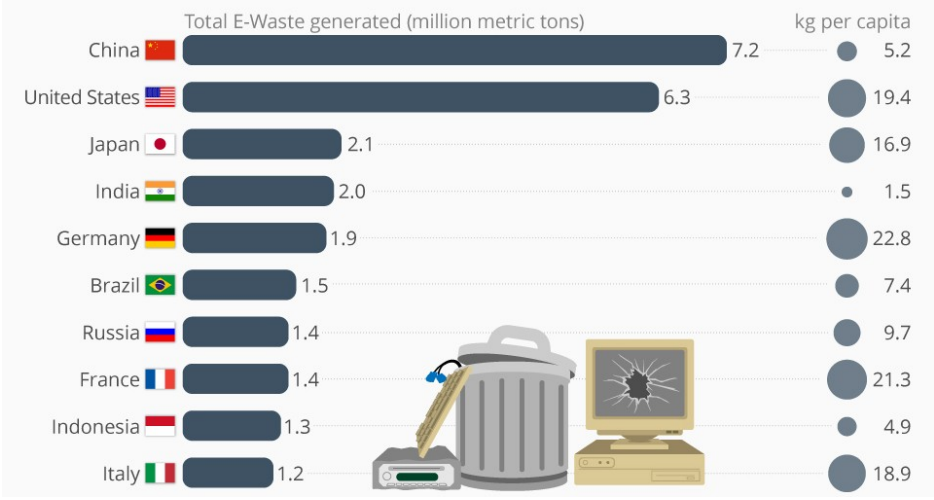

Figure 1.3: Top 10 countries by the amount of e-waste generated in the year of 2016, according to the Global E-Waste Monitor 2017 [Bla+17].

Given the critical conditions in certain locations on Earth, it is necessary to underline significant statistics regarding e-waste generation. It is not always possible to reach the data of every year at every time, therefore, we may demonstrate the data from an acceptable year that is not too far from the year we are in, since the figures usually do not drastically differ between a couple of years. As an example, as it's illustrated in Fig. 1.3, the 2016 data [17]. regarding the generation of e-waste around the world can be looked into. The statistics may not surprise us, after all, major economies with increased populations are expected to use and discard

17 Richter, F. (2017, December 14). Infographic: These countries generate the most electronic waste. Retrieved February 23, 2021, from https://www.statista.com/chart/2283/electronic-waste/

more electronics than others. However, when looked at the individual e-waste generation, the statistics [18] differ. As Fig. 1.4 illustrates, countries with relatively higher GDP per capita, seem to appear on top considering individual e-waste generation as well.

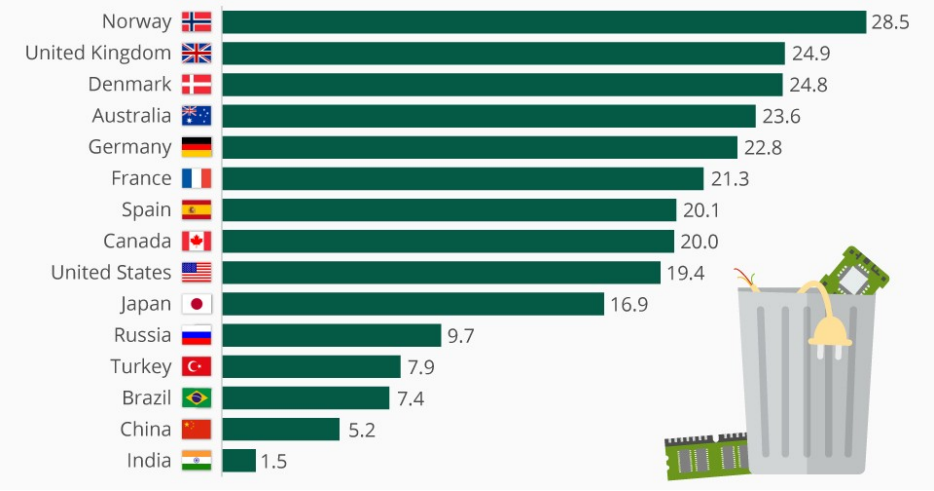

Figure 1.4: Top 15 countries by the amount of e-waste generated per inhabitant, according to the OECD report published in 2019[19].

Environmental and health issues are not the only reasons to take e-waste generation seriously. There is also the financial aspect to be considered. An electronic device is composed of certain parts. The most valuable parts are the ones to include precious metals such as gold (Au), silver (Ag), platinium (Pt), palladium (Pd), followed by less precious tin (Tn) and copper (Cu). Table 1.1 illustrates the kilogram value of these elements (as of early November, 2020). Palladium, gold and platinum seem to have high value in the market compared to the rest of the metals. The worth of these metals should also be observed over the span of the past decade to get a

18 McCarthy, N., Richter, F. (2019, February 28). Infographic: The world's worst electronic waste offenders. Retrieved February 23, 2021, from https://www.statista.com/chart/17175/e-waste-generated-per-inhabitant-in-selected-countries/

19 Siegerink, V. (n.d.). How's life in the digital age? Retrieved February 23, 2021, from http://www.oecd.org/publications/how-s-life-in-the-digital-age-9789264311800-en.htm

clearer picture on the value. Fig. 7.2 and Fig. 7.1 in Appendix 7.1 shed some light into the matter by illustrating the price per Kilogram for these metals over the decade. One could conclude that gold and palladium have been experiencing a clear rise in the last half decade, whereas platinum has been dropping, even though retaining a still high value in the market.

Figure 1.5: Gold ore to be processed in order to acquire pure gold, found in Larder Lake, Ontario, Canada [20].

It must be noted that the extracted metal elements from the e-waste parts are mostly pure, unlike the ones mined in metal mines. Taking gold as an example, once mined as ore, the gold inside is still to be processed. However, if it is extracted from a circuit board of an e-waste device, it can be directly smelted into a gold bar and delivered to the market or industry. Fig. 1.5 illustrates gold ore that is still to be processed. When it comes to gold extraction from e-waste parts, the WEF (World Economic Forum) report [UIU19] states that there is 100 times more gold in a tonne of discarded mobile phones than in a tonne of gold ore. Mobile phones are clearly not the only e-waste devices with rich gold content. Computers, especially their microprocessors, contain a fair amount of gold, as illustrated

20 John, J. (2014, July 23). Hydrothermal GOLD-QUARTZ vein IN Fuchsitic Metamorphite (LARDER Lake Gold Ore, Ontario, Canada). Retrieved February 23, 2021, from https://www.flickr.com/photos/jsjgeology/14722527472/

in Fig. 1.6. Followed by RAM (Random Access Memory) and motherboard, computers are clearly valuable e-waste devices.

On the other hand, it is definitely not a trivial process to extract these precious metals from e-waste devices. First of all, any e-waste device has to be disassembled into parts. The disassembled parts should be categorized further and those with high precious metal content (e.g., circuit boards) are fed into a chemical pipeline involving various blocks (depending on the industrial application), such as cyanide solutions, acid treatments, electrolysis and even bio-leeching [Mad+15]. These techniques, however, won't be explained further, since extracting metals from circuit boards through chemical routines is beyond the scope of this thesis. This thesis focuses on the first stage in the aforementioned process: the disassembly of the device.

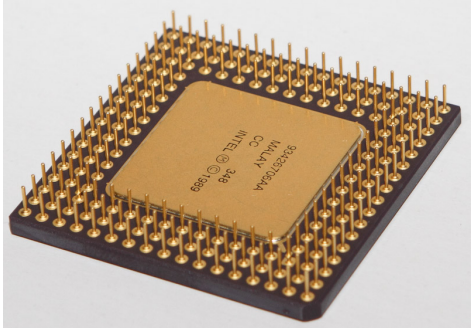

Figure 1.6: Underside of an Intel 486 Processor with golden plate in the middle, and golden pins around of it[21].

Disassembly of e-waste devices is usually carried out by human operators working on various e-waste devices in recycling plants. Operators usually wear protective clothing and face masks in order to be isolated from hazardous gases that could be released during the disassembly. As human life gains value in developed countries, there are efforts to automate the disassembly processes of e-waste devices. This is in perfect parallel with the

21 How much gold is in a Computer? (n.d.). Retrieved February 23, 2021, from http://therefiningcompany.com/How-Much-Gold-Is-In-A-PC.aspx.

22 Daily metal spot Prices copper Price. (n.d.). Retrieved November 02, 2020, from https://www.dailymetalprice.com/metalprices.php.

Metal	Price($) per Kilogram(kg)
Palladium	72532.08
Gold	60308.37
Platinum	27553.19
Silver	759.56
Tin	17.650
Copper	6.6945

Table 1.1: Kilogram value of the precious metals found in e-waste devices, by 02.11.2020[22].

aforementioned environmental and financial aspects of e-waste recycling. As a matter of fact, using automated systems and/or robots instead of human labor could even be more efficient in the long run, since human labor brings aspects such as taxation, insurance and healthcare into the picture. It is therefore a worthwhile research subject to assess the capability of automated systems in e-waste recycling domain.

There has been an attempt to address the desired assessment. The European Union has been funding the "IMAGINE" project[23] under the Horizon 2020 research framework[24]. The project aims to develop a robotic system -guided by means of artificial intelligence- to automatically disassemble a given e-waste device, such as a computer hard drive. The project is an international effort, thus the consortium is formed by multiple partner universities from countries such as Germany, France, Spain, Turkey and Austria. Along with the universities involved, a recycling company called *Electrocycling GmbH* [25] is also a part of the consortium. The developed system is to be tested with an actual computer hard drive taken from the respective e-waste heap at Electrocycling.

23 https://www.imagine-h2020.eu/
24 https://ec.europa.eu/programmes/horizon2020/en
25 http://www.electrocycling.de/en
26 https://www.imagine-h2020.eu/

Figure 1.7: IMAGINE consortium consists of multiple partner universities and Electrocycling GmbH [26].

Figure 1.8: IMAGINE project has received funding from the European Union's Horizon 2020 Research and Innovation programme (H2020-ICT-2016-1, grant agreement number 731761).

1.2 Thesis Structure

This thesis is organized into seven chapters divided into three main parts. First, the introduction and the literature review are found in Chapter 1 and Chapter 2. The second part describes an overview of the proposed scheme, background and implementation of the methods used in the proposed scheme in Chapter Chapter 3, Chapter Chapter 4 and Chapter Chapter 5. Finally, a quantitative evaluation of the proposed visual scheme, as well as the conclusion, are presented in Chapter 6 and Chapter 7. Additionally, large images, tables and relevant side-information are found in Appendices 7.1 and 7.1. An illustration to address this thesis organization is presented in Fig. 1.9.

Chapter 1: **Introduction**, presents an overall introduction of this research, including the research motivation, scope and structure of this thesis.

Chapter 2: **Related Work** provides an overview of the existing research

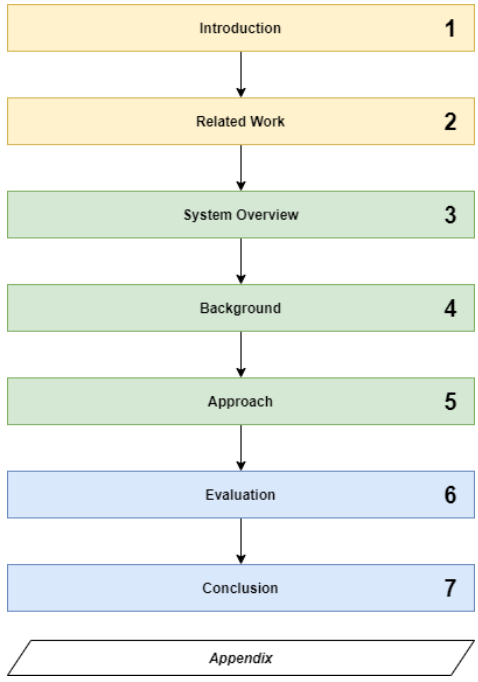

Figure 1.9: Organized thesis structure. Recommended to follow throughout for a complete and better understanding.

in the relevant areas, namely disassembly automation and vision-aided disassembly.

Chapter 3: **System Overview** introduces the IMAGINE framework on which the proposed visual scheme is built. This high-level description of the entire system is necessary in order to understand the asynchronous interaction and low-level information exchange between the processing blocks in the system.

Chapter 4: **Background** provides significant knowledge with respect to the development of the visual disassembly scheme. This in-depth look includes a theoretical introduction to the deep convolutional neural networks (DCNNs), classical computer vision methods (such as Hough Transform), as well as stereo imagery and point cloud technology.

Chapter 5: **Approach** explains the design and functionality of the individual components the proposed visual scheme has. It also underlines how

these individual components interact with each other and where they stand in the proposed pipeline, as well as the entire IMAGINE framework.
Chapter 6: **Evaluation** quantitatively evaluates each individual component mentioned in the previous chapter using relevant metrics specified for each individual vision task. It also assesses the feasibility of generalization, i.e., which other devices the proposal visual scheme could be used in order to achieve automated disassembly.
Chapter 7: **Conclusion**, summarizes the entire work, underlines the pros and cons of employing such a system in a potential recycling plant. It discusses the possible implications and improvements the proposed system could have, and what more could be done in order to optimize it in the future.

2 Related Work

This chapter presents a review of related literature required to develop a visual intelligence scheme for a robotic disassembly automation. The content is divided into four sections describing related subjects: overview of disassembly, types of disassembly in terms of automation, the role of vision in the disassembly domain, and the case-study product (LCD screen disassembly). Conclusions and problem statement are shared at the end.

2.1 An Overview of Product Disassembly

2.1.1 End-of-Life Product Treatment

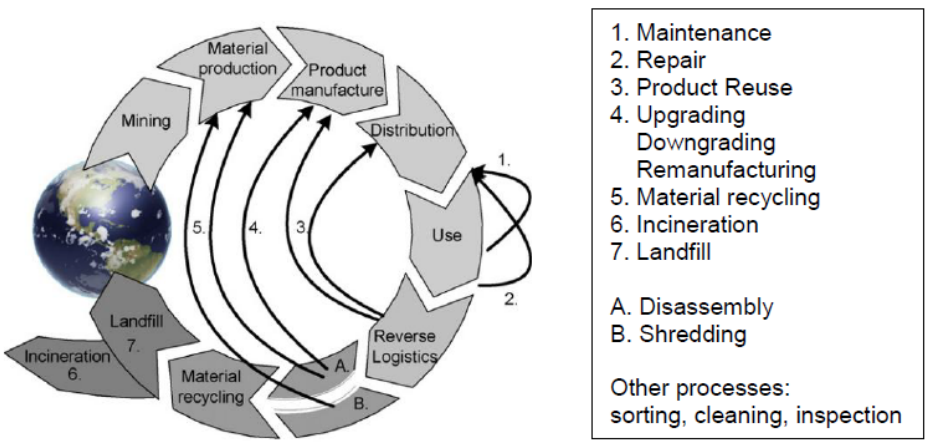

Figure 2.1: Scenario of End-of-Life products [Duf+08].

The scenario for the End-of-Life products is illustrated in Figure 2.1. Life Cycle Assessment (LCA) investigates the environmental impacts of the investigated objects, systems or products from exploration and supply of materials and fuels, through production and operation, to their disposal/recycling [Peh06]. This investigation is carried out on EOL (End

	Refurbishing	Re-manufacture	Reuse	Recycle	Incinerate/Landfill
Unprocessed products	X			X	X
Modules	X	X		X	X
Components			X	X	X
Damaged components				X	X
Waste					X

Table 2.1: Destination of output of a disassembly facility [LG04].

Of Life) products in order to identify proper treatment options in regard to environmental and economic perspectives. To continue on with the aforementioned EOL options, the products first must be disintegrated into individual components, parts, or material according to the requirement of each treatment process. The disintegration process is typically performed in two ways: shredding and disassembly. *Shredding* is a destructive process that roughly breaks the products into small pieces or particles that will be supplied to the recycling process [VC15]. The outcome of the shredding process is low-quality blends of material and requires further sorting to separate the valuable materials from the waste. The outcome shredded pieces or particles are physically sorted by density and magnetic property using a number of techniques, i.e. magnetic, electrostatic, etc. Shredding is commonly implemented in industry practice due to the low operating cost [VC15]. However, a major disadvantage is the loss of value of the parts and components that turned out as shredded pieces. In addition, hazardous components that potentially contaminate the workplace are problematic [LG04]. *Disassembly* systematically separates the product into its constituent parts, sub-assemblies, or other groupings [KRK07]. The detached components can be supplied to a wide range of treatment processes according to the desired purpose and downstream conditions as shown in Table 2.1. It serves not only the treatment of EOL products treatment but also repair-maintenance purposes if the proper techniques are applied. A major problem is the usually high operating cost, which can exceed the value recovered from EOL products. Therefore, if it becomes economically infeasible, it is usually avoided in industry practice. The possible solutions to make the disassembly process economically feasible are discussed in the following sections.

	Generated E-Waste (Mt)	Recycled E-Waste (Mt)	Generated/Recycled (%)
Asia	24.9	2.9	11.7
Americas	13.1	1.2	9.4
Europe	12	5.1	42.5
Africa	2.9	0.03	0.9
Oceania	0.7	0.06	8.8

Table 2.2: E-Waste generation and recycling per continent, 2019 [For+20].

2.1.2 Disassembly of Products

The EOL treatment process has become more concerning due to a large number of the products disposed. In the field of life cycle engineering, much research focuses on the products having high material-return rate, short life-cycle, and high volume of waste, such as electronic and electrical waste and EOL vehicles. According to the European Directive 2002/96/EC [Ele03], Waste Electrical and Electronic Equipment (WEEE) covers a wide range of electrical and electronic products categorised into 10 groups (e.g., household appliances, IT & telecommunication, consumer equipment, and electrical and electronic tools, etc.).

In 2005, the amount of e-waste in Europe was around 8.3 - 9.1 million tonnes per year consisting of 40% for large-household appliances and 25% for medium-household appliances [Ore+01]. This figure was then expected to grow 2.5 - 2.7% every year [Hui08]. In 2019, the number of WEEE in Europe was around 12 million tons per year as shown in Table 2.2. According to the latest report [For+20], Asia is the continent that generates the most e-waste, whereas Europe is the one that recycles the most. It is worth to mention that, globally, only 17.4% of e-waste is documented to be formally collected and recycled [For+20]. It is also significant to state that the number of e-waste is 8% of the whole municipal solid waste (MSW) worldwide [BPB07]. These numbers should be kept in mind with the bitter fact that every year the life cycles of electronic products are slightly decreasing [SSP00]. More than 800.000 tonnes of electrical and electronic waste were collected in Germany in 2017, 90% of it came from households. This is the equivalent of 9.13 kilograms per capita and year [Bal+15]. The remaining 10% came from businesses. Regarding vehicles waste; in 2006, around 230 million cars were in use in Europe, and 10.5 million tonnes are disposed of every year. According to the European

Directive 2000/53/EC, at least 80% (by mass) of EOL vehicles must be reused and recovered; meanwhile, 85% of them must be recycled [Vig+10].

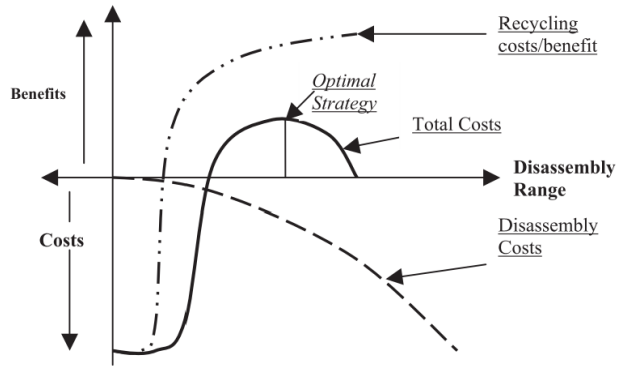

Figure 2.2: Determination of optimal disassembly strategy [DM03].

Since the disassembly of products is one of the key steps of the efficient EOL treatments, a number of investigations have been conducted in regard to the environmental and economical aspects [Che01; GG99; SW95]. The disassembly approach is also compared with the conventional waste treatment, i.e. disposal and landfill [Ore+01]. These investigations concluded that a disassembly approach greatly benefits the environment but it is not economically feasible due to the expense resulting from both direct costs (e.g., labour and machine) and costs (e.g., stock and logistics). However, the disassembly process can be economically feasible if the optimal disassembly strategy with respect to the cost and benefit is implemented as illustrated in Figure 2.2. Therefore, the research direction currently focuses on developing the strategy for the economically feasible disassembly.

One of the works [Duf+08] states that the factors influencing profitability of the disassembly process are analysed by applying Principal Component Analysis method (PCA) on the case study of disassembly activities. According to the findings, profitability is related to three factors: completeness of disassembly, EOL facility, and usage of automation. In short, the depth of disassembly is increased if the process is performed by relevant manufacturers. A high degree of automation positively affects profitability, and the slightly high investment of an end-of-life facility is not in conflict with profitability.

2.2 Automated Disassembly

As human labor costs increase in developed countries, the manual work of disassembly is slowly giving way to automated systems. It is important to understand what degree of autonomy is feasible nowadays, and what is the most efficient process. To this end, two types of automation approaches are considered.

2.2.1 Semi-Automated Disassembly

A number of research works regarding the semi-automatic disassembly have been conducted. Semi-automatic disassembly (or Hybrid system) integrates human operators and an automatic disassembly workstation (e.g., robots with disassembly tools) in order to improve the efficiency of the disassembly process. The majority of the process is carried out by machines that can operate automatically. The machines are used to improve efficiency and conduct hazardous tasks, e.g. heavy duty destructive operation. Consequently, human operators can focus on more sophisticated tasks rather than performing the operational work [Kno+02]. Another work [KHS07] states that the hybrid system is necessary in the flexible system in order to support various product families. Manual operation is involved when the automatic operations fail. A work [FKS06] from 2006 also states that the hybrid system allows achievement of the disassembly process economically but manual operation must be involved. The major drawback of automated systems is their instability due to a non-determined disassembly sequence and various product conditions.

In the aforementioned work [KHS07] a hybrid, flexible system was developed to disassemble a wide range of product families. The study focuses on the automatic generation of plans and the control sequence of a system consisting of three robot arms and conveyor belts. The robots are responsible for heavy duty tasks, such as plasma cutting of the sidewall of washing machines. "Before the process starts, the system evaluates the degree of autonomy of the overall task from the product information and availability of the system. Consequently, the tasks can be properly distributed to the manual or automatic workstations" (Vongbunyong, 2015, pp. 43). Two years later, the authors extended [KKS09] this concept to develop a disassembly line for LCD screens, as illustrated in Figure 2.3.

Screw removal and object handling operations are performed by Selective Compliant Articulated Robot Arms (SCARA) while manual operations perform the rest of the process.

Figure 2.3: Hybrid system for LCD screen disassembly, where humans and automation systems or robots cooperate to achieve the goal of disassembly [KKS09].

Another work [ZDK01] extends the semi-automatic disassembly cell with a modular concept by configuring the cell controller using hierarchical control and information distribution of the disassembly process. The system is designed for extracting embedded components from printed circuit boards (PCB). Regarding modular systems, the working components (e.g., robot arms, fixtures, and quality control system) are grouped as a subsystem. To supervise the communication and co-ordination tasks in and between each subsystem, the cell controller is used [VC15].

In conclusion, a major advantage of semi-automatic disassembly cells is the flexibility to deal with uncertainties and variation in the products. Economic feasibility can be achieved since the automatic workstation can perform the tasks more efficiently. Meanwhile, humans will get involved in the top-level control or when the automatic operations fail. However, the operation cost due to the human labour still exists in this concept. Economic feasibility may not be able to achieve especially in developed countries where the labour cost is very high. Therefore, this concept should

be developed to a human-free disassembly environment where a higher degree of autonomy is needed. The fully-automatic disassembly is explained in the following section.

2.2.2 Fully-Automated Disassembly

In comparison with the semi-automatic disassembly, a degree of autonomy is increased by incorporating sensor modules, prior knowledge of products, and a high-level task planner. The configuration of the system is similar in a number of studies, typically consisting of four components such as robot arms, vision system, disassembly and handling tools, and other optional sensors. A number of selected research works using complete disassembly cells are presented in this subsection. Moreover, since the vision system is typically used in fully-automatic disassembly, much research has been focused on this component, which is explained in the next subsection.

One of the most complicated and advanced disassembly cells for disassembling computers was developed in 2004 [Tor+04]. "The cell consists of two industrial articulated robots equipped with force/torque sensors and selected interchangeable disassembly tools. Both robots work co-operatively through a task planner automatically generating paths and trajectories based on a graph model proposed by the same main author" (Vonbunyong, 2015, pp.44). Another research [Gil+07] implemented the multi-sensorial system that combines information between a tactile sensor and the vision system in order to perform visual-servoing of the robot. The conceptual test was conducted by removing a bolt from a straight slot. Later research [Gil+06] focused on the vision system that detects partial occlusions of the components to simplify the disassembly task. The conceptual test was done in the detection of circuit boards. In conclusion, this system exhibited an ability to solve the uncertainty problem at the operational level by using an integrated sensor system. However, according to the high-level planning part, the disassembly sequence plan is based on precedence relations among assemblies [TPA03b]. This method can generate the plan automatically, but the user still needs to indicate the precedence relation and specific information of the product structure *a priori*. The input from the vision system is only to indicate the detailed geometry used for the operation level. There is no feedback information sent back to the higher level planner to acknowledge the current situation of the process. Therefore, it can be

concluded that the system is able to disassemble only a product for which the structure is known.

One study [Bük+01] developed a disassembly system for wrecked cars. This project targets the disassembly of wheels with variation in the size of the wheels, the number of bolts, and the position of the wheel [VC15]. Active stereo cameras are used to reconstruct the 3D structure of the product. "Principle Component Analysis (PCA) is used to determine the component that is potentially difficult to recognise due to the uncertainties in EOL condition (e.g., rusty)" (Vongbunyong, 2015, pp.46). In their related work [BH96], the authors suggested a knowledge-based approach with a neural network to solve the problem of occlusion in complicated scenes. In short, this research focused on increasing the flexibility of the vision system to deal with uncertainties in EOL condition.

Another notable effort [ElS+12] developed the disassembly system for EOL computers for reusing and recycling purposes. The system consists of an articulated industrial robot and a camera system [VC15]. The system deals with uncertainties by utilizing two components: the visual sensor and the online genetic algorithm. First, the visual sensor provides a 2.5D map of the detected component by integration between a 2D-camera with a laser range sensor. Template matching is used to recognise and locate the components according to the 2D template supplied in a BOM (Bill of Materials). Second, the operation sequence for removing the detected component is generated by an online GA as the process goes. As a result, the optimal and/or near-optimal sequence is generated by minimising travel distance and the number of disassembly method changes. In conclusion, the key feature of this system is the ability to adapt the plan according to the current situation obtained by the vision system [VC15]. However, a precise BOM that represents the product structure is needed to be supplied *a priori*. The online visual input is only used to identify the operation options, e.g. accessibility, for the components expected. The BOM cannot be modified even if the actual product structure is found inconsistent with the predefined one. This will limit the flexibility to deal with unknown model samples. In addition, the uncertainties at the operational level were not clearly explained in this article.

A comprehensive survey of existing approaches in 2006 [Wei+06] showed numerous limitations also found in later work. First, the focus on detecting

a specific class of component to detect – e.g. screws ([Bai02; BRP16; Uki07], bolts [Weg+15] – reduces the generality of the method and its applicability in the context of recycling where several types of components must be gathered. The use of fully model-based methods [ElS+12; Pom+04b; TN00; TPA03a; Uki07; Xie+08] implies the availability of models (e.g., Computer Aided Design), as explained in the above works. This particular assumption is a strong one, as these models are rarely available for every brand and every type of device. Moreover, the recycling domain does not guarantee that devices are in an intact state. Finally, another important limitation is the use of non-adaptive feature detection techniques [Bai02; BRP16; KJ00; Weg+15; Wei+06]. As mentioned previously, the high intra-class and inter-brand variances of components reduces the relevance of a human-engineered descriptor of a component class. This limitation is only partially explained by the date of publication of some works [Bai02; KJ00; Wei+06].

Furthermore, there have been some efforts that are worth mentioning briefly (as there were no significant achievements, yet they are indeed in the literature of automated disassembly). First, a work from 2006 [Mer+10] proposed an ontology-based architecture with a multi-agent system (MAS) for disassembly of digital cameras. The ontology is used to describe the benefit of each operating module on the current task. Hence, the optimised tool-path can be generated. Secondly, there is also a work [Bai06] that presented a strategy of real-time tool path generation and error recovery.

On one hand, it seems that there is still a substantial lack in generalizable, device- and environment-independent methods that can be used in disassembly processes. On the other hand, Deep Convolutional Neural Networks (DCNN) offer a powerful solution to analyze the inner structure of devices in the context of disassembly. These methods have the potential to solve the problems highlighted before: the specificity of a detector can be tackled by training on a dataset that includes samples of multiple parts of the targeted device (and extended by retraining later on) and thus being able to recognize these parts in all the devices that use them. The lack of adaptability is addressed by the nature of deep networks: as machine learning methods, they are problem and feature agnostic before training. Finally most of these methods are CAD model-free, meaning that they do not require device-specific description sheets. They learn the relevant

features of a part class by themselves, abstracting from the details of a specific brand and model. In E-Waste recycling context, devices found in the same family of devices often include similar parts such as PCBs, screws, wires, etc. Extracting features from these using visual paradigms usually leads to reproducible results that can be used and further developed for other devices as well. In the next subsections, such methods are looked into.

2.3 Vision Systems in Disassembly

This section focuses on the role and application of the vision system in automatic disassembly systems. In general, machine vision serves a large number of applications in robotics and autonomous machine research, such as the assembly process of electronic products, process control, and quality control [Low10]. From a case study of disassembly of used cars investigated by Tonko et al. (1999) [Ton+99], the authors summarise the vision-based constraints that are frequently encountered in the disassembly process. The constraints mentioned are the objects located in front of a complex background, partial occlusion, detection of rigid objects, 6-DOF (Six Degrees Of Freedom) estimation of objects, and low contrast image due to some covering mixture (e.g., oil and dirt conditions) [VC15]. The vision system must be able to handle these conditions under controlled conditions (e.g., uniform lighting) in this research. However, these constraints might be slightly changed in the disassembly of other products. In summary, the research conducted in the context of vision and disassembly falls into the following disciplines:

1. Object recognition and localisation.
2. Optical problem (e.g., distortion, shading, contrast, etc.)
3. Object and image quality (e.g., clutter, occlusion, etc.)
4. Camera configurations, frames mapping, and coordinate system.
5. Modelling and representation of objects.

With these disciplines taken into consideration, the literature could be classified in categories such as recognition and segmentation, detection and classification, configuration of camera and coordinate system, model representation. In addition, the computer vision library that is expected to be used for programming is also reviewed. As there are many libraries and frameworks available, this thesis only considers the libraries and frameworks used developing the proposed pipeline.

2.3.1 Camera Configuration

Cameras are one of the widely used sensors in the context of automated disassembly schemes. This is mainly due to the fact that visual input carries a fine level of detail about the objects present in the scene. An automated disassembly scheme may utilize a monocular or a stereo camera in order to capture the disassembly scene containing the object and, optionally, the surface plane as well as the manipulation tool. This subsection aims to introduce the early works regarding camera configurations.

In a related work mentioned before [Tor+04], two technical problems are pointed out. The optical problem (intrinsic) and the difference in coordinate systems (extrinsic). First, the optical problems involve the internal characteristics of cameras, e.g. focal length, distortion, and centre point. This problem is resolved by calibration of the internal parameters of the camera, e.g. model of the lenses. Second, the ambiguity from different coordinate systems can be resolved by transformations of the telescopic system (translation and rotation). The extrinsic parameters can be derived from the absolute positions of the working elements, e.g. robot base, robot configuration, camera, and worktable. Subsequently, the homogeneous matrix can be derived, representing the mapping between each coordinate system. With respect to the position control of the robot, visual-servoing is broadly used to locate the position and orientation of the robot arm in 3D. The information can be presented in the image-based (pixel) and the position-based (mm degree) approach. One study [Ton+99] suggests using a position-based approach because it represents more explicit knowledge for position control. Furthermore, the authors discuss and suggest camera-robot configurations (eye-hand-configurations) that allow multiple objects tracking.

More about the camera calibration is discussed in the Chapter 5, where

the proposed pipeline's camera configuration is justified, and the calibration procedure is explained in detail.

2.3.2 Recognition, Segmentation, Detection, Classification

The problem of disassembly scene analysis can be formulated as a problem where machine learning paradigms (e.g., segmentation, classification) and traditional data (e.g., pointcloud) analysis methods are used in order to recognize the present parts of the target device. In contrast to other application domains, the context of recycling E-Waste adds the following constraints:

- The position estimation of parts must be precise enough to allow manipulation of small parts in the device.
- There is a strong degree of occlusion because parts are intertwined (e.g., the platters and R/W head of a hard drive)
- There is substantial intra-class variance for specific parts depending on the device brand or model or the potential damage of the part.

Recognition vs. Segmentation

Object recognition serves purposes such as classifying between a product and a component, or detecting the desired component to be disassembled [Tor+04]. Given this definition and the above constraints, it is important to realize the fact that the mere recognition of a part without its physical boundary information is insufficient in the context of disassembly. Since any manipulation action is parameterized with the coordinate information, position estimation of parts is essential. There are many other domains where recognition alone does not suffice, and thus, there is another problem called *Semantic Segmentation* problem that needs to be addressed.

The *Semantic Segmentation* problem (also known as *pixel-wise classification problem*) has been addressed in domains ranging from robotics and autonomous driving, to medical research, agriculture and human-computer interaction [DHS16; Dvo+17; FSB19; Li+17; Lin+19; Liu+18]. In these

27 SemTorch, David Lacalle Castillo, Retrieved February 23, 2021, from https://pypi.org/project/SemTorch.

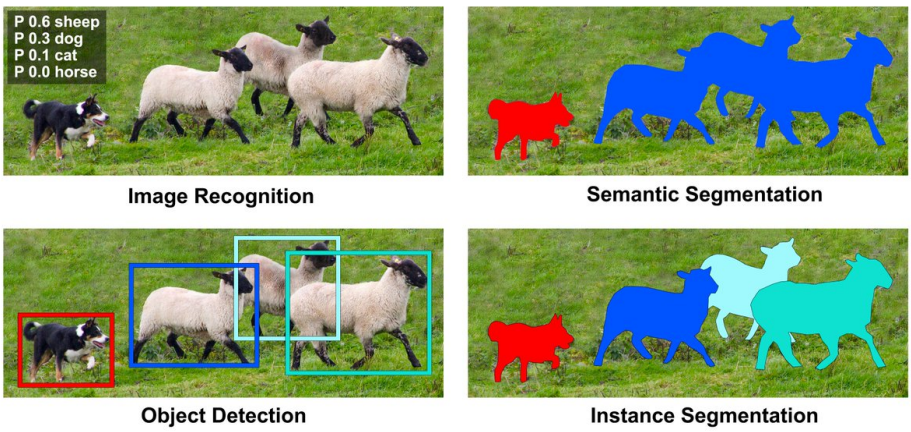

Figure 2.4: Differences of outputs in between common problems in computer vision. [27]

domains, the state-of-the-art performance is achieved using deep learning methods, especially Convolutional networks (CNNs or ConvNets) [KSH12]. Fully Convolutional Networks (FCN) have been the standard algorithm to achieve pixel-wise segmentation of images [BKC17; LSD15; RFB15b] in various domains such as medicine, autonomous driving, and domestic robotics. They extend CNN by replacing fully-connected layers by convolutional layers, allowing for arbitrary-sized input with no need for region-proposal. The trade-off however, due to the nature of the layers, is that the prediction of boundaries lacks precision. This problem is addressed by a set of improvement to the networks (e.g., multi-resolution architecture with skip connections [GF16; Lin+17a], mask refinement [Pin+16]). Additionally, pixel-wise segmentation requires labeled data which has a higher cost of production [Lin+14]. Weakly-supervised methods have been developed to tackle this issue [Zha+19]: these methods iterate between learning using coarse bounding boxes as labels then refining them into more precise masks [DHS15; Kho+17]. They achieve similar performance as fully supervised methods at a lower labelling cost. Another main approach is called Region-Based Semantic Segmentation. This method relies on a pre-processing of the input image into candidate objects (region proposals) for which features are extracted. These features are then used for the classification.

There is an entire family of R-CNN (Region Based Convolutional Neural Networks) [Gir+14; Gir15; Ren+15] that have been evolving. The current state-of-the-art of this family is called *Mask-RCNN* [He+17] which is based on Feature Pyramid Network (FPN) [Lin+17b]. Mask-RCNN has been used by Facebook AI Research [Jou+16] as well as by medicine studies [Che+19; Cou+19; Joh18].

There are situations where semantic segmentation may also not suffice. Tasks that require the awareness of object instances fall under a specific problem called the *Instance Segmentation* problem. Instance segmentation is similar to semantic segmentation. However it goes a bit deeper. It identifies the object instance each pixel belongs to. This way, even if there are objects of the same class, their instances could be distinguished, as illustrated in Fig. 2.4. Mask-RCNN and its ensembled models are currently the state of the art for object detection and instance segmentation. The reason for this is the region-based detection mechanism mentioned earlier. The core idea behind Mask-RCNN is to scan over the predefined regions called anchors. RPN (Region Proposal Network) does two different types of predictions for each anchor. First is the score of the anchor being foreground, and the bounding box regression. The fixed number of anchors with the highest score are then chosen, and the regression adjustment is applied to get the final proposal for object prediction at the network head. There are other architectures that could be used for this task, such as the ensembled model Cascade-RCNN [CV18]. There is also the recent work of Mobile-RCNN [How+17] that could be considered.

The most notable achievement in the context of segmentation of a disassembly scene came in 2019 with the work of Jahanian et al. [Jah+19]. The work claims to tackle the multi-faceted dense circuit problem by building upon state-of-the-art CNNs and end-to-end (no pre/post-processing) optimizing for multi-task learning of detection, localization, and instance semantic segmentation, while accounting for crisp boundaries of small components. Specifically, the method extends the Mask R-CNN [He+17] by addressing the inherent problems of CNNs: the trade-off between semantic top levels and detailed bottom levels. First, the authors explore the proposal feature by pooling feature maps from all levels (top and bottom) of the backbone while benefiting from lateral skip-connections between all levels. The key idea here apparently is to regain the higher spatial

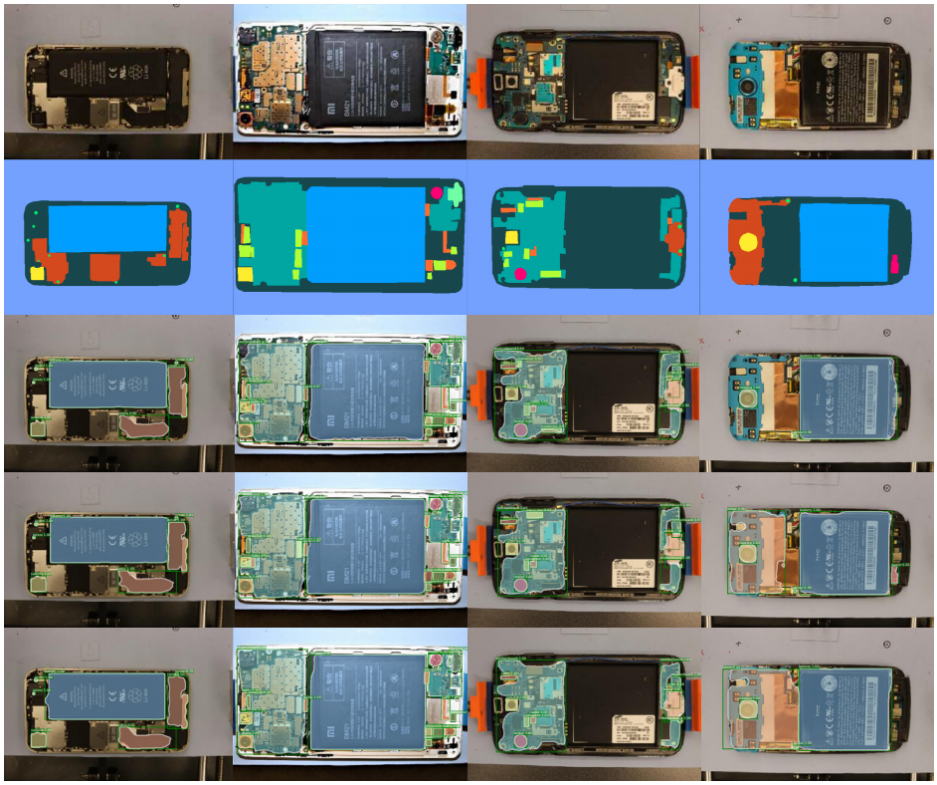

Figure 2.5: Results of the method proposed by Jahanian et al. [Jah+19]. From top, rows are: Input image, ground truth, Mask R-CNN + FPN, Mask R-CNN + FPN + Adaptive Pooling, and Mask R-CNN + FPN + Adaptive Pooling + Edge Detection. Note that the last two rows qualitatively have more accurate (respect to the baseline) boundaries detection for each object, and the last row has even better boundaries detection. Authors note that they intentionally picked images where they had several objects in the dense board, otherwise doing the multi-tasks on an image of a single component is much easier and less interesting.

resolution information of the objects' boundaries from the feature maps at lower-levels of the CNN hierarchy. Second, they add an auxiliary branch for predicting objects' boundaries on each Region of Interest (RoI), in parallel with the existing branch for classification, bounding box regression, and masks segmentation. Thereby, this recent work achieved increased

accuracy for the segmentation of the phone dataset by over 3% of the standard metric. More importantly, it retained the crisp boundaries for the robotic tasks (as compared to Mask-RCNN) while maintaining the same level of computing speed. Fig. 2.5 illustrates a series of images from the work.

There are, however, a few points to mention. The first point is the fact that the proposed work was only evaluated on a limited dataset of smart mobile phones, with hand-picked brands and models (e.g., Apple iPhone 3GS, iPhone 4, iPhone 4S, iPhone 6, Samsung GT-i8268 Galaxy, Samsung S4 Active, Samsung Galaxy S6 and S6 Edge, Samsung S8 Plus, Pixel 2 XL, Xiomi Note, HTC One, Huawei Mate 8, 9, 10, and P8 Lite). As the training dataset consists of images for a limited variety of brands and models, the method cannot be used in an actual recycling plant where a phone disassembly line is in action. Moreover, all of the images found in the dataset were of smart mobile phones, leaving out a big chunk of the regular mobile phones accumulated in the recycling plants, waiting to be recycled. Therefore, the suggested method does not completely solve the smart phone disassembly problem, let alone solve the mobile phone disassembly problem. There is also the fact that granularity is an issue when it comes to objects such as mobile phones. Indeed, the reported average precision in this work for the small objects (AP_S) is only 35.3, making it an unusable product at the end of the day.

Detection vs. Classification

There are plenty of tasks involved in an automated disassembly scheme that impose *Image Classification*, which helps the scheme to classify what is contained in an image, whereas *Object Detection* specifies the location of multiple objects in an image. Location information here, however, should not be confused with what segmentation yields, which is a pixel-wise mask of each object in the image.

In most of the EOL devices, components are tightly attached to each other via entities such as screws and bolts. For any intelligent scheme to conduct disassembly of an EOL device, location and type information of such entities have to be found out. Consider a computer hard drive (3.5") such as the one illustrated in Fig. 2.6, where four screws are seen to be holding the lid. Weather a human operator or an intelligent visual

Figure 2.6: A 3.5" computer hard drive with four Phillips-1 screws on its lid.

scheme, one of the primary tasks here is to detect the screws (finding their locations) and to classify them based on their head types. In context of a visual scheme, this vital piece of information could then be used in various further blocks such as manipulation, where the location and type information is necessary to conduct visual servoing over the detected screw's location, as well as to pick up the suitable tool for unscrewing action. Accurate detection and classification methods are pivotal to an intelligent disassembly scheme, as trying to unscrew on a wrong location (misdetection) or picking up a wrong tool (misclassification) could lead to damaging the robotic tool tip, as well as the screws on the EOL device, hindering the entire disassembly effort.

Under the light of aforementioned facts, the literature on detection and classification tasks in the context of disassembly has been reviewed. As automated disassembly has been investigated for a while now [Drö+14; Weg+15; Wei+06], there are some schemes [BRP16; Bük+01; Els+12;

Pom+04a; Xie+08] for automating certain processes. While reviewing the literature on detection and classification methods, it has been noticed that none of the proposed schemes (up to 2019) have offered a generalizable, extendable, and universal solution of the detection problems of vital entities (e.g., screws, bolts). Numerous algorithms have been suggested for screw detection as a part of automated disassembly procedures, which are extensively discussed in Chapter 5.

As for the classification task, the research is quite narrow. Similar to the detection, there is no universal, generalized method that actually handles multiple types of an entity. This is due to the fact that many of the industrial plants are using fixed-templates and/or various hard-coded information in order to carry out the classification tasks [DHP04; NF15]. Since these plants mostly have an assembly line robot that is specifically tailored to assemble a certain part on another, it is not necessary for the scheme to be universal and generalising. Therefore, it is safe to state that one of the goals of this thesis is to bring a universal, template-independent and generalized solution to detection and classification problems of screws.

2.3.3 Model Representation

A model is used to describe the information for an object prior to performing in the disassembly process (e.g., prior knowledge in a database). Torres et al. [Tor+04] explains two approaches, namely, relational model and geometric model. First, a relational model represents the relationship among components via their connections. They are represented by hierarchical graphs which remain simple, despite the increased number of components. Second, a geometric model represents the product in multi-dimension. It presents the physical information for each component (shape and size) and relation between each component (location and contact surface). Tonko and Nagel [TN00] found that most geometrical models of rigid and valuable components can be depicted by polyhedra, non-polyhedra, quasi-polyhedra,and other primitive geometries.

Hohm et al. [HHT00] suggest modelling of the environment by grouping objects into two types. First type is what they call active objects (the main part of the product), whereas the second type is called passive objects (the connective components, e.g. screws, cables, and snaps). Moreover, Jorgensen et al. [JAC96] suggest organising a number of product models as

a hierarchical tree to facilitate the object recognition process. Each node represents attributes of each sub-assembly or component. Furthermore, Buker et al. [Bük+01] present the approach to reconstruct the 3D model by active-stereo vision. This method benefits the operation in terms of representing a clear scene of position and orientation of the object, especially in visual-servoing (e.g., to prevent collision between the product and robot arm).

There are a couple of problems using model representations in intelligent disassembly schemes, however. First, an intelligent disassembly scheme would have to account for differences in EOL devices (i.e. brands, models, sizes). While this could be overcome by introducing models of each model, brand and size, it either forces the scheme to retrieve these models from their official sources or it forces an active-stereo vision pipeline to create 3D models in prior. If the entire EOL device category is considered, neither of these approaches are feasible, as there are countless devices and models in the picture. On top of that, electronic manufacturers are usually not willing to share the schematics of devices they manufacture. Thus, the idea of using model representations in an intelligent scheme is only realistic within an extremely narrow set of devices and models.

2.3.4 Computer Vision Libraries/Frameworks

Computer vision libraries are available in various language platforms (e.g., C/C++, Python, Java, Matlab, etc.). The libraries commonly used in research and development are Open Source Computer Vision (OpenCV) [BK00], in C/C++ and Python, ImageJ for Java [Ras+97; Ras12], and Image Processing Toolbox for Matlab [MAT10]. OpenCV is selected to be used in this research since it provides most of the algorithms used in the aforementioned literature, e.g. template matching, camera calibration, etc. Moreover, it is compatible with C/C++ which is typically used in robotic research due to the fast processing and accessibility at the machine level, and with Python which is typically used in deep learning research due to its ease of use and compatibility with ROS (Robot Operating System) [Qui+09].

2.4 Product Case-Study: LCD Screens

This section gives information of the case-study device, Liquid Crystal Display (LCD), in the perspectives such as overview of the impact, EOL treatment and disassembly process. The effort can be called a spin-off project of the IMAGINE effort, which is the main motivation of this thesis. Together with the partners from the Electrocycling GmbH [28] in Goslar, Germany, an LCD frame separator has been developed. The subsections following will state the problem and explain the implemented method.

2.4.1 End-of-Life Treatment of LCD Screen Monitors

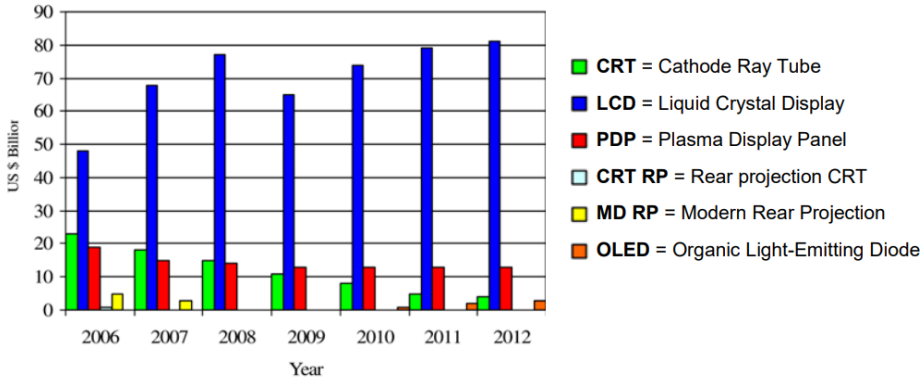

Figure 2.7: Predicted sales of types of monitors[ROL11].

Cathode Ray Tube (CRT) monitors have been replaced with LCD screens over the past 10 years. Kernbaum et al. [KFS09] state that more than 120 million units of LCD screens were sold worldwide in 2008 and they predicted LCD screens to be used in approximately 90% of desktop computers in 2010. As of 2020, there are no computers manufactured or sold with CRT. With the cutting edge technologies such as LED (Light Emitting Diode), OLED (Organic Light-Emitting Diode) and QLED (Quantum Dot Light Emitting Diode), the age of CRTs is basically over. In addition, Ryan et al. [ROL11] predicted the sales of LCD screens to be US$80 Billion which was

28 http://www.electrocycling.de/

approximately four times higher than the sales of other types of monitor in 2012, as illustrated in Figure 2.7. Therefore, the number of disposals is continuously increasing. For instance, in Germany alone, Kernbaum et al. [KFS09] predicted that more than 4,000 tons of LCD screen monitor would be disposed of by 2012. The report [Buc+12] in 2012 by the Öko Institut [29] states the sales numbers in quantities, by which, we could approximate the disposed LCDs by 2020. According to the report, in 2010, 2,576,000 of LCD computer monitors and 8,258,000 LCD televisions were sold in Germany alone. Although these are not disposed LCD monitors, rather the number sold, these televisions inevitably become EOL products, as every electronic product has a limited life cycle. If we take the average weight of an LCD monitor to be 5 kilograms, we get approximately 12,880 tonnes of only computer monitors, which is way higher than the predicted numbers by Kernbaum et al. It should also be underlined that the calculation considered only the computer monitors, not the televisions, as it is not simple to calculate how much would 8,258,000 televisions weigh, due to various screen sizes of televisions. However, it is critical enough to see that the predictions of the research community in 2009 does not even come close to the realities of today. Therefore, it must be acknowledged that the impact from these products has increased significantly, and EOL treatment needs to be considered.

2.4.2 Disassembly of LCD Screens

There have been two works [KKS09; ROL11] published on LCD disassembly previously. Each of these works examines the LCD device in terms of layers/components, as well as the disassembly sequence to be considered. Figures 7.3 and 7.4 in Appendix 7.1 illustrate modules and components of LCDs. This case study, however, will only consider the LCD TV frame removal process of the disassembly sequence, as the goal of the spin-off project is only to automate this particular stage where the human labor could be replaced with high level automation.

At the designated recycling plant, the frame removal is done by a milling machine operated by a human. The TV is laid over and locked tight by the two side-holders of the milling machine. Afterward, the human

29 https://www.oeko.de/

operator conducts his manual measurements using a ruler, and finds out where the frame starts, where it ends. He also uses a sheet of paper where aspect ratios/dimensions for the LCD TVs are given. According to the measurements conducted at the recycling plant, a human operator spends approximately 1.5 minutes while removing the frame. Time was measured from the moment when the human operator lays the LCD TV over the milling machine and locks the holders from both sides. Since this very initial step has to be repeated regardless of who or what handles the frame removal, it is excluded from the time measured. It must be also said that the milling machine is controlled by a computer with its specific software installed on it, as shown in Figure 2.8. Fortunately, there is a C++ API (Application Programming Interface) released that is utilized to automate certain tasks to a degree.

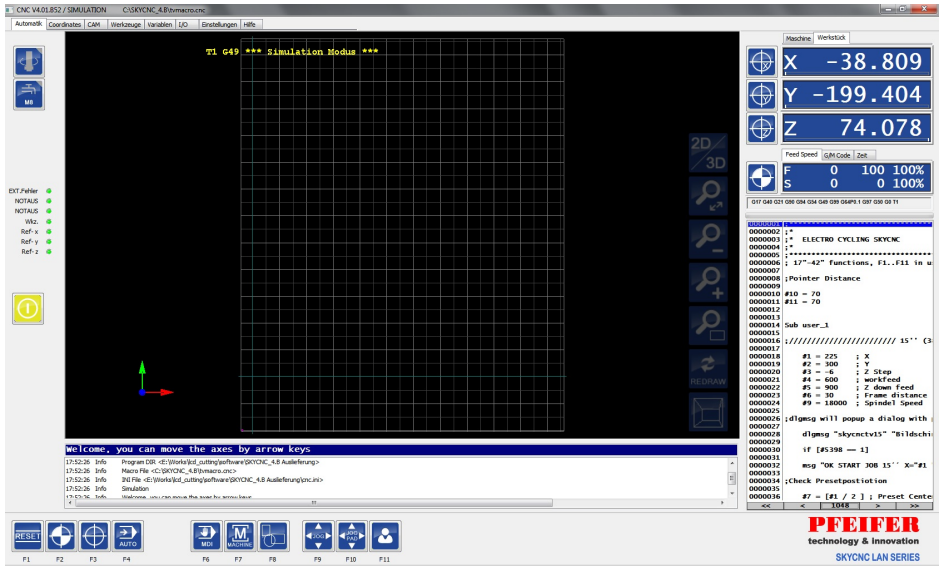

Figure 2.8: A GUI snapshot from the SKYCNC software from Pfeifer Technologies [30]

Given the above scenario, one could conclude that a high level automation

30 SkyCNC WM Serie Fräsmaschinen. (2021, January 12). Retrieved February 23, 2021, from https://www.pfeifer-technology.de/skycnc-wm-serie-pfiffige-grossformat-bearbeitung/.

scheme has to conduct the same operations either in less or equal time, as otherwise it may not be preferable by the engineers at the plant. It should also be added that the proposed pipeline should not involve cutting-edge/military-grade sensors that could exceed the recycling plant's budget spared for the task. All in all, the goal is to come up with an affordable pipeline that uses the same computer running the milling machine (i.e. no extreme computational load) and the regular sensor(s) available in the market. The motivation behind mentioning this spin-off project in this thesis is to provide an actual automatization of a human-operated disassembly line. Since this spin-off project is not directly linked to the proposed scheme in this thesis, the following subsections only cover it on the surface level.

2.4.3 Semi-Automated Disassembly System for LCD TVs

The aforementioned criteria for the project reduce the options for sensors to a few, including camera, ultrasonic sensor and a line-laser. It also eliminates the possibility of using computationally demanding algorithms or frameworks, as the computer running the milling machine does not have an external GPU or a high performance CPU, or RAM. All these conditions further restrict the choices of sensors and algorithms to be considered. It is then proposed to use the current setup with some minor modifications. The existing setup consists of a milling track (see Figure 7.7 in Appendix 7.1) on which a TV is placed and fixed by a movable bar, and a milling machine with an ultrasonic sensor to detect/set height.

By measuring the frame height (from the TV surface) as well as the TV height, the manual operation conducted by the human operator could be automated. Two sensors (see Figures 7.5 and 7.6 in Appendix 7.1) are decided to be employed for the designed algorithm.

The proposed cutting procedure starts with placing a TV manually on the milling track at the fixation edge, as shown in Figure 2.9. There is a warning shown on the control panel of the CNC milling machine (or by ways of LED color) in case it was not placed correctly, and, the TV is then re-placed. A default initial position of the milling machine is as shown in Figure 2.10A, and the machine inspects the top-right corner of the TV in order to generate a milling path, thus, one has to make sure that this corner is NOT damaged. If both the upper-right and the bottom-left

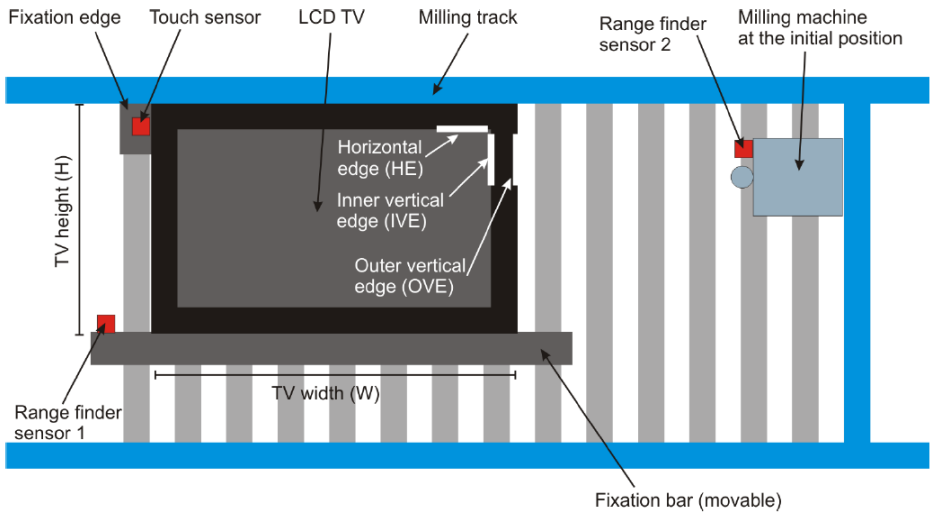

Figure 2.9: A schematic diagram of the LCD cutting setup.

corners of the TV are damaged, then the human operator can change initial position of the milling machine (as shown in Figure 2.10B) by choosing an alternative initial position on the control panel of the CNC machine.

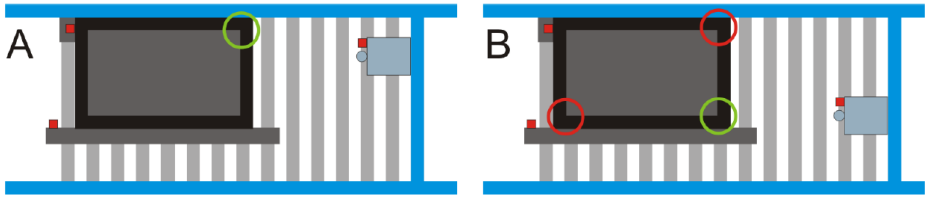

Figure 2.10: Initial positions of the milling machine: A) a default initial position; B) an alternative initial position in case both the upper-right and the bottom-left corners of the TV are damaged and cannot be detected. Green and red circles denote non-damaged and damaged corners, respectively.

After making sure the promising corner is selected, the fixation bar is moved towards the TV to fixate it. The human operator initiates the process, where corner detection runs. The height of the TV (H, see Figure 2.9) is estimated by using data of the range finder sensor 1. Following, the

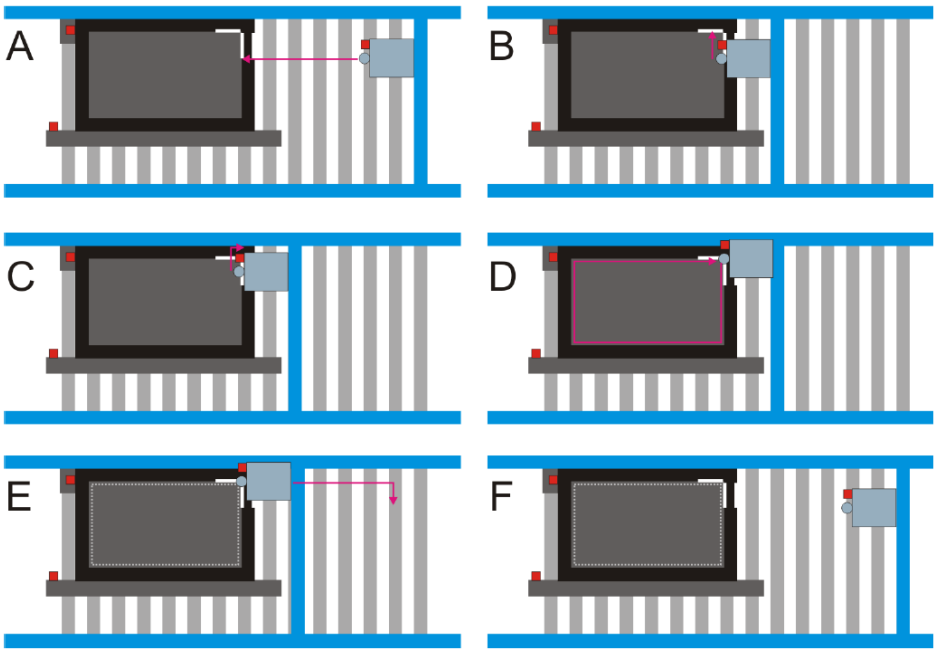

Figure 2.11: Steps of the cutting procedure: A) drive until inner vertical edge (IVE) of the TV is detected; B) drive until inner horizontal edge (HE) of the TV is detected; C) go to the milling start-point; D) cut the screen; E) return to the initial position (F).

milling machine drives towards the TV (see Figure 2.11A) until the inner vertical edge of the TV (IVE, see Figure 2.9) is detected. Note that on the way the outer vertical edge (OVE) is detected, which is used to estimate the width of the TV (W, see Figure 2.9). Here, data of the range finder sensor 2 is used to detect edges of the TV. Afterward, the milling machine drives up (see Figure 2.11B) until the inner horizontal edge (HE, see Figure 2.9) is detected. The milling start-position is computed and the milling path is generated based on the previously estimated measures: TV height (H), TV width (W), the width of the horizontal edge and the width of the vertical edge. Note that the height of the milling path is also computed based on data of the range finding sensor 2. Following this, the milling machine drives to the milling start-point (see Figure 2.11C) and waits for the initialization of the cutting process. At this point, it is crucial that

the human operator inspects the milling start-position. This is done to check the start-position and to make sure that the corners were detected correctly. In case of a significant error, the human operator is to adjust the starting-position manually. If there are no errors, then the cutting procedure starts with the human operator's approval. The milling machine then cuts the screen in anticlockwise direction along the precomputed path in an open-loop mode (see Figure 2.11D). After the path is driven completely, the milling machine drives towards its initial position (see Figure 2.11E, and Figure 2.11F), where the process ends.

Figure 2.12 depicts the scanning results from a sample LCD TV, where the spikes correspond to height changes along the scanning path. One could make a few observations here. First, the readings show a significant drop after the point 1, where the frame starts. The reason for the drop is due to the fact that the distance between the range sensor and the frame reads less in comparison to the previous steps in the path. Point 2 in the plot corresponds to the frame's end, where the screen surface begins. As a result, a small increase in readings is noticeable. As the path is taken, point 3 and 4 are reached, where screen height is found. There is also the fact that not all LCD TVs have rectangular shapes, and thus, some of the frames have circular plastic addition as observed in Figure 2.12. Such features cause additional spikes in readings, however, as the frame boundaries are determined by taking only the first 4 spikes, the rest is ignored, and thus, they do not cause a problem. There are more scanning results (see Figure 7.8 that can be found in Appendix 7.1.

Throughout the investigations, it was noticed that some TVs that are highly damaged on the corners are causing problems. Naturally so, since our approach imposes a scanning procedure on a path including the sides of the TV. If both sides of a TV are damaged, then the human operator switches to the manual mode, and continues to process the TV. It must be also noted that the human operator has to be present all the time where the automation takes place. There is always confirmation before the scanning and cutting procedures start, as well as an emergency stop, which is essential in any milling machine. It should also be underlined that the ultrasonic sensor RPS-412A has a golden plate surface (see Figure 7.5 that is not supposed to make physical contact with any object or particle.

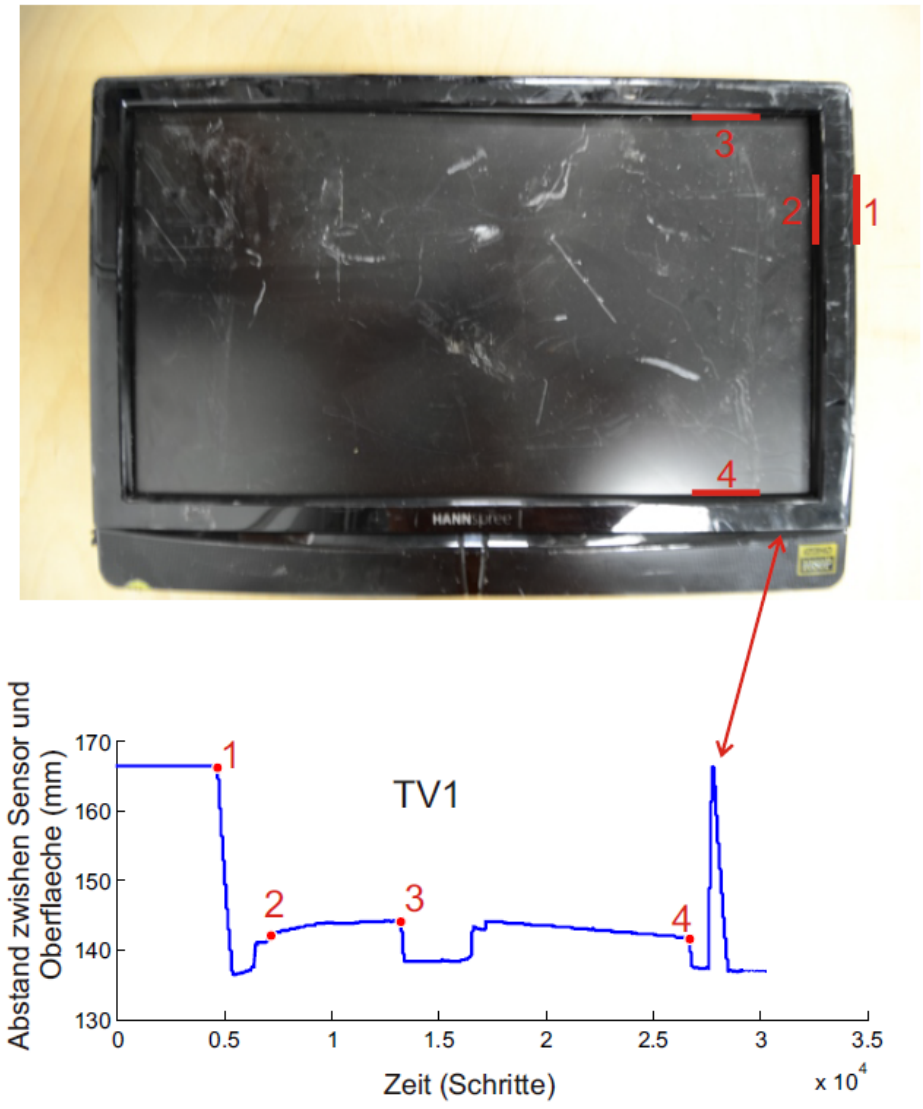

Figure 2.12: Scanning results obtained from the range sensor 2, including a bump in the frame which makes the pointed spike in readings. Y-axis in the plot is the distance in millimeters, whereas X-axis is time (steps).

Understandably so, since contact with the golden plate surface may cause misreadings, leading to path errors.

After the system is set up, the time elapsed for the automated scanning and cutting procedures are calculated. It appears that the designed automation system takes 1 minutes 30 seconds on average, which is less than the human operator.

2.5 Conclusion & Problem Statement

After reviewing the literature, it is acknowledged that automation in disassembly routines is inevitable, as many disassembly routines consist of repetitive actions that can be parameterized. These parameters are approximated by using various sensors such as cameras, ultrasonic sensors, etc., that are, along with the necessary vision and/or data processing algorithms, forming the perception block of an automation scheme.

However, there is still a great lack of intelligence in automation of disassembly routines, preventing them from processing various devices, or different models of the same devices. This is due to the fact that most of the employed methods are either relying on predefined model representations (e.g., 3D models) or classical computer vision methods where the generalization aspect largely ignored. Model representations are difficult to obtain due to company polices of manufacturing plants, and creating 3D models of each and every EOL device/model in the recycling world seems unlikely. On the other hand, classical computer vision methods are usually hindered by varying light conditions and they are not generalising enough to handle different situations in the disassembly scenarios.

It could be then stated that the current literature lacks a generalising, template-independent, universal visual intelligence scheme that could be extended to various EOL devices with little or no effort.

3 System Overview

This chapter presents an overview of the planned automated disassembly system which includes the perception block that is the subject of the thesis. Therefore, it is essential to mention the underlying pipeline that enables the entire system to function. The content is divided into three sections describing the blocks -and their interactions- in the pipeline, perception block in slightly more detail, finally the planned tasks and responsibilities of the perception block. Most of the information found in this chapter is taken from the project documentation that was written before the project started. Therefore, the found information aims to inform the reader to give an understanding on the abstract level. Technical details are naturally omitted as they have been subjected to change throughout the project.

3.1 IMAGINE as an Automated Disassembly System

As mentioned in the first chapter, IMAGINE [31] is a project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761. Its consortium consists of six universities from various countries and a private recycling company in Germany. Each partner in the consortium has been given its responsibilities and tasks they are supposed to address during the project lifetime.

It is therefore required to describe the planned high-level architecture of the automated disassembly system and to point out the specifics of the perception block that is going to be described in this thesis.

3.1.1 Objectives

It is necessary to mention IMAGINE's objectives in this thesis, since the proposed scheme was developed to address these objectives. The scientific objective of IMAGINE is specified as enabling robots to understand how

31 https://www.imagine-h2020.eu/

their environment behaves and how to interact with it, resulting in substantial advances in autonomy, flexibility and robustness of interaction in the presence of unforeseen changes in the environment. Specifically, in the context of IMAGINE understanding means the ability of the robot to:

- infer which actions possibly apply in a given state of the environment, and how to parameterize the actions to achieve a desired effect;
- discern to what extent its actions succeed, and if the action effect differs from the expected outcome, to infer possible reasons and recover from failures.

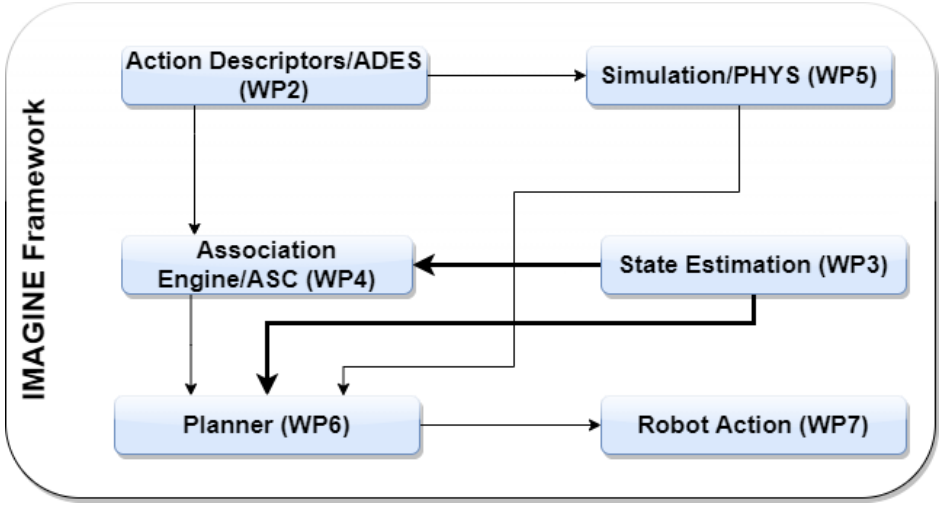

Figure 3.1: Principal relations between work packages.

On the other hand, the industrial objective of IMAGINE is to drastically augment the level of structural understanding of devices and functional understanding of actions available to robots, enabling them to infer how to disassemble entire categories of electromechanical devices and appliances, to monitor the success of their own disassembly actions, and to synthesize effective recovery strategies for problems they encounter during disassembly. (IMAGINE Consortium, 2016, p.3) To address this vision, IMAGINE focuses on the important industrial and environmental use case of large-scale recycling of electromechanical devices and appliances [Con16].

3.2 Core Components of IMAGINE

After stating the industrial and scientific objects of IMAGINE, activities of it must be mentioned as well. The activities of the IMAGINE project are organized into 7 technical work packages (WP). Each of these work packages contains work that requires rich collaboration internally, while work packages interface with each other in well-defined ways. A graphical overview of the work packages and the principal flow of information between them is shown in Figure 3.1, where important connections involving the focused package in this thesis are shown with bold arrows. The WP titles do not require further explanation at this point, and match the structure of the IMAGINE system as outlined in further subsections in this chapter. Each WP is explained in sufficient detail later, with the main focus being on the WP called **State Estimation**, which is the perception bloc mentioned previously. The core functional element is a generative model based on an association engine (ASC) and a physics simulator (Simulation). *Understanding* is given by the robot's ability to predict and simulate the effects of its actions, before and during their execution. This allows the robot to choose actions and their parameters based on their simulated performance, and to monitor their progress during execution by matching observed to simulated behavior. (IMAGINE Consortium, 2016, p.2)

Moreover, a short summary of core components (each implemented in its respective work package) should be described. Due to their administrative and exploitative roles, the work packages 1, 8 and 9 are skipped. Rest of the work packages and their descriptions are as follows.

- **ADES:** Action descriptors incorporate human expert knowledge and are validated and refined in simulation and real execution (work package 2).
- **State Estimation:** This component analyzes the scene using vision and depth sensors (work package 3), and is the subject of this thesis.
- **ASC:** The association engine (work package 4) learns mappings from scene descriptions to actions, from scene descriptions to action parameters, and from scene descriptions, actions and their parameters to expected action outcomes.

- **PHYS:** The physics simulator (work package 5) plays a core role in the robot's understanding of the scene context and its own actions, by simulating actions to reproduce or predict their actual effects under various parameter settings.

- **Planner:** This component (work package 6) plans the next action to be performed, given a scene description and a set of possible actions. Taking into account output variances and success probabilities, a short-horizon planning process determines the most favorable safe action by anticipating possible failures and dead-ends, e.g., those caused by irreversible/damaging actions.

- **Execution Engine:** This component is responsible for executing the actions chosen and parameterized by the interplay of the above components (work package 7).

- **Multi-Functional Gripper:** Robot execution relies on this innovative end effector with advanced control strategies (work package 7) for flexible, rapid, robust, and cost-effective manipulation using affordable, off-the-shelf robot arms.

Since this thesis focuses on a specific work package (WP3), the following sections and subsections are explaining the expected tasks from this package, and its possible interactions with other packages.

3.3 State Estimation Package of IMAGINE

The objective of this package could be summarized as estimation of the state of the scene containing the object being disassembled, its relevant parts, and tools using computer vision methods. The following subsection specifies the foreseen tasks that are likely to be solved. It should be kept in mind that the planned methods in the State Estimation Package of IMAGINE do not necessarily have to one-to-one match with the methods employed in the end-product.

3.3.1 Description of Work

- **Task 1: Objects and Poses** entails performing visual object recognition and pose estimation of parts and components. This task inevitably imposes the usage of different 2D or 3D (Kinect or stereo) vision setups to record the scene.

- **Task 2: Relations** is interested in estimation of geometry and relations, including physical contact. This task is strongly linked to the pose estimation aspect of the previous task. Recognizing decisive structures such as gaps (for grasping or levering), grasp-affording locations, hammer-hit-affording locations and free-direction-for-removal of a part (this is strongly related to hit-affordance) are core functionalities in this frame. Following the previous task, one can model parts and their poses with some degree of accuracy. As a consequence, gaps could be found where there are no parts.

- **Task 3: Visualization/User Interface** is planned to provide a UI (user interface) to visualize the results from the previous two tasks (e.g parts, all relations and affordances). This is rather a technical task. It is needed in the development process for result verification, and for the training procedures of the DCNNs (Deep Convolutional Neural Networks). The UI would also allow the developers (and even the end user, if needed) to label device parts.

3.3.2 Perception of the Scene

State Estimation is inevitably set to use certain sensors to perceive the scene. As stated in the task descriptions before, the information to be extracted requires perception of 2D and 3D structures alike, such as parts and gaps, respectively. In order to satisfy this requirement, the IMAGINE project is planned to employ certain sensors. While the exact roles of these sensors are specified in Chapter 5, it is helpful to mention their types and the information they acquire to have an accurate overview of the State Estimation component. In Figure 3.2, a simplified graphical illustration of scene perception is depicted.

Any visual perception pipeline deploys relevant sensors to achieve its tasks. The State Estimation component is set to use various computer vision

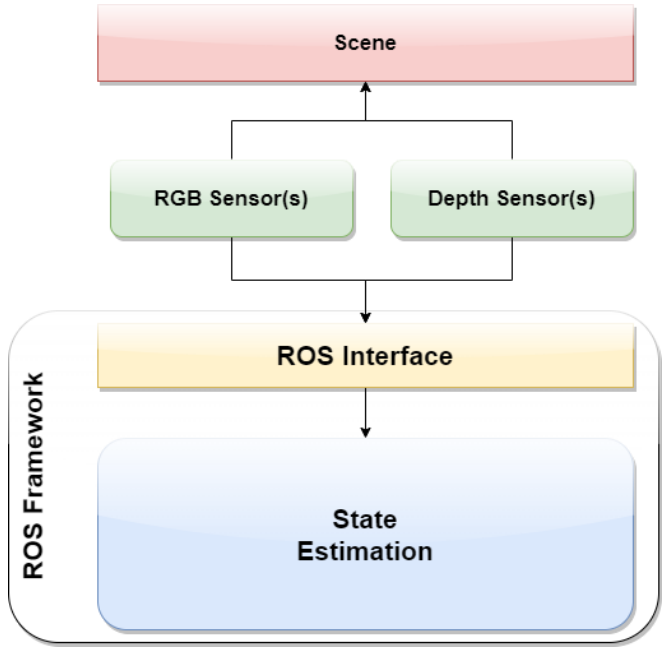

Figure 3.2: Graphical representation of scene perception involving State Estimation (WP3) and possible sensor types.

algorithms involving RGB and RGB-D processing. Thus, it is essential that the chosen sensors providing the necessary information are operating within the desired ROS Framework, and, are in interaction with the State Estimation component through a ROS interface in place. It is sufficient to underline this interaction in this chapter. Further technical details on sensors and visual information acquisition are provided in Chapter 5.

3.3.3 Inter-Component Relations

The State Estimation package inevitably interacts with other core components of the IMAGINE project. This subsection aims to shed light on these interactions and to inform the reader on the workflow, again, on an abstract level. Before mentioning these interactions, however, it must be stated that the IMAGINE project uses Robot Operating System (ROS) [Qui+09] as its underlying framework. ROS is an open-source framework

for the development of robot applications. It offers a suite of tools to build and distribute software, client libraries to support C++, Python and Lisp access, and a library of software packages that can be imported into projects. One of the main design decisions of ROS is modularity. ROS offers a flexible way to arrange different software solutions into modules that can be arranged and switched out to make the robot system flexible. A ROS application is represented as a computational graph with single processes called nodes that are connected by topics. A node can get input information from several other nodes that are connected to the node by a topic. This means that any IMAGINE component sends and receives information encapsulated by the available and suitable means of ROS. The choice of underlying framework is not a core requirement, rather merely a decision made considering the convenience.

The ROS core also has several tools that allow visualization and recording of the data flowing through the computational graph. To visualize the outputs of the vision algortihms, the RVIZ tool[32] is used, a three-dimensional visualization tool that can display the processed point clouds in every stage. Another important ROS module is dynamic reconfigure[33], which allows to reconfigure parameters of individual nodes from a GUI(Graphical User Interface) at any time without recompiling the module. This is not only helpful for debugging reasons but in combination with RVIZ helps to visualize the impact of changes to the different algorithm parameters.

State Estimation is planned to be in interaction with two other components, namely the Association Engine (WP4) and the Planner (WP6), as illustrated in Figure 3.3. Each interaction entails passing of certain information essential to the workflow. In case of the interaction with the Association Engine, this information is the part information, whereas in other case it is the predicates. The term *part information* here refers to information relevant to all parts that are detected and/or recognized in the scene, whereas the term *predicates* refers to the physical relations between these parts (e.g., `next_to()`, `contains()`, etc.). At this point, it is important to give a couple of definitions to empower the presented workflow.

As stated previously, the main objective of State Estimation is to estimate

32 http://wiki.ros.org/rviz
33 http://wiki.ros.org/dynamic_reconfigure

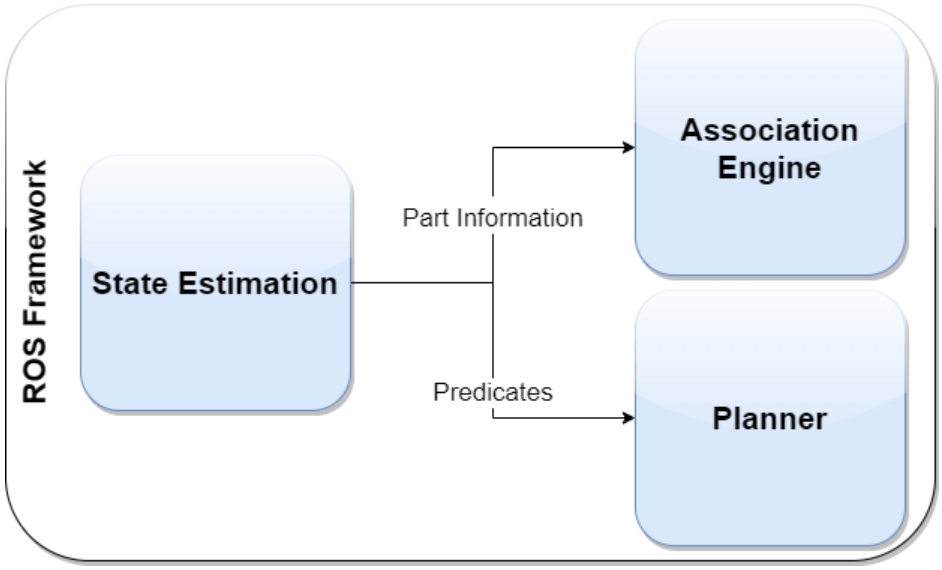

Figure 3.3: Graphical representation of inter-component interactions involving State Estimation (WP3).

the state of the scene containing the object being disassembled, and, its relevant parts. This estimation is expected to result with the acquired scene information, most of which is part information (e.g., boundaries, centers, etc.). Association Engine retrieves the part information to further evaluate the affordances, whereas Planner processes the predicates to come up with a set of instructions to disassemble the device. Further specifications and technical details of these interactions are explained in Chapter 5, where the developed pipeline is described in depth. For this chapter, it is found sufficient to specify the inter-component relations on high-level.

3.3.4 Summary

In conclusion, one could summarize the overview of the State Estimation component that is planned to perceive the scene using appropriate sensors operating under ROS framework. It is planned to address three main tasks as mentioned before, each satisfying a requirement in the project. While doing so, it is in interaction with two other components of IMAGINE,

passing relevant information regarding the state of disassembly. Figure 3.4 depicts a graphical illustration of this summary.

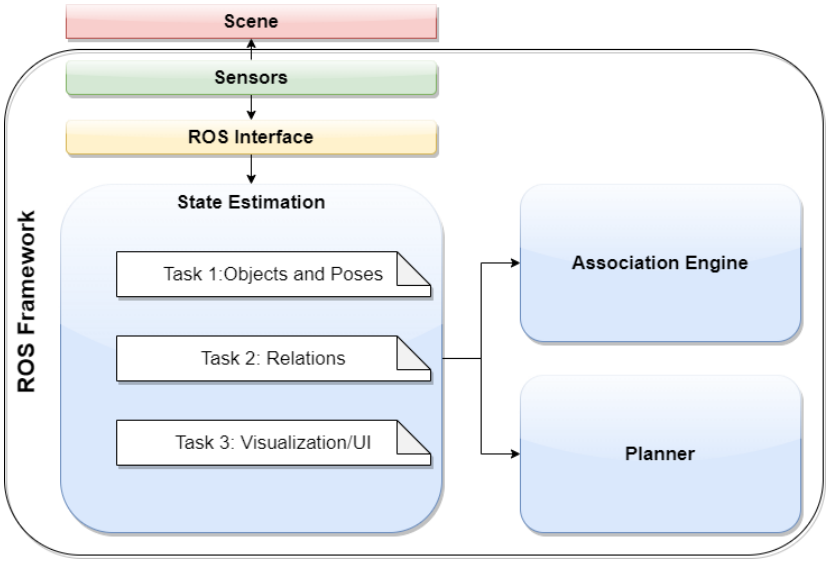

Figure 3.4: Graphical representation of State Estimation package, its tasks and interactions with other components in IMAGINE.

4 Background

This chapter aims to provide the fundamentals that are required to understand the approach presented in the thesis. Since there are various subjects to take into account such as image acquisition, stereo geometry, deep learning, point cloud clustering, the chapter is organized in sections and subsections addressing each. The chapter starts with the basics of computer vision such as image acquisition, and goes on with stereo imagery, and depth acquisition. It then explains how depth acquisition could be utilized to form very strong data structures called point clouds, and elaborates the required operations in point cloud domain. Finally, it introduces the basics of machine learning, and goes further into deep learning, which is one of the most essential concepts playing a major role in this thesis[34].

4.1 Image Acquisition

It is essential to provide the definitions of several terms that are pivotal to the processes further explained in this work. The most basic structure, an *image*, could be denoted as two-dimensional functions of the form fx, y [MP13]. The value of f at spatial coordinates x, y is a scalar quantity whose physical meaning is determined by the source of the image, and whose values are proportional to energy radiated by a physical source (e.g., electromagnetic waves). As a consequence, fx, y must be non-negative and finite; that is, function fx, y is characterized by two components: (1) the amount of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene. (Mohapatra, 2013, p.1047-1054.) Most of the images this work studies generated by the combination of an light source and the reflection or absorption of energy from that source.

34 The chapter contains materials that are supplied by the Bachelor's thesis projects [Bri20; Tro20] of the undergraduate students in the neuroscience group of University of Göttingen. Their contributions are acknowledged.

There are numerous ways to acquire images, but the objective in all remains the same: to generate digital images from sensed data. The output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. To create a digital image, one needs to convert the continuous sensed data into a digital format. This requires two processes: *sampling* and *quantization*. An image may be continuous with respect to the x and y coordinates, and also in amplitude. To digitize it, one has to sample the function in both coordinates and also in amplitude. (Gonzalez, 2007, p.52) Indeed, it is stated as "Digitizing the coordinate values is called sampling, whereas digitizing the amplitude values is called quantization" (Shandilya, 2019, p. 66).

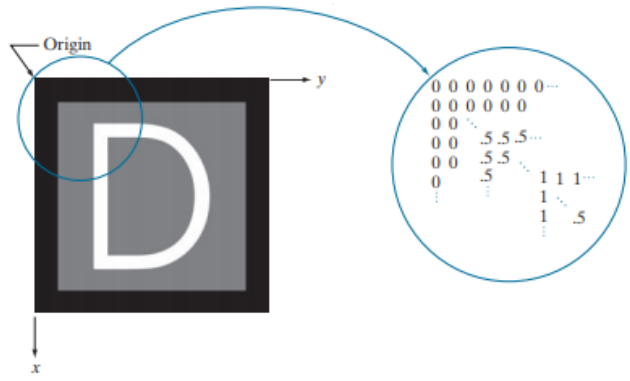

Figure 4.1: Image displayed as a visual intensity array, along with its 2D numerical array representation. The numbers 0, .5, and 1 represent black, gray, and white, respectively. [GW02]

Let fs, t represent a continuous image function of two continuous variables, s and f. One first converts this function into a digital image by sampling and quantization [GWE04], as explained above. Suppose that the continuous image is sampled into a digital image, fx, y, containing M rows and N columns, where x, y are discrete coordinates. For notational clarity and convenience, the computer vision community uses integer values for these discrete coordinates: $x = 0, 1, 2, ..., M - 1$ and $y = 0, 1, 2, ..., N - 1$.

Thus, for instance, the value of the digital image at the origin is $f0, 0$, and its value at the next coordinates along the first row is $f0, 1$. As Figure 4.1 shows, one defines the origin of an image at the top left corner. This is a convention based on the fact that many image displays (e.g., TV monitors) sweep an image starting at the top left and moving to the right, one row at a time. More important is the fact that the first element of a matrix is by convention at the top left of the array. (Shandilya, 2019, p. 66) Choosing the origin of fx, y at that point makes sense mathematically because digital images in reality are matrices. In fact, in many occasions, the community uses x and y interchangeably in equations with the rows (r) and columns (c) of a matrix. Each co-ordinate position is called as *pixel*. Pixel is the smallest unit of the image it is also called as picture element. Therefore, digital images are composed of pixels, each pixel represents the color (gray level for black and white images) at a single point in the image. A digital image is a rectangular array of pixels also called as *Bitmap*. From the point of view of photography the digital images are of two types [Cof05; Sac96]: Black and White images, and color images.

Black and white images are made of different shades of gray. These different shades lies between 0 to 255, where 0 refers to black, 255 refers to white and intermediate values refer to different shades of black and white. Grayscale refers to the range of neutral tonal values (shades) from black to white. Color images, on the other hand, are made up of colored pixels, and color can capture a much broader range of values than grayscale. The spectrum – the band of colors produced when sunlight passes through a prism – includes billions of colors, of which the human eye can perceive seven to ten million [MKS17]. The electronic capture and display of color is complicated. RGB (Red, Green, and Blue) is the most commonly adopted color system. [Pet05] A colored image is usually called *RBG Image*.

After defining what a digital image is, its types and representations, one could continue with how an image is acquired. This requires understanding the image acquisition process. "The general aim of image acquisition is to transform an optical image (real world data) into an array of numerical data which could be later manipulated on a computer, before any video or image processing can commence an image must be captured by camera and converted into a manageable entity" (Sigmura, 2015, p. 24). The image acquisition process consists of two steps. First, the optical system focuses

the energy that is reflected from the object of interest [Sug+15]. Second, a sensor that measures the amount of energy reflected.

Image acquisition is achieved by various suitable cameras, different cameras for different applications, depending on the objective. For instance, if an X-Ray image is needed, a camera (film) that is sensitive to X-Ray is used, a camera that is sensitive to infrared radiation is to be used if infrared images are desired. For the regular images one mostly encounters in daily life (pictures) cameras that are sensitive to visual spectrum are used. Note that the image acquisition is the first step in any image processing system.

4.2 Monocular and Stereo Imagery

There are two major visual perception techniques considered in the computer vision domain: monocular vision and stereo vision. Depending on the application and resources, developers are inclined to use one of them, or even both.

Monocular vision refers to type of vision in which either a camera is used alone, or multiple cameras used separately (i.e. with no correlation with each other). By using the cameras in this way, the field of view is increased, while depth perception remains limited. A monocular camera is a very common type of vision sensor used in computer vision applications. At this point, however, before one goes deeper into further details, the camera model has to be defined. We use the description of the *pinhole camera* as the model for most of our imaging devices today. The pinhole camera models the standard photo camera, video camera, which also model the eye. The graphical illustration of the model can be seen in Figure 4.2. "Light from a point travels along a single straight path through a pinhole onto the view plane. The object is imaged upside-down on the image plane" (Fleet, 2006, p.33) [35].

Figure 4.2 presents a camera with center of projection O and the principal axis parallel to Z axis. Image plane is at focus and, thus, focal length f away from O. A 3D point $P = X, Y, Z$ is imaged on the camera's image plane at coordinate $P_c = u, v$.

When we look around as humans, the 3D world is optically projected onto the retina (photo sensitive layer) in our eye. The luminance (visual

35 https://www.dgp.toronto.edu/ hertzman/418notes.pdf

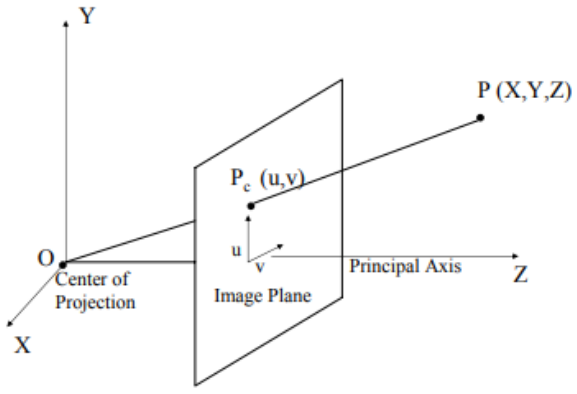

Figure 4.2: Graphical representation of pinhole camera model.

energy) is measured in millions of photo sensitive cells (the rods and cones) in the retina. The collection of all these measurements makes up an image of what we see, in much the same way as a collection of pixels on the screen forms an image. Similarly, the pinhole camera projects a 3D scene onto a 2D retina. In such a projection, the 3D information is lost. All points on a straight line from the optical center of the camera are projected on the same point on the retina. We cannot recover depth from one image (at least not from one point (pixel) in the image). The human visual brain also uses our knowledge of the 3D world around us to infer depth. Two eyes/cameras (at minimum) are required to measure depth. It is necessary to take a look at the setup with two cameras (stereo vision), and model the way the two images are related to recover depth.

Stereo vision is the second paradigm in computer vision that can recover the aforementioned lost third dimension by taking at least two images of the scene from different vantage points and using the correspondences between the images to calculate a depth estimation. This approach is closely related to the way our eyes allow us to perceive depth. Human eyes are placed in lateral different positions to perceive the world from two slightly different views. The difference between these two views is called binocular disparity and is a key factor for depth perception in the visual cortex. Computer stereo vision uses two or more cameras as sensors to create images of the scene. Similar to the eyes, the cameras are also placed

in different positions to create images from different vantage points. Given the two images, a stereo vision algorithm solves the problem of estimating depth. This is explained in the camera calibration section.

4.3 Camera Calibration Theory

Camera Calibration is the process of determining the camera parameters. In the calibration process, two different sets of parameters are estimated: the extrinsic and intrinsic parameters. The extrinsic parameters are used to transform the three-dimensional world coordinate system to the three-dimensional camera coordinate system; while the intrinsic parameters are used for a projective transformation of the three-dimensional camera coordinate system into the two-dimensional image coordinates. (Keep in mind that the camera model that is being considered here is the pinhole camera model, where it is assumed that a camera is essentially a pinhole lying at the front of an image plane.) Since the image formed inside the camera is on a 2D plane of pixels, and the object in front of the camera is in 3D continuous space, there must be a projection from 3D continuous-valued vector space to 2D discrete-valued vector space. The camera image loses raw 3D information, and quantization noise is also introduced. Nonetheless, a 3D 'sense' from the 2D plane image can be acquired. The *pinhole* model simplifies camera calibration calculation since it involves only linear systems based calculation. However, there is no lens involved in the model. Therefore, the model does not account for lens-based distortions.

4.3.1 Coordinate Systems

As there are changes of spaces throughout the process, there are three coordinate systems involved in the process as shown in Figure 4.3.

- **World coordinate system** refers to the coordinate system where the actual physical object resides and is pointed with reference to. The 3D object in the real world gets its coordinates from this coordinate system.

Figure 4.3: Coordinate Systems involved in Pinhole Camera Calibration System

- **Camera coordinate system** refers to the 3D coordinate system the camera refers to, while getting the information of the object within its cone of vision. However, in the 3D coordinate system the Z plane is withdrawn, as it always refers to the image plane. Therefore, transformation is considered for only $x - y$ plane.

- **Pixel Coordinate System** refers to the 2D coordinates system to locate each pixel on the image plane inside the camera. Since it refers to only pixel location, it is quantized.

Basically, during image formation, the world coordinate gets transformed to the camera coordinates by the camera matrix and camera matrix gets transformed to image coordinates by perspective projection [ZZ14].

Camera Matrix Transformation

As mentioned above, the world coordinate system is at first transformed to camera coordinates by the help of camera matrix. If a point in 3D World Coordinate System is denoted as X_w, Y_w, Z_w (point X, Y, Z in figure 2) and projected inside the camera plane as X_c, Y_c (point C in Figure 4.4) and camera matrix is denoted as P, then the relationship can be denoted by Equation 4.1.

$$\begin{bmatrix} X_c & Y_C & 1 \end{bmatrix} = \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix} P \tag{4.1}$$

The dimensionality augmentation in Equation 4.1 is because of the inherent suppression of the Z_c and because of the nature of the camera matrix P, which is explained later.

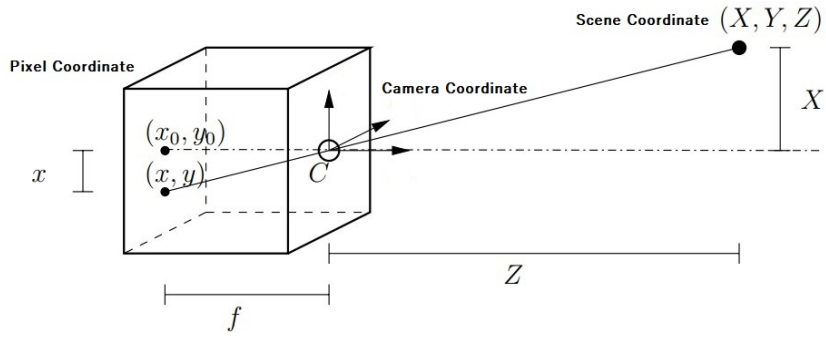

Figure 4.4: Graphical presentation of camera matrix transformation.

Perspective Projection

After the transformation to camera coordinate system, the camera coordinates are transformed to pixel coordinates by perspective projection. It follows the principal of similar triangle illustrated in Figure 4.5, thus, if x, y are pixel coordinates, then, from Equation 4.2 one can get the pixel values of the image inside the camera.

$$x = f\frac{X_c}{Z}, y = f\frac{Y_c}{Z} \tag{4.2}$$

Since x, y are discrete, Equation 4.2 becomes:

$$x = \left\lfloor f\frac{X_c}{Z} \right\rfloor, y = \left\lfloor f\frac{Y_c}{Z} \right\rfloor$$

4.3.2 Camera Parameters

Camera parameters are of two kinds, extrinsic and intrinsic. Extrinsic parameters are the ones related to the orientation of the camera and subject to change due to any rotation and translation of the camera. Intrinsic parameters are the ones that are related to internal parameters of the camera. Together they form the camera matrix P.

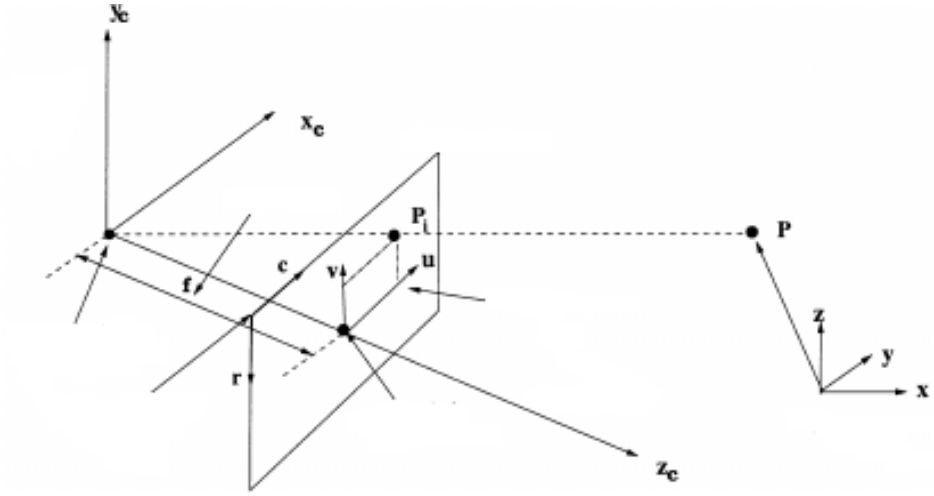

Figure 4.5: Graphical representation of perspective projection [HZ03].

Extrinsic Parameters

Extrinsic parameters (sometimes called external parameters): orientation (rotation) and location (translation) of the camera. If Rotation matrix is denoted as R and Translation matrix is denoted as t, then Extrinsic parameter matrix is denoted as, $R|t$. If R has the parameters R_x, R_y, R_Z denoting rotation around X,Y and Z axis and t has the parameters, t_x, t_y, t_Z denoting, translation through X,Y and Z axis, then $R|t$ can be expanded as,

$$\begin{bmatrix} R_x & 0 & 0 & t_x \\ 0 & R_y & 0 & t_y \\ 0 & 0 & R_y & t_z \end{bmatrix} \tag{4.3}$$

Intrinsic Parameters

The camera intrinsic parameters are the following.

- f focal length is basic description of photographic lens, sometimes referred as camera constant.

- γ aspect ratio models the non-quadratic light-sensitive elements of the camera.
- s skew models the non-rectangular light-sensitive elements.
- x_0, y_0 principal point, which is essentially the orthogonal projection of the focal point onto the image plane.

  Together they form the intrinsic camera matrix:

$$\begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.4}$$

Thus, the camera matrix P can be expressed as Equation 4.5:

$$P = \begin{bmatrix} R_x & 0 & 0 & t_x \\ 0 & R_y & 0 & t_y \\ 0 & 0 & R_y & t_z \end{bmatrix} \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

By recursive substitution from Equation 4.5 to 4.3.1 and then to Equation 4.1, we get the complete camera equation.

4.4 Practical Camera Calibration Processes

In practice, the camera parameters are calibrated with the help of known pattern objects. Chessboard is an object where each square is in equal distance. Therefore, referencing to each coordinate in the system would be simple calculation to put into the camera equation.

For ideal condition, γ is set to 1 and s is set to 0, thus the camera intrinsic matrix simplifies to

$$\begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

However, since practical cameras are not ideal pinhole camera, there are distortions involved in camera image for which calibration is required as well. These distortions can be classified as radial and tangential.

Figure 4.6: Chessboard based calibration [BK00].

Radial distortion occurs when light rays bend more near the edges of a lens than they do at its optical center [MAT10]. The smaller the lens, the greater the distortion.

If x, y represents distorted camera coordinates and r is the radial distance of the camera lens, then the correctional coordinates x_{cor}, y_{cor} is denoted by Equation 4.7:

$$\begin{aligned} x_{cor} &= x1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \\ y_{cor} &= y1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \end{aligned} \tag{4.7}$$

Tangential distortion occurs when the lens and the image plane are not parallel. The distortion coefficients model this type of distortion [MAT10], as shown in Equation 4.8:

$$\begin{aligned} x_{cor} &= x + 2p_1 xy + p_2 r^2 + 2x^2 \\ y_{cor} &= x + 2p_2 xy + p_1 r^2 + 2y^2 \end{aligned} \tag{4.8}$$

where p_1, p_2 are tangential distortion coefficients.

Therefore, the parameters k_1, k_2, k_3, p_1, p_2 are first determined in order to compensate for the apparent and actual pixel value. After that, x_{cor}, y_{cor} are used for later steps in the process.

P is calculated by considering multiplication resultant equivalency of

each entry. By considering p_{ij} to be the element for each entry Equation 4.5 becomes:

$$\begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{14} \end{pmatrix} = \begin{bmatrix} R_x & 0 & 0 & t_x \\ 0 & R_y & 0 & t_y \\ 0 & 0 & R_y & t_z \end{bmatrix} \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Substituting the value from Equation 4.8 with Equation 4.1, the final camera calibration equation becomes Equation 4.9:

$$X_c, Y_c, 1 = X_w, Y_w, Z_w, 1 \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{14} \end{pmatrix} \tag{4.9}$$

which can be solved by 12 linearly independent equations acquired from different experimental values of X_w, Y_w, Z_w.

4.5 Stereo Camera Calibration

Equation 4.9 is the baseline to calibrate camera with only one lens. Stereo camera, on the contrary, refers to a system of camera where multiple lenses are included or two cameras are mounted together by a small distance [Nev76]. One intuitive way to calibrate stereo camera is by calibrating each lens individually using the pinhole model. However, this has some certain limitations of disparity [Lan+10] and reconstruction [AA90]. Additionally, while constructing the camera system, it may be possible that the system consisting of multiple cameras fused as a single lens [GG93; LK00]. A better approach is the use of *epipolar geometry* [JKS95].

In epipolar geometric model, each pinhole camera model derived and calibrated from the above mentioned single camera calibration technique is called a *fundamental matrix*. If there are two lenses, then there will be two fundamental matrices. Later, these two fundamental matrices are joined by two symmetric matrices, and they form another matrix called the *essential matrix* [JKS95].

Let two pinhole cameras (left and right) be defined by their fundamental matrix K_l and K_r where $K_l = P_l I_3, 0$ and $K_r = P_r R, t$ are in a homogeneous

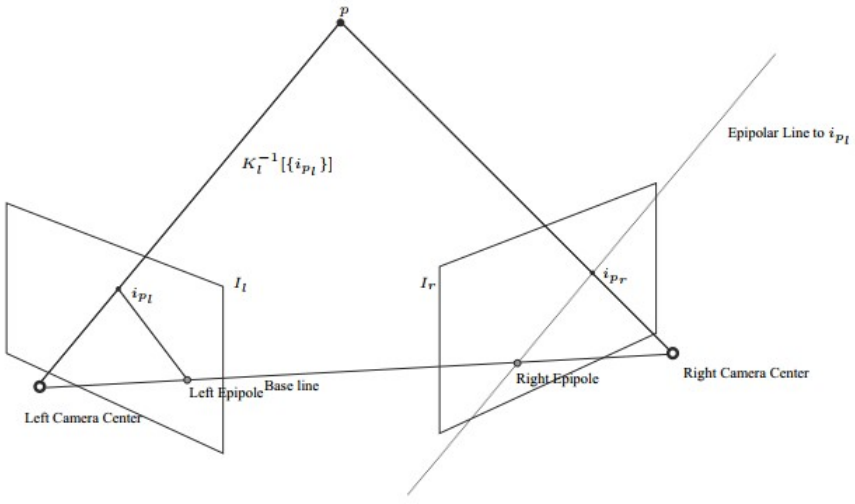

Figure 4.7: Graphical representation of epipolar geometry.

frame of reference. Camera coordinates of the left camera is coincident with the frame of reference of the homogeneous frame of reference. The skew symmetric transform matrix is defined as in Equation 4.10

$$Q_t = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix} \tag{4.10}$$

where $t = t_1, t_2, t_3 \in R^3$, If we defined $Q_t x = Q_t \times x$ as the cross product of two vectors, then, Equation 4.11 and 4.12 are the definitions of fundamental matrix F and essential matrix E.

$$F = \left(P_r^{-1}\right)^t Q_t R P_l^{-1} \in \mathbb{R}^{3\times 3} \tag{4.11}$$

$$E = Q_t R \tag{4.12}$$

Because of the cross product and the multi matrix multiplication involved, it is almost impossible to solve Equation 4.11 and 4.12 analytically. Therefore, the numerical techniques are the best approaches to solve them.

Some of the key points that are kept in mind regarding the fundamentals of epipolar geometry while designing these numerical techniques are the following such as,

- When two cameras are involved in general position and orientation, the image points lie along the intersection line of the epipolar plane.
- Every point in the left image, has a correspondence with each point in the right image. This is referred to as *correspondence problem* [OA05].

Correspondence Problem

Correspondence Problem is the most fundamental concern in stereo camera calibration. The old school approach follows the principal of correlation-based mutual information minimization. If I_l and I_r are image templates obtained from right and left images, then sum of squares difference between the image pairs can be defined in Equation 4.13

$$C = \sum_{i,j} \left[I_l(x_i, y_j) - I_r(x_i, y_j) \right]^2 \tag{4.13}$$

And the mutual information of the image point random variables X and Y are defined in Equation 4.14

$$IX, Y = E_{X,Y} \left[\log \left(\frac{PX, Y}{PXPY} \right) \right] \tag{4.14}$$

The objective is to minimize the mutual information by iterative obtaining of the parameter C, so that each C in every iteration gives the minimum IX, Y. However, with the advancement of image processing techniques, researchers have looked into ways to solve the correspondence problem more in frequency domain, such as wavelet domain [SB07]. Pixel domain researchers are also finding newer parameters to optimize the stereo camera parameters, such as Sum of Absolute Differences [HAN10]. In recent years, handcrafted feature based computation [WCH+92] are giving way to neural network based models. Therefore, several neural network based approach has also been developed to solve the single lens as well as stereo camera calibration problem [AHF99; Do99].

Disparity Estimation and Semi Global Matching

Despite the robust neural network-based model to estimate camera parameters, there are other imperfection measurements to implement when it comes to FPGA and other embedded camera systems, because of space and performance constraints. One such measurement is camera disparity, which refers to the distance between two points in the left and right image template [Qia97].

Measurement of disparity plays an important role in various applications such as automated driving and robotics [Mat89]. Two approaches are intuitive in measuring disparity: the local method and global method. The local method refers to the normalized cross correlation-based method [HLL10] of pixel matching. However, the method suffers from several drawbacks such as false assumption of perfect similarity distance between image templates and epipolar plane and correlation error due to depth discontinuity [Hir11].

A more robust and manageable approach is the *Semi-Global Matching* approach, which works by matching the neighborhood pixels of each image templates containing the best similarity. The algorithm [Hir11] takes two images from the left and right camera and performs matching cost calculation, directional cost calculation, and post-processing. For matching cost calculation, Hamming Distance between the Center-Symmetric Census Transform (CSCT) is one of the most robust metrics in the literature [SLR13].

4.6 Point Clouds

Point clouds are one of the frequently used datasets to represent objects or space. A single point on a sampled surface in the cloud can be represented by its geometric coordinates X, Y, and Z. Point clouds are a convenient way of containing a large number of spatial records into a dataset that can then represent a whole. If colour information is provided, the point cloud becomes 4D.

Point clouds are usually generated using 3D laser scanners, LiDARs, or stereo cameras. The 3D scans are then put together, forming a full capture of a scene, using a procedure called *registration*, whereas in case of stereo

cameras, 3D point coordinates are retrieved by stereo imagery. As this thesis implements methods that are only manipulating the point clouds, a complete description of point cloud generation won't be described. Instead, the manipulation operations will be explained.

4.6.1 Thresholding

Thresholding is one of the most basic point operations in computer vision and is used to segment points based on the specified dimensional intervals. Since thresholding presents a simple way to segment images, it is often used in computer vision applications as a preprocessing step for reducing noise or finding foreground objects. It is also relatively cheap in computational terms since it only needs one comparison per pixel. In a simple form, thresholding is applied to an image given by the intensity fx, y at pixel position x, y. It groups intensity values into a background class and a foreground class based on a threshold T using the following equation [GWE04]:

$$gx, y = \begin{cases} 1 & \text{if } fx, y > T \\ 0 & \text{if } fx, y \leq T \end{cases} \tag{4.15}$$

The result of this operation is a binary image, where all pixel intensities above the threshold are set to 1 and all below to 0. A simple application for thresholding is to convert grayscale images to binary images. For a well-chosen threshold T the grayscale image with intensities $\{0, 1, \ldots, 255\}$ is mapped into a binary image of intensities $\{0, 1\}$. There are several algorithms in the literature that are used to apply thresholding. This section briefly informs the reader about these.

According to an extensive survey on thresholding algorithms conducted by Sezgin et al. [SS04], there are 6 categories in thresholding methods. The histogram shape-based method analyzes the peaks, valleys and curvatures of the smoothed histogram. The pixels' intensity is used, and certain assumptions are made on the properties of the histogram (e.g., bimodal) [Kaw+18]. As an example, the minimum algorithm takes a histogram of the image and smooths it many times until there are only two peaks in the histogram, as shown in Figure 4.8" [Van+14]. Next, there are clustering-based methods, where the gray-level samples are clustered in

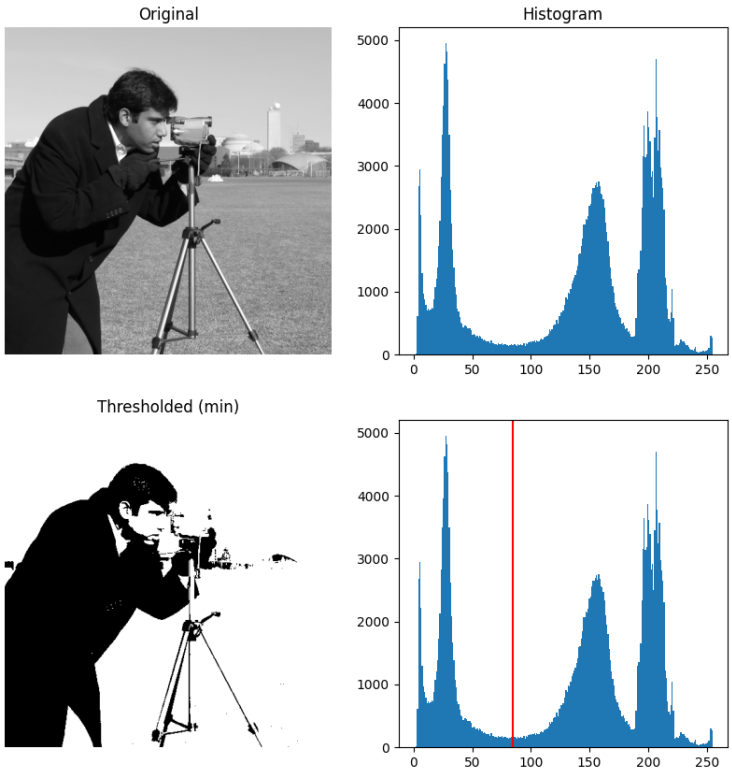

Figure 4.8: Histogram shape-based thresholding applied using minimum algorithm on a bimodel image[Van+14].

two parts as background and foreground (object), or are modeled as a mixture of two Gaussians [Kwo04]. Entropy-based methods [ZW11] result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binarized image, etc. Object Attribute-based methods [Rai20] search a measure of similarity between the gray-level and the binarized images, such as fuzzy shape similarity,

edge coincidence, etc. Spatial methods [CYV00], on the other hand, are reported as "Spatial methods use higher-order probability distribution and/or correlation between pixels. Last of all the classical local methods adapt the threshold value on each pixel to the local image characteristics. In these methods, a different T is selected for each pixel in the image" (Sezgin, 2014, p.2).

In many cases, instead of finding a threshold by hand, the threshold can also be calculated automatically based on the information given in the image in an unsupervised manner. This makes the thresholding process more computationally expensive nevertheless automates the binarization into fore- and background. For automatic thresholding there exist global methods that apply a single threshold to the whole image as well as local methods that threshold different image parts on locally computed thresholds. Global methods are often histogram-based, these methods calculate a histogram of the intensity values of the image. Based on this histogram a threshold value is chosen either through statistical analysis or the shape of the histogram. These methods work well when there is a bimodal distribution such that most intensity values are spread narrow around two unique peaks that are most distant from each other.

A simple global histogram-based algorithm for automatic thresholding is Otsu's Method [Ots79]. The basic idea of Otsu's method is to split the intensity values of the image into two classes. An optimal threshold should separate these classes maximally such that a clear distinction between fore- and background intensities is reached. Separation is quantified by the between-class variance, which will be maximized by Otsu's method. This method is used in cases where simple bimodal distribution with two homogeneous classes don't exist. Thus, if there is no valley, one method of determining T is to minimize the total variance within both classes, which is the main idea behind Otsu's method.

More details of Otsu's method are found in the original paper [Ots79] and [GWE04], which also features a broader introduction to image segmentation by thresholding.

4.6.2 Clustering

Clustering or cluster analysis classifies data points into groups called *clusters* based on their properties. It can also be seen as unsupervised

classification of data, as clustering works without prior knowledge about the given data objects. This makes clustering a powerful tool for structuring the unstructured data of stereo system (e.g., yielding point clouds). This is particularly interesting for point cloud processing, since clustering offers a robust way to group points, which may belong to the same entity. Imagine a point cloud of the scene consisting of points that belong to objects, and gaps between them. The point cloud could be segmented by using appropriate clustering algorithms. The points belonging to different objects and gaps have different features that could be utilized by these clustering algorithms.

There are plenty of clustering algorithms available in the literature, an extensive survey on the matter was conducted by Rokach et al. [Rok09]. However, this subsection is only interested in three commonly used algorithms: K-Means [Mac+67], DBSCAN [Est+96a], HDBSCAN [MHA17a].

K-Means

The basic idea behind K-Means is to cluster points into groups based on their distance to a mean point, the centroid [Mac+67]. The first step is to choose K random centroid points, which will form the K clusters. Next, the distance from every point to each centroid is computed, and the point is assigned to the cluster of the closest centroid. Given the assigned data points, the centroid is updated by taking the mean of the points in the cluster. This moves the centroids in the direction of the center of a group of points. Assignment and update steps are repeated until the centroids have found their final position and don't change anymore. In a best case result, the centroids lie in each cluster center such that the data is split in a reasonable way.

The K-Means algorithm is used to partition a given set of observations into a predefined amount of k clusters. The algorithm as described in Macqueen et al. [Mac+67] starts with a random set of k center-points (μ). During each update step, all observations x are assigned to their nearest center-point (see equation 4.16). "In the standard algorithm, only one assignment to one center is possible. If multiple centers have the same distance to the observation, a random one is chosen" (Rao, 2020, p. 21) [Rao+20].

$$S_i^t = \{x_p : \|x_p - \mu_i^t\|^2 \leq \|x_p - \mu_j^t\|^2 \; \forall j, 1 \leq j \leq k\} \tag{4.16}$$

Afterwards, the center-points are re-positioned by calculating the mean of the assigned observations to the respective center-points (see Equation 4.17).

$$\mu_i^{t+1} = \frac{1}{|S_i^t|} \sum_{x_j \in S_i^t} x_j \tag{4.17}$$

The update process repeats until all observations remain at the assigned center-points. Therefore, the center-points would not be updated anymore. This means that the K-Means algorithm tries to optimize the objective function shown in Equation 4.18. K-Means always ends in a local minimum, since there is only a finite number of possible assignments for the number of centroids and observations available, and each iteration has to result in a better solution [MKS18].

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{4.18}$$

$$\text{with } r_{nk} = \begin{cases} 1 & x_n \in S_k \\ 0 & \text{otherwise} \end{cases}$$

The main issue with K-Means is its dependency on the initially chosen centroids. It could be that the centroids are dragged by outliers, or outliers might get their own cluster instead of being ignored.

The most prevalent way is to perform multiple clusterings with different start positions [Sch15]. Afterwards, the clustering that occurred most is treated as correct. Another, newer approach is the so-called K-Means++ by Arthur and Vassilvitskii [AV06]. This extension to the K-Means algorithm tries to distribute the initial centroids over the given data to minimize the probability of bad outcomes. The initial points are set according to the authors by the following steps:

1. Take a random data point from the data X and mark it as centroid c_1
2. Choose another centroid c_i with the probability $\frac{Dx^2}{\sum_{x \in X} Dx^2}$ where Dx

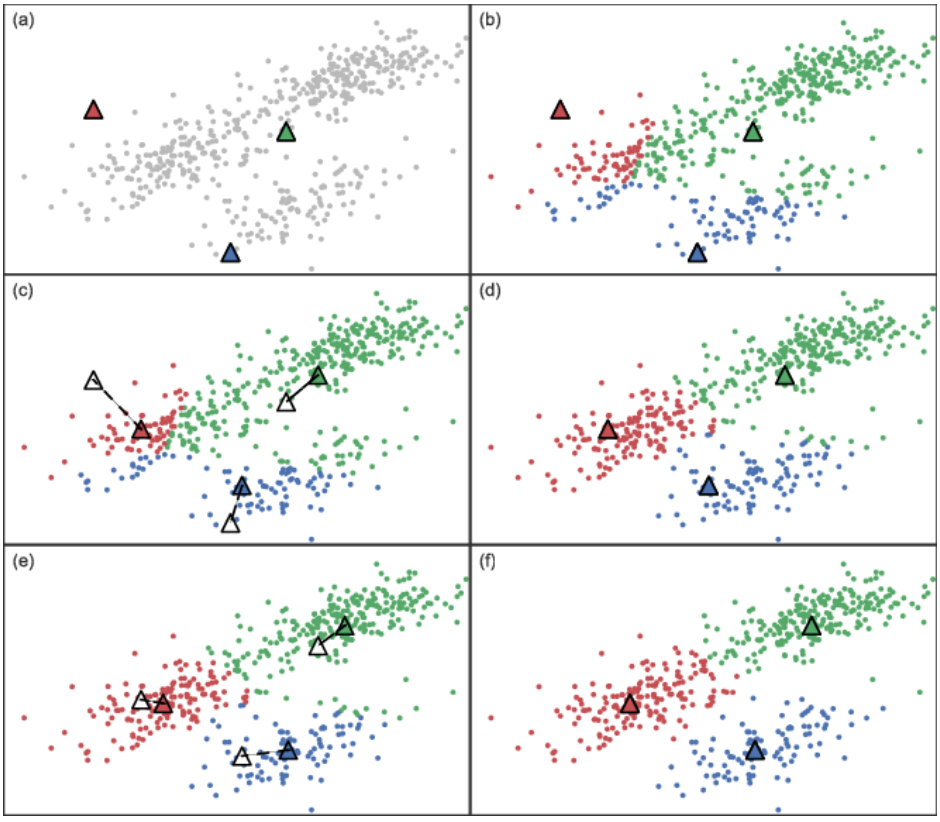

Figure 4.9: K-Means example in 2D space. (a) Initialized cluster centroids at random positions (b) Each data point is mapped to the closest centroid. (c) The centroids are moved to the mean of each cluster's mapped points. (d, e, f) Until convergence, the steps are repeated [BPP17].

denotes the shortest distance from the data point x to its closest, already chosen centroid.

3. Repeat 2. until all k initial centroids are chosen.

Afterwards, the standard K-Means algorithm as described above is performed. The authors also showed that with this initialization algorithm, K-Means++ can be approximately computed in $O \log n$, compared to $On^{dk+1} \log n$ for the standard algorithm. However, K-Means has a major

disadvantage compared to other clustering algorithms. The hyperparameter K has to be chosen before clustering. Applications where the number of clusters is known in advance (e.g., where the number of entities in the scene are fixed), K-Means can be a good fit. Otherwise, more advanced algorithms such as DBSCAN and HDBSCAN can solve the problem by finding the number of clusters based on the data structure.

DBSCAN

DBSCAN is a density-based clustering algorithm proposed by Martin Ester et al. [Est+96b]. A density-based clustering algorithm views clusters as regions of densely packed points in contrast to K-Means, a prototype-based algorithm focusing on the distance between individual points and the prototype, the centroid.

DBSCAN estimates the density of a point by counting the points in the neighborhood with a radius of *eps* around this point. Choosing *eps* right is important since a too-small *eps* only includes the point itself, while a too-big *eps* could include all points. Using the density, DBSCAN classifies points as *Core*, *Border* or *Noise* points. Core points are points in the middle of the cluster that fulfil the requirement that they have at least a minimum number of points $MinPts$ in their neighborhood of *eps*. A border point is a point which is in the radius of a core point but has less than $MinPts$ points in his neighborhood. Noise points are all other points that are neither core or border points.

Given the point classification, DBSCAN builds clusters by grouping core and border points. First, all noise points are eliminated. Then any two core points, if they are within a distance of *eps*, are assigned the same cluster. Finally, all border points that are in distance of a core point are grouped into the same cluster as the core point.

One of the biggest strengths of DBSCAN is the ability to handle noise. Noisy regions without many data points will have a low density and therefore are filtered by density-based approaches. Besides the built-in denoising, DBSCAN can handle clusters of variable sizes and shapes. In contrast to K-Means, the number of clusters is not needed as a hyperparameter. An average runtime of $\mathcal{O}n \log n$ [Tan+18] can be reached, with meaningful values for *eps* and $MinPts$ and the use of indexing structures.

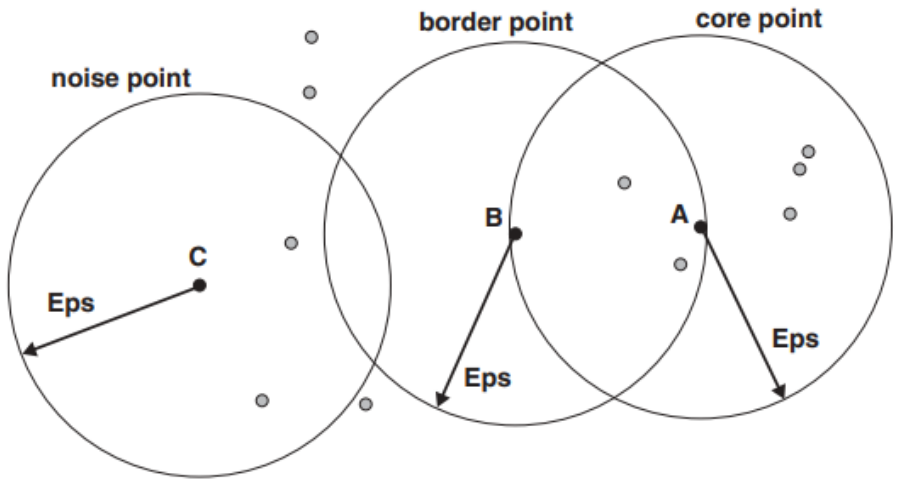

Figure 4.10: The three point types used by DBSCAN: Core, Border and Noise points [Tan+18].

HDBSCAN

DBSCAN relies heavily on finding a fitting value for the *eps* parameter. For low-dimensional data this is done by adjusting *eps* until a satisfying result is reached. For higher-dimensional data this is often not possible since it is hard to visualize the data and unclear in which way data points are similar. Hierarchical density-based spatial clustering of applications with noise (HDBSCAN) [CMS13] extends DBSCAN to a hierarchical clustering method, that finds the *eps* distance parameter automatically by building a hierarchy of all possible distance values. An example of this hierarchy is seen in Figure 4.11.

The dendrogram shows the size and arrangement of clusters for different distance values. Clusters are color-coded in their number of points with the biggest clusters in yellow to the smallest clusters in purple. A cluster always contains all clusters below itself. For instance, at a distance of 0.0 all points are noise points since no point has a second point in its *eps* distance. With increasing distance, points can be classified as core and border points, and by doing so, clusters are built. Clusters are merged

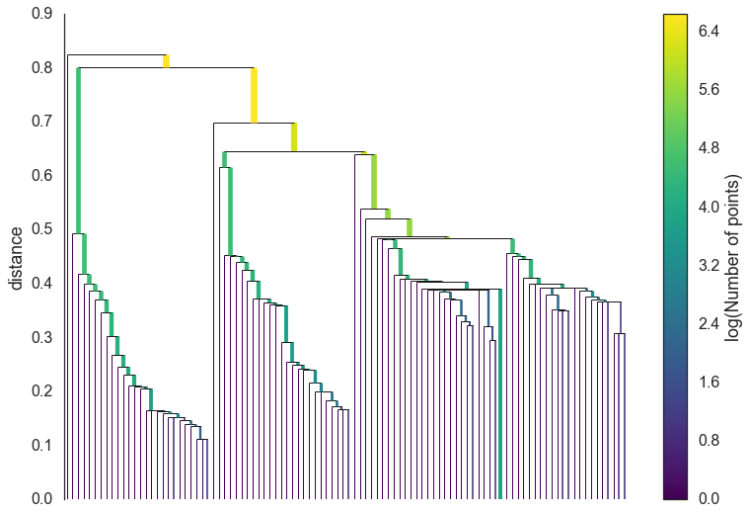

Figure 4.11: Dendrogram of clusters for different [VD09].

with increasing distance until all points are grouped into a single cluster. Additionally, the diagram in Figure 4.12 illustrates the labeled result of hierarchical clustering.

In the case of DBSCAN, a threshold distance *eps* is given on which clusters are chosen. This can be visualized by a horizontal line at that distance in the dendrogram. All clusters hit by this line are the result arrangement of DBSCAN. HDBSCAN takes a different approach, instead of using a global threshold to choose the clusters, it selects clusters from different distances, while fulfilling the constraint that it can't choose a cluster that is below an already selected cluster. Clusters are chosen based on their persistence over different distances. A cluster that holds over multiple distance values should contain points that are very similar, while a cluster that merges quickly probably contains loosely related points. To reflect the persistence, HDBSCAN computes a stability value for each cluster based on their length of existence and their number of points.

Visual illustrations of input and output data regarding this example can be viewed in Figures 7.9 and 7.10 respectively, and Appendix 7.1. Additionally, more information on HDBSCAN is found in the paper by Campello et al. [CMS13], where they also show optimizations of HDBSCAN

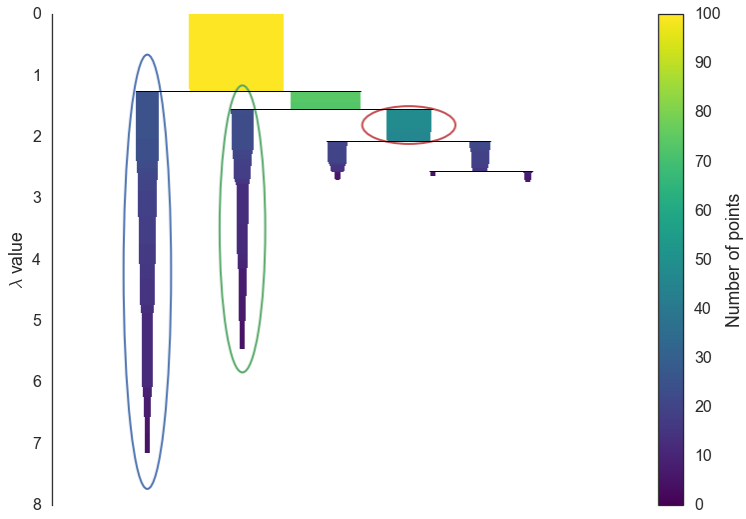

Figure 4.12: Labeled clusters as a result of HDBSCAN [VD09].

to achieve a runtime of On^2. A recent work [MHA17b] further optimized the algorithm to a runtime of $Onlogn$.

4.7 Machine Learning

According to the definition of Tom Mitchell in 1997, machine learning is the area of artificial intelligence with a focus on computer algorithms that improve automatically through experience [Mit+97]. Arthur Samuel, who is a pioneer in machine learning, conducted his research to find an answer to a very fundamental question of that time. The effort was to figure out how computers could learn to solve problems without being explicitly programmed [Sam59]. Today, we define machine learning algorithms that build models based on sample data, known as *training data*, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a variety of domains and applications today, ranging from online commercials to autonomous driving, where it is challenging or unsuitable to develop conventional algorithms to perform the required tasks.

There are three main categories of learning this thesis considers to

investigate under machine learning. It is important to define them before going any further. In case of the first category, *supervised learning*, one has input data that has been labeled. This is similar to us writing a name on the back of a photograph, so that others or even ourselves can identify the person on that photograph after some time. In technical terms, the algorithm is first presented with training data of examples including the inputs and the desired outputs, thus allowing it to learn a function. One has to keep in mind that in order to perform supervised learning, one needs a labeled dataset. These conditions are not always met. One does not always have a labeled dataset for the predictions he seeks to make. This situation brings up the second category of learning, *unsupervised learning*. This refers to learning without a ground truth such as labels to correct the error the model makes when guessing. In technical terms, the algorithm is given samples from the input space only, and a model is fit to these observations. A very common example here would be a clustering algorithm. The last category is *reinforcement learning*. In this paradigm, an agent explores an environment and at the end receives a reward, which may be either positive or negative. In effect, the agent is told whether he was right or wrong, though not how. One could give examples from daily lives of us humans to solidify this definition. Playing a game of chess could be one of these examples, since we don't know whether we have won or lost until the very end. Another one could be a waiter in a restaurant, where he has to wait until the end of the meal as only then he becomes aware of whether or not a tip is involved. Much more could be written about reinforcement learning. However, since this thesis only considered supervised learning, it rather skips the details of reinforcement learning.

4.7.1 Neural Networks

Another algorithmic approach from the early machine-learning community, namely artificial neural networks, came and mostly went over the decades. "Neural networks are inspired by our understanding of the biology of our brains - all those interconnections between the neurons. However, unlike a biological brain where any neuron can connect to any other neuron within

36 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

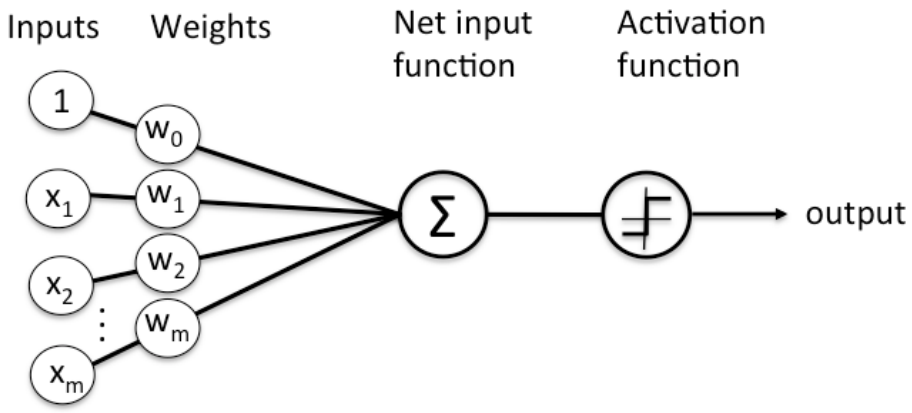

Figure 4.13: Graphical illustration of a neural network node [36].

a certain physical distance, artificial neural networks (ANNs) have discrete layers, connections, and directions of data propagation" (Copeland, 2016, p.2) [37]. In short, ANNs try to simulate the connected networks of neurons in the human brain and enable computers to behave as interconnected cells. It's crucial to state, however, that the neural networks are not learning the information the way human brain does, they are rather only inspired by the biological neural networks. Diverse elements of the human brain are tasked with processing various pieces of information. These elements of the brain are organized hierarchically, or as one can also say, in layers. This way, as information flows through the brain, each level of neurons processes it to provide intuition, and eventually passes it to the senior layer. It is this described layered model of processing information and making decisions that artificial neural networks try to simulate. This model presents three layers of neurons: *the input layer* (where the data enters the system), *the hidden layer* (where the information is handled) and *the output layer* (where it is decided what to do based on the data), as illustrated in Figure 4.13. However, artificial neural networks can get more complex than this described model, and include multiple hidden layers.

37 Copeland, M. (2019, July 29). The difference between AI, machine learning, and deep learning. Retrieved February 25, 2021, from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

The aforementioned term *layers* are made of *nodes*. "A node is just a place where computation happens, loosely patterned on a neuron in the human brain, which fires when it encounters sufficient stimuli. A node combines the input from the data with a set of coefficients or weights. These either amplify or dampen that input, thereby assigning significance to inputs with regard to the task the algorithm is trying to learn (e.g., which input is most helpful is classifying data without error?). These input-weight products are summed, and the sum is passed through a node's so-called activation function. This is done to determine whether and to what extent that signal should progress further through the network to affect the outcome (e.g., classification). If the signals pass through, the neuron is 'activated'" (Nicholsan, 2021, p.2) [38]. Especially in the field of learning neural networks, much progress has been made over the last years [KP18; Lit+17; Wan+19; Zha+17]. However, this thesis does not go deeper into the basics of neural networks, it rather investigates the topic of deep learning. This forms the basis of a few employed blocks described in Chapter 5.

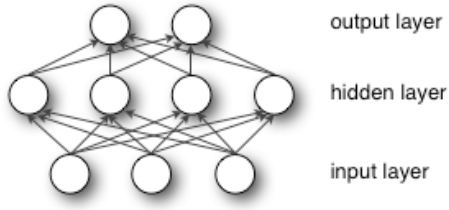

Figure 4.14: Graphical illustration of a neural network with layers [39].

4.7.2 Deep Learning

Definition of deep learning is found in many sources. One of the prominent authors in the field, Ian Goodfellow [Goo+16] (a researcher at OpenAI), stated "Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy

38 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

39 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

of concepts. Since the computer gathers knowledge from experience, there is no need for a human computer operator to formally specify all of the knowledge needed by the computer. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones; a graph of these hierarchies would be many layers deep. Deep-learning networks are distinguished from the more common place single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multi-step process of pattern recognition" (Goodfellow, 2016, p.34).

Initial versions of neural networks (e.g., the first perceptrons) were quite shallow. They were only composed of one input and one output layer, and at most one hidden layer in between (see Figure 4.14). More than three layers (including input and output) qualifies as 'deep' learning. "In deep learning networks, each layer of nodes trains on a distinct set of features based on the previous layer's output. The further one advances into the neural network, the more complex features the nodes can recognize since they aggregate and recombine features from the previous layer. This is known as *feature hierarchy,* and it is a hierarchy of increasing complexity and abstraction. It makes deep learning networks capable of handling very large, high-dimensional datasets with billions of parameters that pass through non-linear functions" (Nicholsan, 2021, p.4) [40].

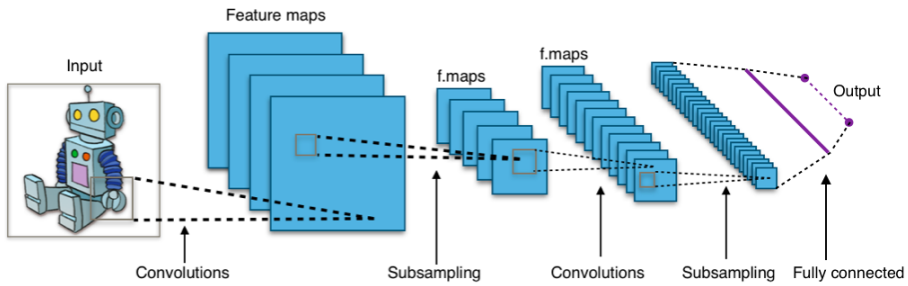

Figure 4.15: Graphical illustration of a convolutional neural network [Aph].

A *deep convolutional neural network* (DCNN) is a feed-forward neural network that is designed to process structured data such as images or audio.

40 Nicholson, C. (n.d.). A beginner's guide to neural networks and deep learning. Retrieved February 25, 2021, from https://wiki.pathmind.com/neural-network

The effectiveness of a DCNN comes from convolutional layers which are efficient in picking up patterns. In Figure 4.15 the structure of a DCNN is shown. In the beginning, the DCNN receives an input, which is an RGB image. On each part of the image, kernels are applied by performing convolutions. In this process, every kernel produces a downsampled version of the image, which is referred to as a *feature map*. The kernel output is then often fed into an activation function before it is sub-sampled even more by pooling operations. If this process is repeated, kernels are applied on the feature maps (usually the number of kernels applied on the feature maps increases over in each step), resulting in smaller feature maps. If the feature maps are small enough, they finally get flattened and fed into a fully connected neural network. In this sense, one can perform classification operations on abstract data like RGB images.

A convolution between an image/feature map I and a kernel K is described as

$$I * K_{i,j} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} Ii - m, j - n \cdot Km, n \tag{4.19}$$

where $k_1 \times k_2$ is the dimension of the kernel. In the case of a convolutional neural network, the image has a width W, a height H and a specific number of channels C ($C = 3$ if RGB is considered). Therefore, the image $Iis \in \mathbb{R}^{H \times W \times C}$, and since there are D filters, $K \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$. In addition, there is a bias $b \in \mathbb{R}^D$ for each filter. Hence, the convolution is given by

$$I * K_{i,j} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^{C} I_{i+m,j+n,c} \cdot K_{m,n} + b. \tag{4.20}$$

The process of such a convolution is shown in Figure 4.16. The learning process of a DCNN with L layers follows the same basic principles as the one of a fully connected network. Since neural network predictions y_p will differ from the corresponding ground truth value t_p, the mean squared error

$$E = \frac{1}{2} \sum_p t_p - y_p{}^2 \tag{4.21}$$

needs to be backpropagated from the last layer $l = L$ to the first layer $l = 1$. "While backpropagating the errors, one can then calculate the weight-

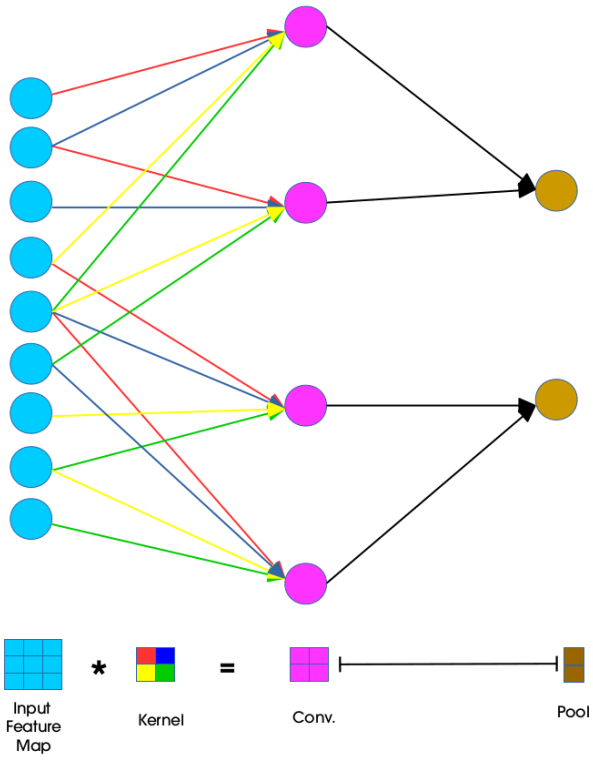

Figure 4.16: Graphical illustration of a forward-passing in a convolutional neural network [Kaf16].

related errors gradient $\frac{\partial E}{\partial w}$. In order to improve the networks accuracy, during training the weights need to be updated in the negative direction of the gradient. In this step, the gradient is weighted with a learning rate α. In the following, we describe a convolution as

$$x^l = x^{l-1} * w^l \tag{4.22}$$

where x^l is the output of the convolution of layer l and x^{l-1} is the output of layer $l-1$. We denote w^l to be the weight matrix between those layers. In order to simplify the explanation of backpropagation, we assume $c = 1$. The process of backpropagation is shown in a simplified way in Figure 4.17.

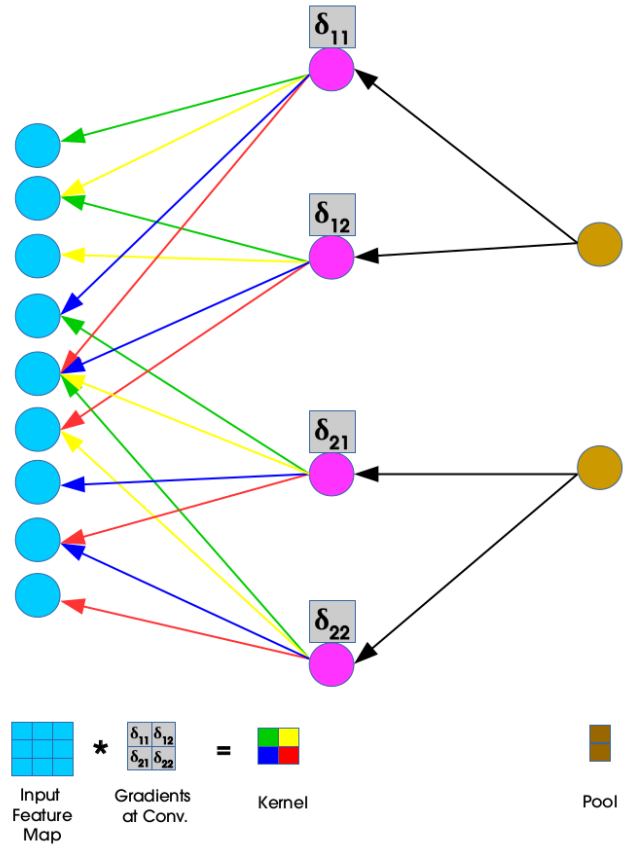

Figure 4.17: Graphical illustration of backpropagation in a convolutional neural network [Kaf16].

The kernel w^l consists of weights $w^l_{m,n}$ which are colored in red, yellow, blue and green (see Figure 4.16). In backpropagation, the gradient of a single weight $w^l_{m',n'}$ of a convolutional kernel is referred to as

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \frac{\partial E}{\partial x^l_{i,j}} \frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}}. \tag{4.23}$$

Here, H and W represent the height and width of the the of the 2-dimensional image x^l of layer l. This image gets convoluted by a kernel of

the shape $k_1 \times k_2$ with $m \in k_1$ and $n \in k_2$. The last expression of the two sums deviates to

$$\frac{\partial x^l_{i,j}}{\partial w^l_{m',n'}} = o^{l-1}_{i+m',j+n'} \tag{4.24}$$

which is the calculated output o of the activation function f in layer $l-1$. However, the question of "How does a single pixel $x^l_{i',j'}$ in the feature map affect the error in layer l?" is yet to be answered. The first expression of the sums of Equation 4.23 denotes as

$$\frac{\partial E}{\partial x^l_{i,j}} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \frac{\partial E}{\partial x^{l+1}_{i'-m,j'-n}} \cdot w^{l+1}_{m,n} \cdot \frac{\partial f(x^l_{i',j'})}{\partial x^l_{i',j'}}. \tag{4.25}$$

The last expression of the previous equation equals the first deviation of the activation function, therefore

$$f' x^l_{i',j'} = \frac{\partial f(x^l_{i',j'})}{\partial x^l_{i',j'}}. \tag{4.26}$$

When back-propagating into layer l, the error matrix

$$\frac{\partial E}{\partial x^{l+1}_{i,j}} \cdot w^{l+1}_{m,n} \tag{4.27}$$

from layer $l+1$ is already known. Therefore, one can finally calculate the gradient, and update the weights" (Trommer, 2020, p. 6-10).

One last issue to clear is to understand how the error is backpropagated through the sub-sampling/pooling layers. In the pooling layers (see Figures 4.16 and 4.17), pooling blocks (shown in purple color) are converted into the so-called winning units (shown in brown color). When forward-passing information, the indices of these units are noted. During the max-pooling, a winning unit consists of the maximum of the pooling block. Thus, during back-propagation, the error is assigned directly to the original maximum value. When using average pooling, on the other hand, the error is weighted by $\frac{1}{N}$ and then contributed to the hole pooling block. N stands for the number of units that are sub-sampled into the winning unit.

DCNNs have made striking differences in many computer vision tasks such as image segmentation, image classification, object detection, and etc.

Although the background information on the mechanics of DCNNs have been shared in this chapter, Chapter 5 goes through the state-of-the-art DCNNs used, and the way they work in detail.

4.7.3 Training Environment/Hardware

Most of the state-of-the-art machine learning algorithms require extraordinary hardware to train or even infer. The memory requirements keep expanding every day, since the DCNNs have become increasingly deep in layers. It is safe to assume that behind every recent machine learning algorithm, there is either a CPU (Central Processing Unit), a GPU (Graphical Processing Unit) or a TPU (Tensor Processing Unit) stressed to train. Since this thesis employs a few DCNNs, it is required to mention the training environments and how they differ in terms of performance. In this subsection, the relative advantages and disadvantages of using CPUs (Intel Xeon[41]), GPUs (Nvidia Tesla P100[42]), TPUs (Google TPU v3[43]) were compared for training machine learning models created using *Tensorflow* and *Keras* (`tf.keras`). There are already studies [Jou+17; Kum+19; WWB19] conducted on this matter. This subsection is going over the concrete results acquired in these studies.

CPUs, GPUs and TPUs are processor units with a particular purpose and architecture. A CPU is designed to solve computational problems in general. Thus, its cache and memory design are assured to be optimal for any programming problem. It executes the instructions for computer programs, handles the essential arithmetic, logic, controlling, and input/output functions of these programs. A GPU is rather designed to accelerate the rendering of graphics. Previously, often the GPU was embedded into the CPU, however, as the machine demand on graphical computation load expanded, it became difficult for the built-in GPUs to accomplish the required tasks in tolerable time intervals. For the past two decades, therefore, standalone GPUs (independent of CPU) are preferred by the machine learning community due to their performance on the required

41 https://www.intel.com/content/www/us/en/products/processors/xeon.html
42 https://www.nvidia.com/en-us/data-center/tesla-p100/
43 https://cloud.google.com/tpu/

tasks. [44] Besides, computer hardware manufacturers such as NVIDIA[45] and AMD[46] valued the demands of the machine learning community by explicitly designing GPUs that are meant for machine learning tasks. In 2015, however, a new paradigm paved the way to an even better performing unit called TPU by Google, which was made public recently in 2018. TPU is an application-oriented integrated circuit used specifically to empower the underlying AI calculations. Since TPU is designed to accelerate deep learning tasks develop using TensorFlow programming framework (owned by Google), compilers have not been developed for TPU which could be used for general purpose programming. Therefore, it requires significant effort to do general programming on TPU, making it a specifically deep learning hardware. TPUs are now open for public use on Google Colab [Bis19] environment, which is a cloud-based service to train DCNNs using either CPU, GPU or TPU. While CPUs and GPUs are manufactured by several companies, TPUs for now are only manufactured by Google, making them unique in this regard.

It's not within this thesis' scope to explain how each of the processors works in detail. However, it is required to underline the differences of each in the context of machine learning. These processors can be compared based on several aspects such memory subsystem architecture, compute primitive, performance, purpose, usage and manufacturers. The most relevant aspect here is the *compute primitive*, which refers to the smallest unit used in the processors, as illustrated in Figure 4.18. CPU utilises a 1×1 *scalar* data unit, wheras for a GPU, this is a $1 \times N$ *vector* data unit. TPU, as the name suggests, utilises the data unit *tensor* which is $N \times N$. As the compute primitive allows more flexibility in calculations, better performance is shown by the processor. This fact alone presents that TPU is definitely the fastest in terms of computation. However, knowing this alone is not sufficient in machine learning context. Therefore, the aforementioned studies conducted experimental evaluations to find out how much faster TPU would be, compared to GPU and CPU. These evaluations

44 Gaonkar, A. (2020, July 15). CPU / GPU/ Tpu - ML perspective. Retrieved February 23, 2021, from https://medium.com/analytics-vidhya/cpu-gpu-tpu-ml-perspective-1f049cd4d43d

45 https://www.nvidia.com/en-us/training/

46 https://www.amd.com/en/graphics/servers-radeon-instinct-deep-learning

are even made publicly available[47] through Kaggle [48] platform, which is the world's largest data science community with powerful tools and resources to achieve data science goals. Kaggle provides users with a platform where they can find and publish data sets, explore and build models in a web-based data-science environment. Moreover, it allows users to collaborate with other data scientists and machine learning engineers from all over the world, and enter international competitions to solve challenging data science problems.

In order to compare the performance of these different processors for accomplishing common data science tasks, the data scientists preferred to use the *tf_flowers* dataset to train a convolutional neural network, and then the exact same code was run three times using the three aforementioned different backends (GPU: NVIDIA P100, CPU: Intel Xeon 2GHz (2 core), TPU: TPUv3 (8 core)).

For the first experiment, one could use the same public code for all three hardware types, which requires using a very small batch size of 16 in order to avoid out-of-memory errors from the CPU and GPU. Under these conditions, it can be observed that TPUs are responsible for an approximate 100 times speedup as compared to CPUs and an approximate 3.5 times speedup as compared to GPUs when training an Xception [Cho17] model. Since TPUs operate more efficiently with large batch sizes, increasing the batch size to 128 is also an option, resulting in an additional 2 times speedup for TPUs and expectedly out-of-memory errors for GPUs and CPUs. Under these conditions, it can be said the TPU is able to train an Xception model more than 7 times faster than the GPU. It must be noted, however, that if the correct compiler is actually present, CPU, GPU and TPU can achieve the same task or result, although by following a different path and thus showing drastically different performance, naturally.

There are, however, a few more interesting findings to be mentioned. First of all, the observed speedups for model training vary according to the type of model, with Xception and Vgg16 [SZ15] performing better than ResNet50 [He+16a]. Additionally, model training is the only type of task

47 Mooney, P. (2020, February 20). When to use CPUs vs GPUs Vs TPUs in a kaggle competition? Retrieved February 23, 2021, from https://towardsdatascience.com/when-to-use-cpus-vs-gpus-vs-tpus-in-a-kaggle-competition-9af708a8c3eb

48 www.kaggle.com

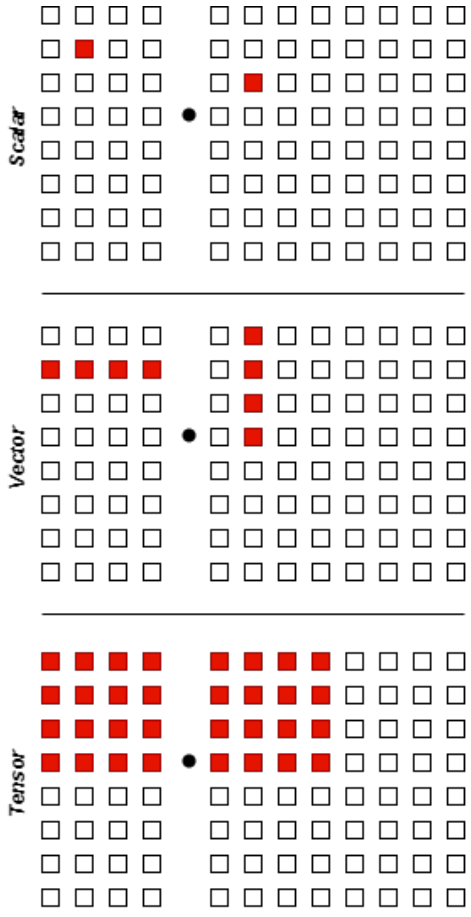

Figure 4.18: Graphical illustration of the compute primitives of CPU, GPU and TPU, from top to bottom, respectively.

where it could be observed that TPU outperforms GPU by such a large margin. For instance, it can be observed that the TPUs are 3 times faster than CPUs and 3 times slower than GPUs for performing a small number of predictions. In exchange, however, TPUs perform exceptionally when making predictions in some situations such as when making predictions on very large batches. However, a typical real-time workload for inference is

to process one image at a time, which cannot fully utilize the TPU power. In short, TPUs are great if one needs to process a lot of images at once.

Another aspect is the cost of using cutting-edge processors as there is a sizeable difference in pricing. TPUs are approximately 5 times more expensive than GPUs. Currently, it costs $1.46 per hour for an Nvidia Tesla P100 GPU, $8.00 per hour for a Google TPU v3 and $4.50 per hour for the TPUv2 with "on-demand" access on Google. If one tries to optimize for cost then it makes sense to use a TPU since it will train the model at least 5 times as fast as if it trained the same model using a GPU. There is also the possibility of using processors for free on Google Colab environment, although the allowed training time is quite limited, causing abrupt interruptions after the 8th hour in a day per account. A common practice therefore is to store the data in a sharded format (i.e. tf records) in a Google Cloud Service [49] bucket (i.e. name of the storage unit) then to pass it to the TPU in large batch sizes using the Tensorflow API (i.e. *tf.data*). This practice introduces speedups up to 5 times compared to the regular procedures.

Last but not least, it must be mentioned that Wang et. al have developed a rigorous benchmark called ParaDnn [WWB19] that can be used to compare the performance of different hardware types for training machine learning models. The authors were able to conclude that the performance benefit for parameterized models ranged from 1 time to 10 times, and the performance benefit for real models ranged from 3 times to 6.8 times when a TPU was used instead of a GPU. It was concluded that the TPUs perform best when combined with sharded datasets (*tf records*), large batch sizes (e.g., 64), and large models (e.g., EfficientNets [TL19]).

In conclusion, for model training and exploration, using TPU with the aforementioned practice should be preferred. It is the fastest way possible to train a large network and to evaluate its performance on the test data. This makes TPU-powered systems suitable for applications where new data is acquired often, making training regular.

49 https://cloud.google.com/

5 Approach

This section describes the proposed pipeline for the visual intelligence scheme developed to address previously mentioned objectives in Chapter 3. First, the object of interest is introduced and necessary datasets and taxonomy are shared. Next, the camera setup is introduced, along with the calibration procedures required to enable pixel projection between the cameras themselves, as well as those cameras and the end-effector. Then methods, frameworks, intermediate procedures required for the pipeline are explained in detail. Finally, the complete pipeline is elaborated, focusing on the inter-component communications.

5.1 Object of Interest

As this thesis proposes a visually intelligent scheme, it prefers to choose the object of interest as 3.5 inch computer hard drives to form a proof of concept. At least one computer hard drive is found in every EOL computer, making hard drives a common item among the e-waste category. On top of that, the magnet of the hard drive is made of an element called Neodymium, which is quite interesting for the recycling plants. Additionally, hard drives are light and reasonably sized, making them suitable objects for manipulation tasks.

There are several manufacturers of these devices. More than 100 computer hard drives including various brands (e.g., Hitachi, IBM, Maxtor, Seagate, WD, Samsung, Toshiba, etc.) were collected throughout the investigation and development. Figure 5.1 illustrates a sample hard drive without its lid. A standard hard drive contains around 12 parts that matter to the disassembly process. The reason why hard drives without lids are considered is due to the fact that the disassembly routines are bound to remove the lid eventually to reach the parts underneath. Therefore, it is crucial that the hard drive is viewed top-down with an appropriate angle

so that the parts and gaps between them are visible enough for the vision routines to make the necessary inferences.

Figure 5.1: Sample hard drive without its lid.

5.1.1 Taxanomy & Datasets

It is important to form a taxonomy for the object of interest, as this taxonomy could be used as a standard ground truth for any vision block to be developed, and later evaluated. Additionally, for the entities such as screws, wires and parts, relevant datasets have to be created to be used in possible machine learning blocks to address the objectives given in Chapter 3. Note that the datasets mentioned in this subsection are the raw, unprocessed datasets with no data augmentation applied on them. The numbers shared are the numbers of raw image samples to be collected.

A taxonomy including the illustrations, names and little descriptions of

hard drive components are given on the IMAGINE Website [50]. Another purpose of this effort was to build a list of common wordings about these different parts, in order to avoid uncertainties and irritations during the communication within the course of the IMAGINE project. It must be underlined that the shared taxonomy was agreed within the IMAGINE Consortium, including the industrial partner Electrocycling GmbH, providing hardware samples and images. While this taxonomy does not claim to be complete, it serves the interests of the project, and it stands as a sufficient structure to carry out the necessary evaluation strategies.

5.1.2 Datasets for Entities of Interest

This thesis considers an EOL device to have four types of entities in and/or on it. These entities are *screws*, *components*, *wires* and *gaps*. This subsection elaborates the need of collecting specific dataset for each entity, and explains the way these datasets were created. Note that the entity wires may or may not be found in every hard drive, however, it is still considered an entity.

Datasets are the core necessities of any machine learning block, as supervised or unsupervised machine learning algorithms heavily depend on the data available. This fact aside, even without machine learning paradigm involved, it is required to keep ground truth data for the future evaluation of any employed algorithm.

Screws

Screws are commonly found entities in any EOL device. They physically hold parts together through their threads fitting the hubs available. This makes screws primary entities to be removed before any part removal could take place. An unsuccessfully handled screw may cause the entire disassembly sequence to be cancelled, and the EOL device to be completely abandoned by the disassembly system or the human operator. Screws must be correctly detected, classified and physically interacted with.

Throughout the investigations, it was observed that most of the EOL devices considered have a certain set of screws. Agreed with the industrial

50 Taxonomy of HDD-Parts and -Components. (n.d.). Retrieved February 23, 2021, from https://imagine-h2020.eu/hdd-taxonomy.php

partner of IMAGINE, 12 types of screws are considered: Torx 6, 7, 8, 9, Allen 2.5, 2.75, 4, Slotted 4, 6.5, 10 and Phillips 1, 2. Note that the length of the thread is irrelevant for the perception block, as they are always occluded. Therefore, any vision routine only considers the head part of the screws. Figure 5.2 illustrates samples from every type and size considered. In order to address detection and classification purposes, over 20000 positive images of screw heads are found to be sufficient.

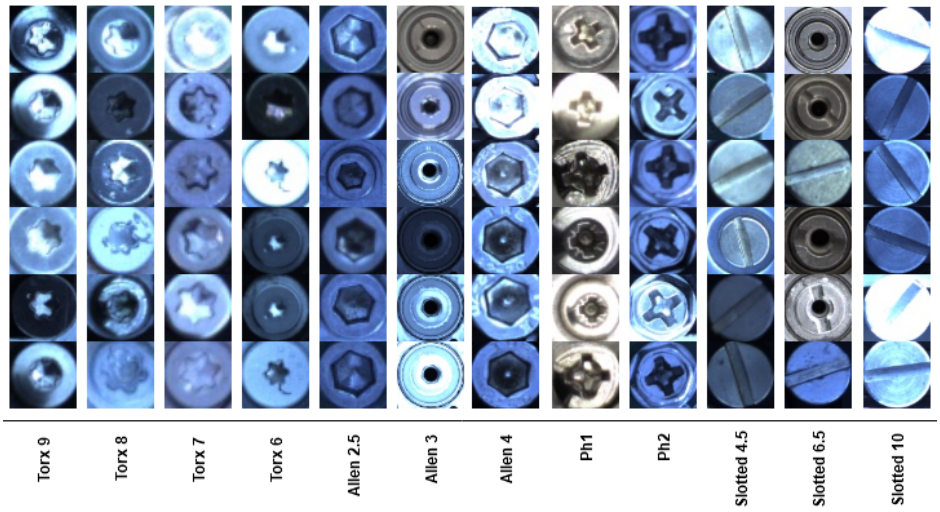

Figure 5.2: Various screw types encountered during the disassembly of EOL devices.

Collecting positive samples of screw heads alone, however, does not suffice. One also has to collect negative samples since the employed machine learning algorithm has to learn what does *not* look like a screw, despite being very similar in shape and feature. To this end, 10000 artefacts were found to be sufficient. Artefacts in this regard may be of any circular looking feature on the EOL device, such as transistors, stickers, empty screw hubs, and etc. Figure 5.3 shows samples of screws and artefacts.

Note that all of of the illustrated images were collected by using the setup shown in Figure 5.7, however, in order to account for great variance in the dataset, the images were collected under slightly different light conditions. They were collected by a tool that is introduced in the section 5.4.

Figure 5.3: Screw (bottom) and artefact (top) samples are the positive and negative samples, respectively.

Components

The taxonomy previously mentioned has been used with a slight difference while collecting images for the component dataset. According to the taxonomy, there are 12 different components. However, one of the components (motor contacts) does not need visual recognition, as it is noticed to come off with either the bay or the PCB (Printed Circuit Boards) attached. Therefore, we decide to collect images for only 11 components. Over 600 images of different brands and models were collected under different light conditions. These images include various stages of disassembly with parts in arbitrary positions (e.g., after a failed action). 5% of all images correspond to damaged devices to account realistically for the state of products in recycling plants. Figure 5.4 illustrates a damaged hard drive from the dataset. Note that the magnet is displaced, possibly due to failure during the manipulation attempt.

Top-down views of the computer hard drives are annotated into 11 classes according to the taxonomy, using the VIA annotation tool by Andrew Zisserman and Abhishek Dutta [DZ19]. Figure 5.5 shows a hard drive with annotated inner parts. The annotations are kept in a *.JSON* file, or a JavaScript Object Notation file[51]. This file format is commonly used for image annotation and loading of these annotations by machine learning framework routines.

51 A specific file format that uses human-readable text to store and transmit data objects formed by attribute–value pairs and array data types [ECM].

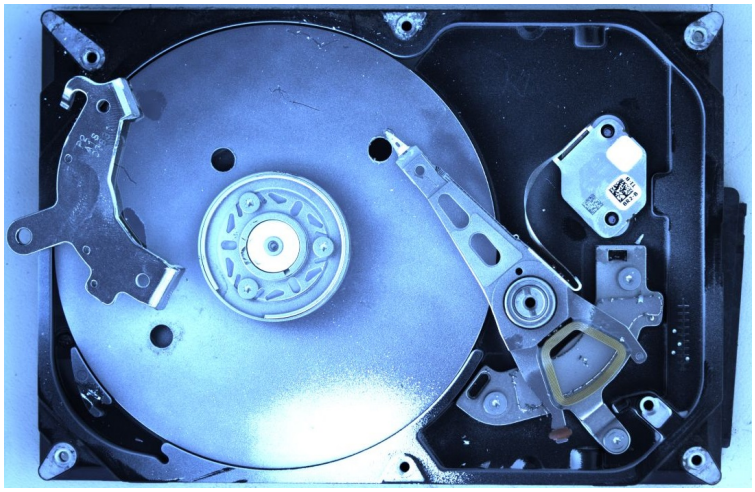

Figure 5.4: A hard drive with a damaged platter and a displaced magnet found in the dataset.

Wires

Wires, unlike screws, do not have specific shapes (e.g., circular), making them only suitable for pixel-wise segmentation schemes. They could be found in segments due to occlusion in EOL devices, they also could be of any color. It is safe to assume that wires are the most varying entity in this domain. This makes it usually very difficult or even impossible to find a dedicated dataset for specific visual tasks to detect wires. Hence, one is forced to deal with limited number of annotated training data. To this end, approximately 100 images of wires were collected manually. The strategy was to use any type of wires (including connectors) and manually create occlusions with arbitrary EOL components. As backgrounds, mostly PCBs were used, as wires are mostly found on PCBs in EOL devices. Figure 5.6 shows a few samples from the wire dataset acquired. Wires were later annotated with the VIA tool [DZ19].

Note that 100 images of wires are only the raw images. As it is explained in further sections, the images are subjected to heavy augmentation process before being used as training data.

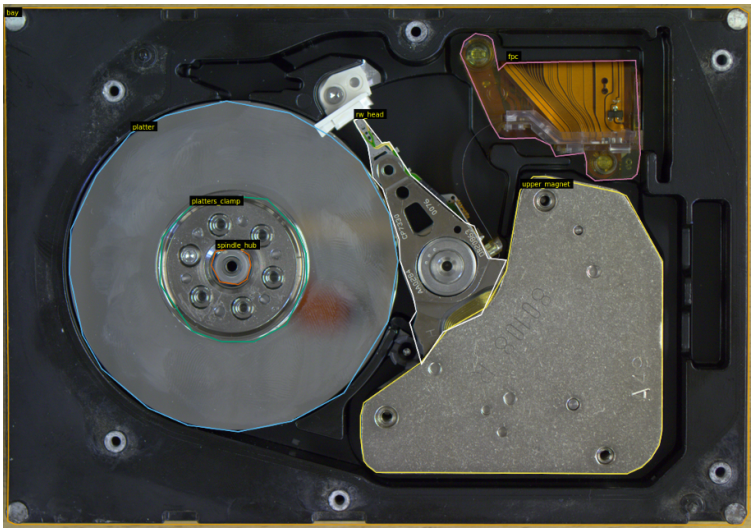

Figure 5.5: Ground truth annotations for a sample hard drive based on the taxonomy.

Gaps

Gaps are unique structures as they are the only 3D entity among all. Although this work does not employ any machine learning algorithm to detect gaps, still, ground truth data must be obtained to evaluate the proposed methods in Chapter 6. To this end, a free, online annotation tool called Semantic Segmentation Editor[52] from is used, allowing point clouds (3D data) to be annotated. These ground truth annotations consist of pointwise segmentations of each gap in a device. The human-annotated data points are then compared to the gap detection's findings to calculate its accuracy.

52 Hitachi-Automotive-And-Industry-Lab. (n.d.). Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor. Retrieved February 23, 2021, from https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

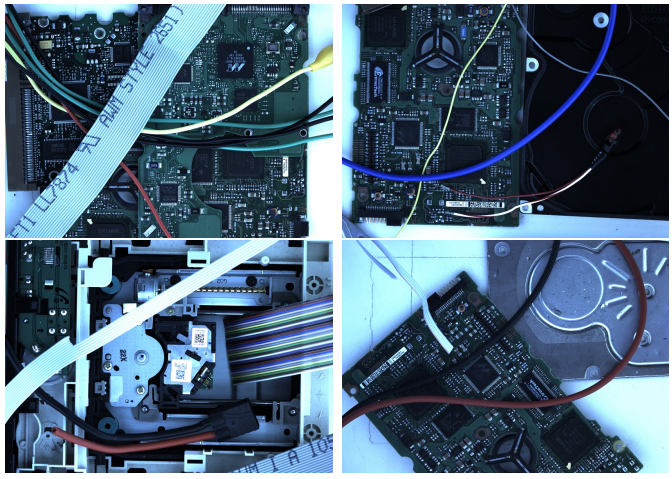

Figure 5.6: Sample raw images found in the wire dataset.

5.2 The Setup

Our proposed visual intelligence scheme needs to process two kinds of inputs. One of them is a high resolution RGB image, since only such high resolution image could provide the subtle details of the cues we are looking for. These cues could be edges, boundaries of parts, curves of circular structures, and color information. A possible intelligent scheme to seek these features has to do it with high quality images. Therefore, for the RGB input, resolution is top-priority. The second type of input is a top-down point cloud, where depth information could be perceived and processed to address the objective regarding the gaps. Therefore, our setup has to operate with sensors (cameras) that provide a high-resolution RGB image and a point cloud with fairly acceptable resolution. The required camera(s) could be chosen from any brand, however, this thesis is going to mention two specific cameras that are used in the proposed setup. Note that the proposed setup does not necessarily require these cameras by definition, it only requires sensors providing aforementioned inputs. Any camera system that provides them could be used within the proposed scheme.

We then propose a setup that uses a tilting table holding the device either horizontal (its surface normal aligned with the RGB camera) or 45°

(aligned with the stereo camera) on request (see Figure 5.7). We use a Basler acA4600-7gc monocular camera which provides images with 4608 × 3288 resolution and 3.5 FPS and a Nerian Karmin2 stereo camera with a depth error of 0.06cm from the minimum range of 35cm. The tilting table is used to direct the device surface to either of the cameras with a precise angle.

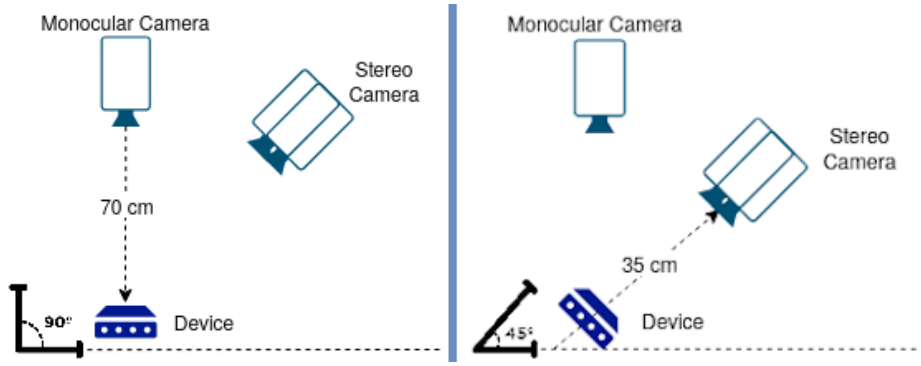

Figure 5.7: Setup used for the proposed scheme: a monocular RGB camera is oriented downwards ($\theta_{Monocular} = 90°$) and a stereo monochrome camera is mounted at $\theta_{Stereo} = 45°$. This configuration is chosen due to the different focal lengths of the cameras. Both cameras are pointing at the device to be analysed. A tilting mechanism allows the system to obtain "top-down" views of the devices with both cameras.

The stereo system consist of a Karmin2 3D stereo camera[53], connected to the SceneScan FPGA. According to their publication [Sch18], SceneScan provides real-time stereo vision processing up to 100 frames per second using a modified version of the *SGM* algorithm [Hir05]. The camera has a resolution of 800 × 600 and a 10 cm stereo baseline distance. Furthermore, the camera was placed in a 90 degree angle above the device with a 35 cm distance to the device. More information about the stereo cameras is shared in Figure 7.11 of Appendix 7.1, where the camera specifications are listed, as well as in Figure 7.12 which lists the lens and sensor metrics.

Similarly, the monocular camera has the specifications shown in Figure 7.13 of Appendix 7.1. According to the official documentation on the

53 GmbH, Technologies, N. (n.d.). Karmin2 - Nerian's 3d stereo camera. Retrieved February 23, 2021, from https://nerian.com/products/karmin2-3d-stereo-camera/

manufacturer website[54], Basler acA4600-7gc GigE camera with the ON Semiconductor MT9F002 CMOS sensor delivers 7 frames per second at 14 MP resolution.

The final disassembly scene therefore refers to the scene perceived by the cameras found in our setup. Basically, the scene has to contain the object of interest without any irrelevant object or artefact nearby as shown in Figure 5.1. Preferably, the scene has to be illuminated in a way that there are no shadows or major reflections caused, as these are known to hinder the vision algorithms used.

5.2.1 Pixel Projection

Having calibrated both of the cameras, the following transformations are made available to the visual intelligence scheme: *Monocular-To-World*, *World-To-Monocular*, *Stereo-To-World*, *World-To-Stereo*: , *Stereo-To-Monocular*, *Monocular-To-Stereo*. These transformations are used to transform any point in the host frame to the target frame. Note that the transformations have been made *ROS Services*, and the ROS package *TF* has been used to conduct the mathematical transformations.

These transformations are important to be able to project any finding on the RGB image acquired by the monocular camera, on the point cloud frame acquired by the stereo camera, and vice versa. Moreover, the pixel projections are definitely required for manipulation purposes. Eventually the frame where any cue is found has to be transformed into the frame where end-effector has to operate. This is made possible by having both cameras as well as the robot end-effector to recognize a common frame of reference (as known as *World*), and use transformations between the *World* frame and their own frames such as *Monocular* or *Stereo* for the vision block, and possibly *End-Effector* for the manipulator.

A practical way to test the pixel projection as well as the calibration procedures is to measure a known point using a physical ruler, and note its coordinates with respect to the *World* frame. Note that this point has to be visible from both of the cameras, since the next step is to find out the image coordinates of that point in both camera images. Marking the point with

54 Basler AG. (2021, February 22). Basler ace Aca4600-7gc - Area Scan Camera. Retrieved February 23, 2021, from https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca4600-7gc/

a colored pen helps identifying the pixels in the images from the cameras. After having the *World* coordinates and their corresponding *Monocular* and *Stereo* coordinates, a transformation such as *World-To-Monocular* or *Stereo-To-Monocular* reveals the correctness of the projection. It is not possible to have 100% correctness since there is always a slight human error while measuring or calibrating, however, the transformed points should not be too far from the physically measured values.

5.3 Proposed Pipeline

The proposed pipeline presented in this thesis is composed of 8 vision components (or blocks), each aiming to address one or more objectives of the State Estimation package, as mentioned in Chapter 3. A graphical illustration of the entire pipeline can be seen in Figure 7.17 of Appendix 7.17, whereas the main vision blocks in the pip line are shown in Figure 5.8. Fundamentally, the pipeline starts acquiring RGB and depth data, and conducts its inferences to find every entity present in the disassembly scene. These entities are components, screws, wires and gaps, as mentioned previously. The main idea is to derive predicates by finding the geometric relations between the entities in the EOL device.

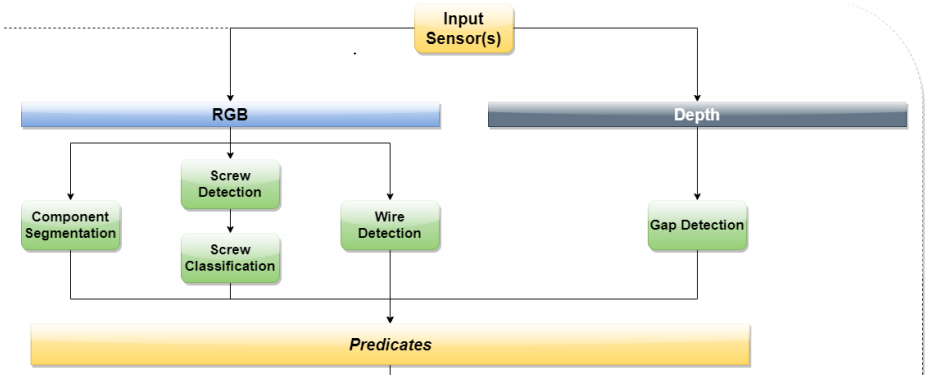

Figure 5.8: Main vision blocks in the pipeline are shown in green, where each block aims to generate predicates regarding the found entities in the scene.

The State Estimation package is responsible for providing the required visual data and the interpretation of it as predicates to two other packages

that are the association engine and the planner, as explained in Chapter 3, illustrated in Figure 3.3. It must be underlined at this point that the provided visual data and computed predicates utilise the calibration information allowing pixel projection and thus common frame of reference for the other packages, as shown in Figure 5.9.

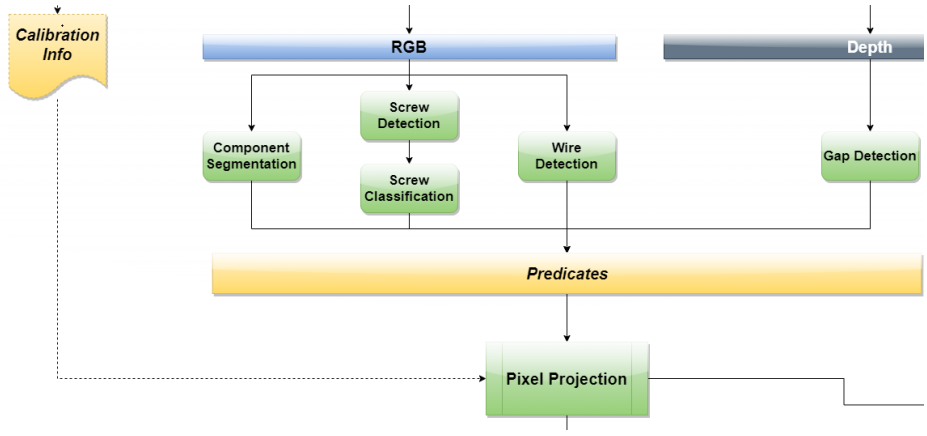

Figure 5.9: An offline calibration procedure allows pixel projection and common frame of reference for the predicates.

Last but not the least, the proposed pipeline employs a *bookkeeping* mechanism as illustrated in Figure 5.10, which is explained in Section 5.9. Basically, the visual intelligence scheme accounts for scene awareness, enabling the system to register the visual changes between consecutive frames. This way, the planner is informed about an abnormal change (e.g., an accidentally dropped part, unintended move of a part) or an unchanged scene (e.g., failed action).

After bookkeeping the consequent scene changes, providing them along with the necessary visual data and predicates to further packages, the State Estimation waits for a ROS signal for another analysis.

5.4 Screw Detection

As mentioned previously, screws are the most commonly encountered entities in any EOL device, making them crucial to the course of disassembly.

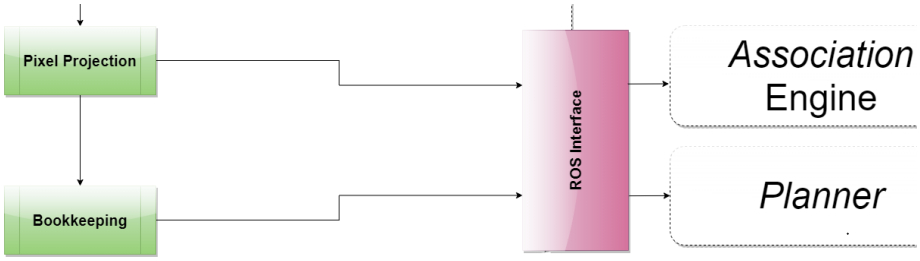

Figure 5.10: Predicates on a common frame are reported to the further packages through ROS interface, their bookkeeping is done as well to assist the planner.

In order to have a continuous disassembly, the screw detection strategy has to be robust enough to account for most of the screws found in or over the device.

Although automated disassembly has been investigated for a while now [Drö+14; Weg+15; Wei+06] and there are some schemes [BRP16; Bük+01; Els+12; Pom+04a; Xie+08] for automating certain processes, none of the proposed schemes is actually offering a generalising, extensible, and universal solution of the screw detection problem [YW19]. Numerous algorithms have been proposed for the task as a part of automated disassembly strategies. "Most of these were either too much model-dependent; meaning screw-specific (i.e. only Torx6) or device-specific (i.e. only electric motor screws) [BRP16; Cru+08] or they were extremely frail due to the fact that the methods they used were quite dependent on classical computer vision methods, which are easily affected by a slight change of illumination [Uki07]" (Yildiz, 2019, p.2). Also, some of the methods [BRP16] require a depth sensor (i.e. RGB-D camera) to conduct the detection.

Another attempt was reported by Ukida et al. 2007 [Uki07]. "The authors tried to perform screw detection using template matching on metal ceiling structures for dismantling and successful reuse of light steel gauges. In this scheme, a hierarchical vision system detects the light steel gauge first and then uses multiple template matching to detect screws. This method also has a very obvious shortcoming: the method depends on a fixed template and therefore it cannot generalize. Also, back then, a light steel gauge had only one type of screw, but there is no guarantee that it will stay like this in the future. Changing the template is tedious and

non-desirable and, thus, this method is highly specific and it also cannot address other metal structures or E-Waste devices" (Yildiz, 2019, p.2).

One study focuses on the disassembly of the electric vehicle batteries using a robot system [Weg+15]. "Their main goal was to detect M5 bolts on the battery joints. They used a Haar-type cascade classifier, which is trained on cropped images of M5 bolts. Then, to improve the performance of classification, false positives detected from the classifier were added to the negative set. Although the approach sounds quite feasible, unfortunately Haar cascades are not performing very good when it comes to classification. They were able to achieve only 50% detection accuracy, which makes the method impractical for industrial use" (Yildiz, 2019, p.2).

Another study worthy of mentioning focused on autonomous disassembly of electric vehicle motors [BRP16]. "The authors tried to detect screws found on electric vehicle motors using an RGB-D sensor (Kinect) [Zha12]. Although the proposed algorithm is scale, rotation, and translation invariant, it heavily relies on traditional computer vision methods such as Harris corner detection and HSV image analysis, which are easily affected by the lighting conditions. Another shortcoming is the fact that they require a depth image from the RGB-D sensor to remove false positives such as holes, which adds computational load" (Yildiz, 2019, p.2)

Therefore, it appears that there is a lack in generalizing, device and screw-independent methods, which can be used in disassembly processes [YW19].

The proposed screw detection scheme is derived from the recent work Yildiz et al. [YW19], which has two modes: *offline* and *online*. In the offline mode, the aim is to collect positive and negative samples for the training of the deep neural networks that we use. Therefore, in the offline mode, the scheme saves possible candidates, which could be screws or artifacts cropped from the camera image. "These images are then to be divided by a human into positive and negative samples (screw/non-screw) for the training session" (Yildiz, 2019, p.2). Figure 5.11 illustrates the offline mode on the right side.

After collecting the training data and training the network, the second mode of the scheme is ready to use. When it comes to inferring of screw locations, as shown in Figure 5.11, once more the same initial function blocks are performed, however this time the system uses the trained model

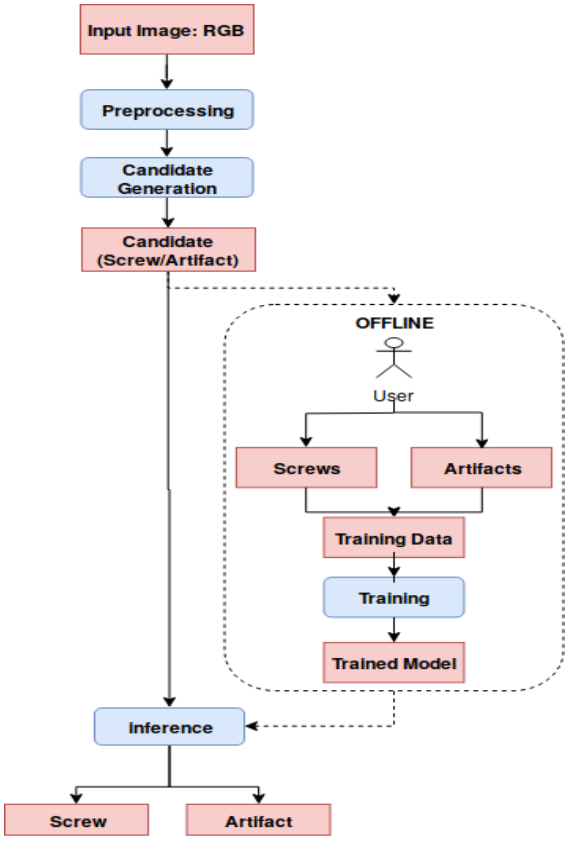

Figure 5.11: The proposed scheme is composed of offline and online stages, which are used for training the network and then for inferring screw locations using the trained model [YW19].

of the network to differentiate between positive and negative candidates [YW19]. "The scheme then marks and returns the locations of the screws seen in the image" (Yildiz, 2019, p.3).

5.4.1 Candidate Generation

As stated above, Hough circle finder is first used to crop the image to only the region where the device is visible. Cropping is done in a parameter dependent way [YW19]. Depending on the device, users can crop the

incoming image as required. Following this, the RGB image is converted into a Grayscale image. This is the preprocessing block seen in Figure 5.11.

"Different types of screws found in the domain of E-Waste have been analyzed to make sure that the method covers all conventionally used screws found in this domain. For this purpose, various electronic devices were investigated, which can be found in myriad numbers nowadays in E-Waste, such as computer hard drives, DVD players, gaming consoles and many more. As expected, almost all screws in the domain were found circular, which is the natural geometry of a screw and represents the central feature to be used to detect a screw object. There are also non-circular screws manufactured, however, those are few and no such screws were encountered in the devices mentioned above. Therefore the method is based on first finding circular structures in the images. Obviously, not every circular structure is a screw, for example stickers, holes, transistors, etc. exist, which are also circular, but not screws. Still, circular structures provide us with priors for screws and the first step of our method is to collect those screw candidates" (Yildiz, 2019, p.3).

As previously mentioned, in order to collect candidates, one runs the program in the offline mode which utilises *Hough Transform* for candidate detection [YW19]. This is a standard computer vision method for circle detection [DH71] and shall not be explained here. Different from the standard Hough Transform, here a version is used that relies on the so-called *Hough Gradient* (of the OpenCV library [BK08]). That is, using the gradient information of the edges that form the circle. For more information on Hough transform, the reader is referred to the handbook published by the creators of the aforementioned library for further implementation details on the algorithm of the Hough Gradient.

5.4.2 Training the Classifiers

As mentioned before, the user manually separates screw types and artifacts by which a classifier can be trained using these positive and negative examples as training data [YW19]. In Figure 5.2 types of screw heads and artifacts taken from various devices found in E-Waste are shown. In general, these screws are found in other device-classes and, thus, the resulting training set can be transferred also to other devices. In that case,

however, one has to increase the number of samples to account for more types of screws.

There are several classifiers in the literature, which one could investigate on the task, such as Xception [Cho17], InceptionV3 [Sze+16b], ResneXt101 [Xie+17], InceptionResnetV2 [Sze+17], Densenet201 [Hua+17], Resnet101v2 [He+16b] models. "These networks achieve over 93% top-5 accuracy on the well-known Imagenet dataset [Den+09]. In order to further improve learning and to reduce overfitting, a dropout layer is inserted before the last fully connected layer of each network. To further reduce overfitting and to come up with a model that can generalize, an additional data augmentation step is applied. There are several data augmentation operations applied to introduce more variety in the data. However, the most important ones are normalizing the image data into a range of [0,1], rotation [0,360], randomly setting the brightness in the range [0.5,1.5]" (Yildiz, 2019, p.3). The experimental evaluation each of these networks on the test data allowed authors to select two to be used in the processing pipeline [YW19].

Evaluating the Models

Several experiments were conducted on the test data collected. Out of the top six state-of-the-art classifiers, the best performing two ones were chosen and combined as ensembles of models, as illustrated in Figure 5.12. Ensembles of models refers to the practice of combining predictions from multiple statistical models to form one final prediction [AWZ18]. This creates opportunity for diversity in the representational capacity of the model. The concept is analogous to anecdotes such as seeking the opinion of multiple professionals, and it has been used in many recent works [ABD16; LZP16; Nob+20; PAV19; ŞA20]. Below the details of the experimental evaluation are provided, along with the presented justification for the decision of using an ensemble.

For the evaluation of the classifiers, the collected dataset mentioned in Section 5.1 is used, consisting of over 20000 samples and split into training, validation and test sets in ratios of 70%, 20%, 10%, respectively. "A computer with Intel Core i7-4770 CPU @ 3.40GHz, 16GB of RAM with GeForce GTX Titan X graphic card was used to train the classifiers. For evaluation of the performance of the screw detector, approximately

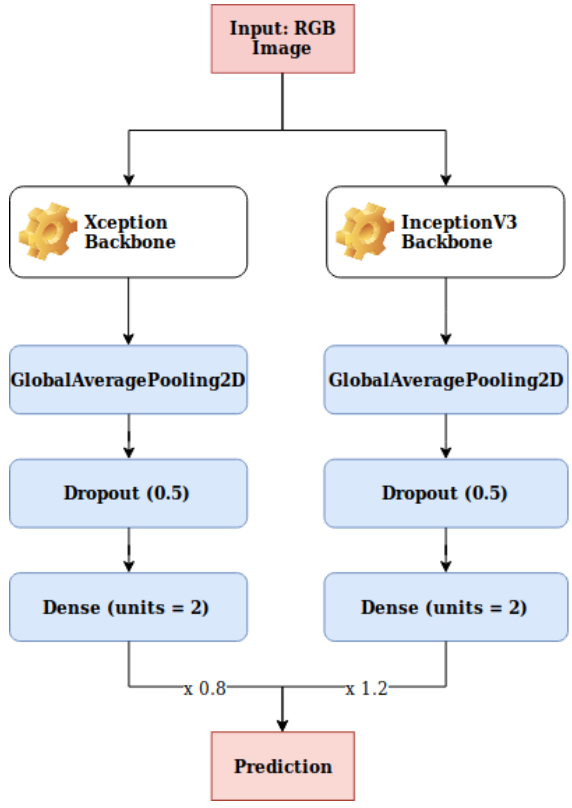

Figure 5.12: The proposed ensembled model, which is a combination of the two best performing networks given our data and classification task [YW19].

300 hard drive images containing over 1500 screws were collected. We split those images into training and test sets with a ratio of 2:1. YOLOv3 [RF18] (a state-of-the-art object detector) was chosen to be re-trained on the training set to demonstrate the efficiency of the screw detection pipeline. YOLOv3 was then compared with the proposed ensemble" (Yildiz, 2019, p.4).

Standard metrics [Eve+10] for classification evaluation were used. The significant bit here is the accuracy of the networks picked for the pipeline. The accuracy of each of them is calculated as follows:

$$Acc = TP + TNTP + TN + FP + FN$$

where TP stands for True Positives, TN for True Negatives, FP for False Positives and FN for False Negatives. To evaluate screw detection performance, Average Precision (AP) was used, which is calculated based on the precision-recall curve. Summary of the experimental results with regards to performance of each classifier against the testing set is given in Table 5.1.

Table 5.1: Experimental results for the chosen state-of-the-art models [YW19].

Model	TP	TN	FP	FN	Acc
Xception	975	3244	25	41	98.5
InceptionV3	973	3262	27	23	98.8
ResneXt101	962	3260	38	25	98.5
InceptionResnetV2	965	3260	35	25	98.6
DenseNet201	897	3272	103	13	97.3
ResNetV2	943	3208	57	77	96.9
Integrated model	**988**	**3254**	**12**	**31**	**99.0**

"From the collected results in Table 5.1, one can conclude the following: All of the investigated models achieve very high accuracy - over 96% on the testing data, with the model InceptionV3 scoring the highest accuracy of 98.8% among single models. One also notices that the InceptionResnetV2 model scores a very high accuracy of 98.6%. However, it should be noted that this model is much heavier than the Xception model, which scores comparable accuracy of 98.5%. It was then decided to combine two best performing models to boost the accuracy of the classifier. At this point the models InceptionV3 and Xception were chosen to build an ensemble of models for final prediction. Since InceptionV3 performs slightly better than Xception, using slightly higher weights on the results of the InceptionV3 model is preferred. The final confidence score used to evaluate the ensemble of models is presented below:

$$score = 1.2 \times \text{InceptionV3} + 0.8 \times \text{Xception}$$

Since the scheme uses an ensemble, one has to adjust the value of the

confidence level carefully. Experiments were conducted to find out the best threshold value for the confidence level, and the threshold for the model was chosen by sliding the threshold value over the range from 0.5 to 1.5 with steps of 0.01. Experiments showed that 0.8 is the best threshold and thus is chosen for the ensemble of models" (Yildiz, 2019, p.4)

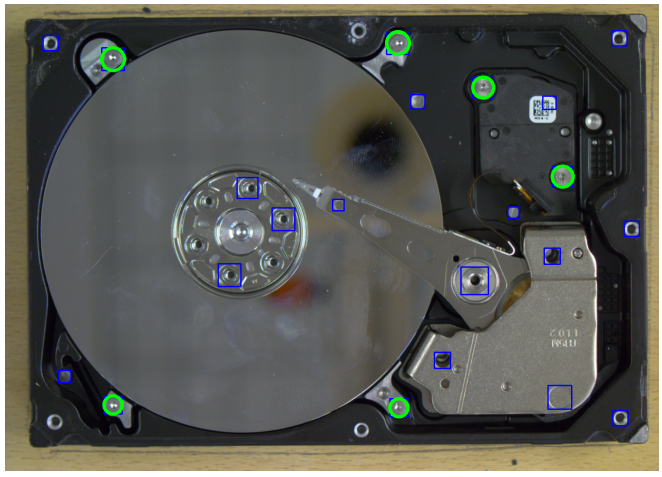

Figure 5.13: Sample detection output of the screw detection pipeline. Blue squares represent the candidates generated by Hough circle finder, whereas green circles represent the classified screws [YW19].

Afterward, the ensemble model is deployed in a screw detection pipeline and evaluated on the test dataset. Figure 5.14 shows the precision-recall curve for the screw detector, whereas Figure 5.13 illustrates a sample detection output. "The pipeline achieves an AP of 80.23, which clearly outperforms the well known detector YOLOv3 with an AP of 66.47" (Yildiz, 2019, p.4). Figure 5.15 illustrates some samples of the detection.

As it could be noticed, despite the fact that the ensemble model achieves accuracy of 99.00, the detector's AP remains at 80.23. The reason for this is the Hough circle finder that generates the screw candidates to begin with. Therefore, the trained classifier is limited with the Hough circle finder's accuracy, being the only limitation of the proposed screw detection pipeline. There are possible solutions to this, such as employing another DCNN to find the circular structures in the image. Although there is a

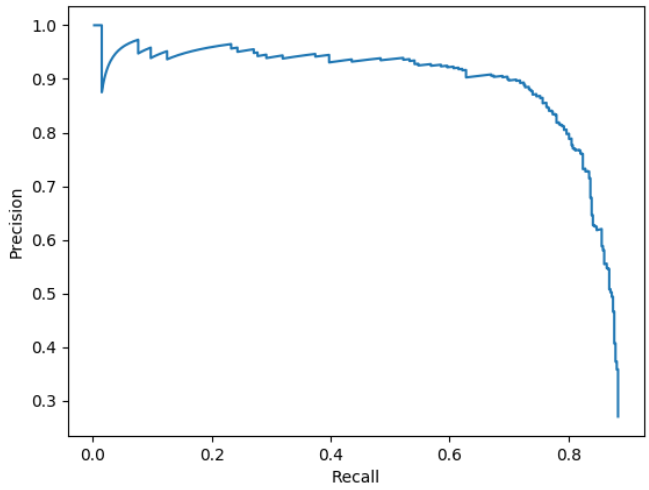

Figure 5.14: Precision-Recall curve for screw detection [YW19].

possibility of such a DCNN working better than Hough circle finder, it comes with the necessity of having a new dataset altogether, consisting of samples from as many EOL devices as there could be. A requirement such as this, however, makes a possible use of DCNN in this case slightly impractical.

In conclusion, the proposed screw detection pipeline tackles the fundamental problem of screw detection in disassembly environments [YW19]. "The problem itself is a challenging one, since screws have variable shapes and appearance, and not every electronic device has the same type of screws. This is the reason why previously developed methods were not useful as a general solution to this problem. Hence the proposed model, which is based on the Hough transform and deep neural networks. The scheme easily lets the user operate the system for any device of his/her choice, as long as the user separates the collected data into screws and artifacts. After doing this and training the network, it is demonstrated that the system achieves real-time performance with quite high accuracy. This has been quantified with hard drive devices of different models and sizes, which have different sizes and types of screws, as documented by the experimental evaluation results of the scheme. The data set, as well as the

Figure 5.15: Sample outputs of the pipeline. Blue rectangles show the candidates generated by Hough Transform and the green circles refer to the true positives predicted by the ensemble model. The proposed scheme is robust to different backgrounds and to different illumination. In all of the images above, the screws were detected correctly [YW19].

ROS-based implementation, are published to facilitate further research [55]" (Yildiz, 2019, p.5).

55 Yildiz, E. (n.d.). Eyildiz-ugoe/screw_detection. Retrieved February 23, 2021, from https://github.com/eyildiz-ugoe/screw_detection

5.5 Screw Classification

Screws in or over the EOL devices can be of many types and sizes. Detecting the screws themselves alone is not sufficient to complete the disassembly, given that there is no priori information about the EOL device. Even if the screw detector finds all of the screws correctly, without the type and size information, it is quite a risky process to unscrew the detected screws. A Torx-9 screw, once tried with an Allen 2.75 key, may result with either damaging the Allen key, or the screw. In both cases the disassembly would terminate entirely for that EOL device, which has to be avoided.

To this end, screw detection is extended to a screw classification [YW20]. Similar to the findings of the literature search conducted for screw detection, screw classification yields no significant work either. The existing schemes do not generalize and pose as universal solutions for the problem of classification of screw heads. Basically, the problem of screw head classification is not even addressed yet by the community. Thus, there is still a substantial lack in generalizable, device- and environment-independent methods to detect and classify screw heads, which can be used in automated disassembly processes.

As it is stated that the classification effort is built on the existing screw detection [YW19], the pipeline has been inherited and added a classification block to it, as illustrated in Figure 5.16. Our previous work enables the user to collect training data by cropping circular candidates from the scene. The cropped circular candidates are then to be divided into their respective classes (artifact, Torx8, Ph2, Slotted6.5, Allen2.75, etc.) by a human.

Afterward, first the detector model can be trained to classify screws from artefacts (circular non-screws structures), as we explained in our previous work [YW19]. Having deployed a model that can differentiate screws from artefacts, it is possible to train and deploy the new classifier, which can classify 12 different types of screw heads, returning the type/size information and locations of the screws seen in the image. It must be underlined that the preprocessing step is directly taken from the previous work [YW19], where classical computer vision is first used to crop the image to only the region where the device is visible, and grayscale it.

E-Waste is a vast category of devices with different structures and materials. A possible screw head classifier should therefore account for

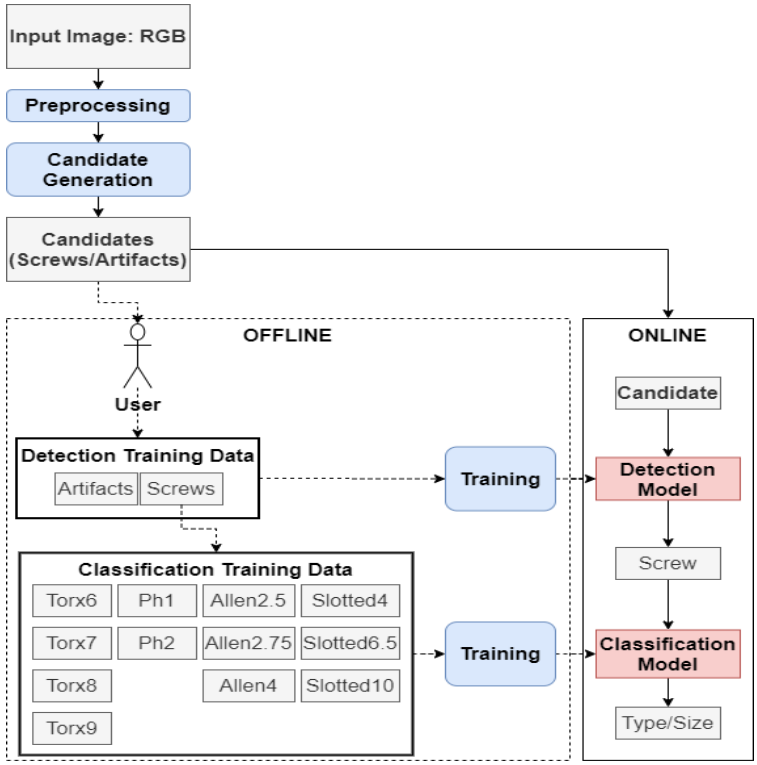

Figure 5.16: Screw classification pipeline extends the previously described screw detection pipeline, making it possible to classify 12 different screw types/sizes [YW20].

most of the screws encountered in the disassembly domain. In order to come up with a scheme that has reasonable levels of generalization ability, different types of screws found in the domain of E-Waste were analyzed. To make sure that the method covers the most encountered and conventionally used screw types found in this domain, experts from the disassembly plant were consulted in cooperation with our university and agreed on 12 types of screw heads, such as different sizes of Torx, Philips, Slotted and Allen heads, as described in Section 5.1. Figure 5.2 illustrates the commonly encountered screw types, suggested by the consulted recycling plant.

The base candidate generator from the previous work [YW19] is directly

used in order to collect candidates. It should be noted that after collecting the candidates, we then switch back to RGB from grayscale, since the classifiers operate far better in colored images than in grayscaled ones.

5.5.1 Training the Classifiers

The state-of-the-art classifiers found in the literature were investigated, and the three top-performing ones were picked for comparison. These networks were performing tolerably well, given a not-so-large dataset for a specific device-class (hard drives of any size). Finally, we decided to evaluate EfficientNets [TL19], ResNets [He+16a], DenseNets [Hua+17], scoring top accuracies on the ImageNet [Den+09]. Additionally, EfficientNets have been used in the latest works [Xie+19] in pursuit of improving ImageNet classification, by using a new self-traning method called Noisy Student Training. Inspired by this effort, we chose EfficientNetB2.

The strategy to evaluate the networks is described as follows. The standard procedure for transfer learning mostly implies cutting the pre-trained model on the last convolutional layer and adding a new sequence of linear layers, called the head. This head architecture is used for all models explored. In the first 10 epochs, only the added final layers of the model are trained by freezing all convolutional layers, not allowing any updates to their weights. Afterward, all layers are unfrozen and the entire network is trained [YW20]. It is useful to use differential learning rates at this stage, as it is not desired to change the early layers of the models as much as the later ones. Therefore, lower learning rates are used in the first layers and higher ones in the end (e.g., Adam optimizer [KB14] with the learning rate of 1×10^{-5}). Figure 5.17 illustrates the model architecture used. Note that the term "Block" here is a higher abstraction used for group of layers.

Similar to screw detection, to further reduce overfitting and to come up with a model that can generalize, an additional data augmentation step including operations such as rotation, brightness and contrast is utilized here as well.

5.5.2 Evaluating the Models

Several experiments were conducted on the test data. Out of the top three state-of-the-art classifiers, the best performing one, namely EfficientNetB2

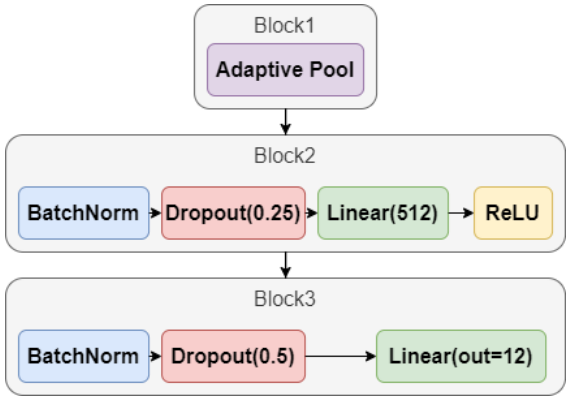

Figure 5.17: Head architecture of the model [YW20].

(one of the best performing models given Noisy Student weights) was used in the pipeline seen in Figure 5.16.

For the evaluation of the screw classifiers, as mentioned previously in Section 5.1, a dataset of over 20000 samples were collected and split into training, validation and test sets using the same ratios in screw detection module. Here, as well, a computer with Intel Core i7-4770 CPU @ 3.40GHz, 16GB of RAM with GeForce GTX Titan X graphics card was used to train the classifiers. For evaluation of the performance of the entire pipeline (detection and classification), an approximate number of 50 hard drive images were collected, containing over 500 screw-like elements as a test set the model has never seen before.

As the pipeline is composed of two main blocks, namely the Hough circle detector and our classifier, EfficientNetB2, it is required to assess the detection as well as the classification abilities of it. To this end, the following strategy was pursued: First, the test images were annotated, each having only one hard drive with top-down view. These images contain drives with or without screws, by which the Hough circle finder could be assessed. These scene images were annotated by marking screws with squares, which would form the ground truth for assessing the Hough circle finder's accuracy. The standard VOC evaluation [Eve+10] was preferred and it was found out that the Hough circle detector actually works with 0.783 mean IoU (Intersection over Union) with the optimal parameters

Table 5.2: Accuracy of the state-of-the-art models with huge variation of hyper-parameters. The top three performing ones are highlighted [YW20].

Model	Grayscale	Size	Loss	Acc.	Min. Acc.	F1	Transfer Learning
EfficientNetB2A	**No**	**256**	**0.1187**	**0.968**	**0.79**	**0.97**	**Noisy Student**
EfficientNetB2A	No	64	0.2144	0.936	0.78	0.93	ImageNet
EfficientNetB2A	No	128	0.1871	0.951	0.85	0.95	ImageNet
EfficientNetB2A	Yes	128	0.2199	0.948	0.67	0.94	ImageNet
EfficientNetB3A	Yes	64	0.2072	0.937	0.75	0.93	ImageNet
EfficientNetB3A	No	64	0.2051	0.939	0.74	0.94	ImageNet
DenseNet121	**No**	**128**	**0.1415**	**0.961**	**0.81**	**0.96**	**ImageNet**
DenseNet121	Yes	128	0.1489	0.957	0.74	0.95	ImageNet
DenseNet121	No	64	0.1896	0.937	0.72	0.93	ImageNet
DenseNet121	No	64	0.2306	0.934	0.71	0.93	ImageNet
DenseNet201	No	256	0.1170	0.966	0.79	0.96	ImageNet
ResNet34	**No**	**128**	**0.1538**	**0.955**	**0.80**	**0.95**	**ImageNet**
ResNet34	Yes	128	0.2026	0.951	0.69	0.95	ImageNet
ResNet50v2	No	256	0.1732	0.942	0.73	0.94	ImageNet

found for the IMAGINE setup. IoU here refers to what amount of screw region is correctly detected by the Hough circle finder. If the detected region for a screw is below 70%, it is bound to result in bad prediction for both detection and classification. It must be also noted that the pipeline is limited by the accuracy of Hough Transform and the screw detection previously introduced. Although the accuracy of Hough circle detection can vary depending on the parameters of the function such as min/max radius, min/max threshold, final accuracy of the pipeline is found 0.75, and calculated as follows:

$$Acc\_P = Acc\_CD * Acc\_SD * Acc\_SC$$

where Acc_P stands for the accuracy of pipeline, Acc_{CD} stands for the accuracy of the circle detector, Acc_{SD} stands for the accuracy of the screw detector, and Acc_{SC} stands for the accuracy of the screw classifier.

For the evaluation of the classifier, the standard metrics of loss, accuracy, min_accuracy and f1_score are considered. Summary of the experimental results with regards to performance of each classifier against the validation set can be seen in Table 5.2.

From the results presented in Table 5.2, one can conclude the following: All of the investigated models achieve very high accuracy - over 90% on the

testing data, with the model EfficientNetB2 scoring the highest minimum accuracy of 85% among single models. Additionally, it is emphasized that the augmentation strategy plays a pivotal role in the classifier accuracy. Especially for circular objects, rotation guarantees that the training data accounts for screws that are rotated for each angle [YW20]. Using the Albumentations [Bus+20] library applied a rotation of 360 degrees, horizontal and vertical flips, as well as brightness and contrast changes.

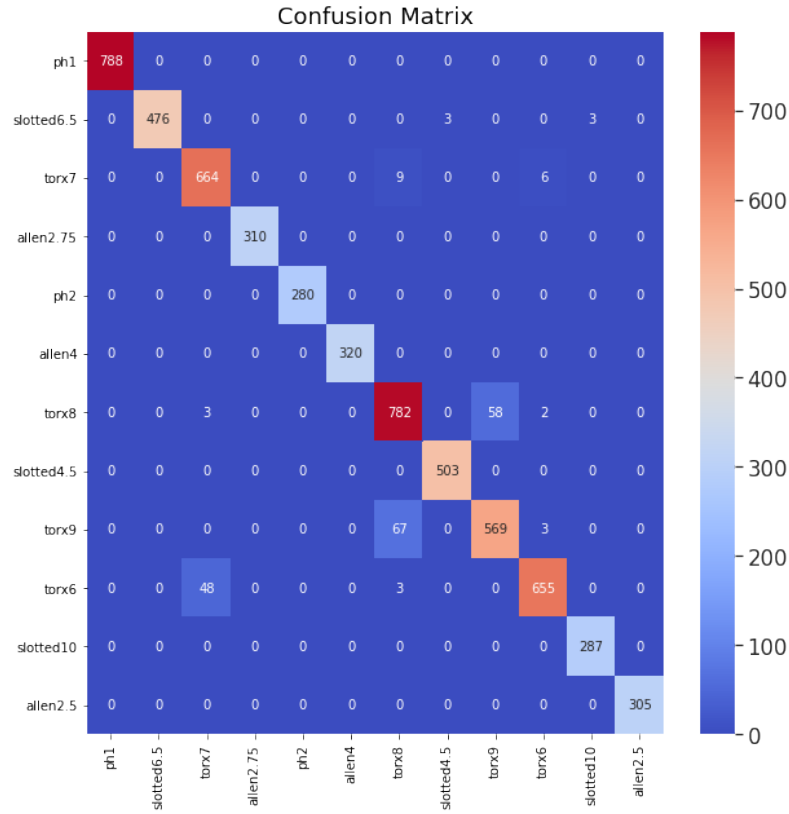

Figure 5.18: Confusion matrix acquired through the experiments [YW20].

The final model is then employed in the pipeline and evaluated on the test dataset. Figure 5.19 shows the precision-recall curve of the classifier, whereas Figure 5.18 illustrates the confusion matrix [YW20]. The screw

classification pipeline achieves an AP of 0.757. The reason why the classifier accuracy is higher than the AP is due to the Hough circle finder, as explained earlier. Hough circle finder manages to score approximately 75% of the time, limiting the pipeline's overall AP. Figure 5.20 illustrates the detection and classification of the screws found on a Hitachi Deskstar 3.5" hard drive during the disassembly [YW20].

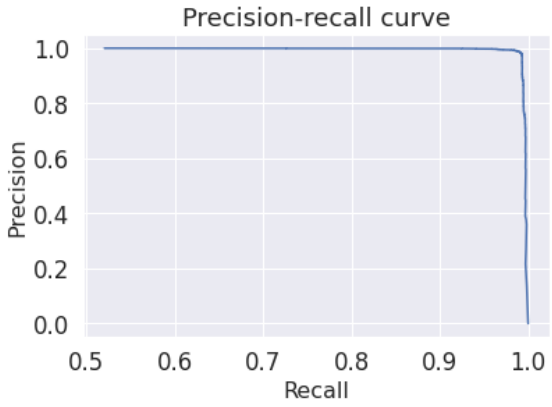

Figure 5.19: Precision-Recall curve for the final model for the classifier [YW20].

In conclusion, the screw classification proposal is the extension of the previous work [YW19], where the scheme lets users to collect as much as data they desire from a device. Given that the user creates the ground truth classification (i.e., separates the collected screw data into their respective type and size categories), the model is able to hit a very high F1 score of 97%, whereas with Hough it was found out that with optimal parameters, circle finder hits 78% IoU for screw regions. "The results were quantified with hard drive devices of different models and sizes, which have different sizes and types of screws as documented by the experimental evaluation results of the scheme. Additionally, screws from other devices were acquired and introduced into the scene to test the generalization ability of the pipeline, achieving promising results as well" (Yildiz, 2020, p.6).

Note that all the misclassifications apart from the intra-class ones (e.g., Torx6/Torx8) are empirically found to be a direct result of how Hough cuts regions [YW20]. In some cases the artefacts are found as Allen2.75, which, however, is a strongly valid classification and detection by the classifiers

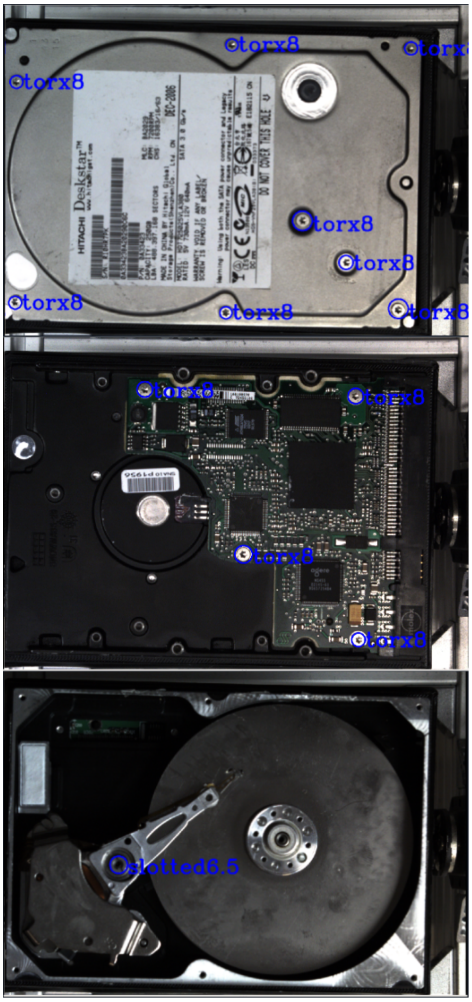

Figure 5.20: Classifications of detected screws during the disassembly of a hard drive. All screw types are found correctly [YW20].

since the candidate cropped by the Hough circle resembles an Allen screw, and, thus the classifier claims so. The suggested approach could therefore provide tangible results by improving the cropping of candidates. However, it is a different research topic altogether, and mostly lies in the area of algorithm building.

5.6 Component Segmentation

An EOL device is assumed to be made of components, and other entities such as screws, wires and gaps. Depending on the device, the number of components inside varies. There are devices such as Ethernet routers, having mostly a PCB inside, making the task of segmentation quite straightforward. However, there are also many EOL devices that contain numerous components, having approximately 12 components inside, such as hard drive. This inevitably implies that using a classical computer vision based method isn't suitable, since the EOL device has various components that do not look like each other, or they cannot be simply detected by any kinds of filtering. Nevertheless, a mechanism is needed to identify and segment the components in the field of camera view.

Thus, the problem here could be formulated as a problem where machine learning paradigms (e.g., segmentation, classification) can be used in order to recognize the present parts of the target device. In contrast to other application domains, however, the context of recycling E-Waste adds the following constraints:

- The position estimation of parts must be precise enough to allow manipulation of small parts in the device.
- There is a strong degree of occlusion because parts are intertwined, e.g. the platters and R/W head of a hard drive.
- There is substantial intra-class variance for specific parts, depending on the device brand, model or potential damage of the part.

The *Semantic Segmentation* problem (also known as *pixelwise classification problem*) has been addressed in domains ranging from autonomous driving, human-computer interaction, to robotics, medical research, agriculture [DHS16; Dvo+17; FSB19; Li+17; Lin+19; Liu+18]. In these domains, the state-of-the-art performance is achieved using deep learning methods, especially Convolutional networks (CNNs or ConvNets) [KSH12]. Fully Convolutional Networks (FCN) have been the standard algorithm to achieve pixelwise segmentation of images [BKC17; LSD15; RFB15b] in various domains such as medicine, autonomous driving, and domestic robotics. They extend CNN by replacing fully-connected layers with convolutional

layers, allowing for arbitrary-sized input with no need for region-proposal. The trade-off however, due to the nature of the layers, is that the prediction of boundaries lacks precision. This problem is addressed by a set of improvement to the networks (e.g., multi-resolution architecture with skip connections [GF16; Lin+17a], mask refinement [Pin+16]). Additionally, pixelwise segmentation requires labeled data which has a higher cost of production [Lin+14]. Weakly-supervised methods have been developed to tackle this issue [Zha+19]. These methods iterate between learning using coarse bounding boxes as labels then refining them into more precise masks [DHS15; Kho+17]. They achieve similar performance as fully supervised methods at a lower labelling cost. Another main approach is called Region-Based Semantic Segmentation. This method relies on a pre-processing of the input image into candidate objects (region proposals) for which features are extracted. These features are then used for the classification. There is an entire family of R-CNN networks [Gir+14; Gir15; Ren+15] that have been evolving. The current state-of-the-art networks of the family is called *Mask-RCNN* [He+17] which is based on Feature Pyramid Network (FPN) [Lin+17b]. Mask-RCNN has been used by Facebook AI Research [Jou+16] as well as by medicine studies [Che+19; Cou+19; Joh18].

Mask-RCNN and its ensembled models are currently the state-of-the-art for object detection and instance segmentation. The reason for this is the region-based detection mechanism mentioned earlier. The core idea behind Mask-RCNN is to scan over the predefined regions, called anchors. RPN (Region Proposal Network), does two different types of predictions for each anchor. First is the score of the anchor being foreground, and the bounding box regression. The fixed number of anchors with the highest score are then chosen, and the regression adjustment is applied to get the final proposal for object prediction at the network head. There are other architectures that could be used for this task, such as the ensembled model Cascade-RCNN [CV18]. Through the experimental evaluation conducted it was found out that Mobile-RCNN [How+17] works better for this domain.

The aforementioned semantic segmentation problem is therefore addressed with DCNNs to detect, recognize and localize the inner parts of the device. Due to the high number of available state-of-the-art RCNN networks, a set of selected methods were trained and evaluated for this instance segmentation problem: MobileNetV1 [How+17], MobileNetV2

[San+18], Cascade-RCNN [CV18], native Mask-RCNN [He+17], and finally the ensemble model of Cascade-MobileNetV1 . All networks were trained with data augmentation turned on (as suggested by the native Mask-RCNN authors), and the same augmentation procedures were applied as suggested in the original publication [He+17]. The accuracy of a network is computed using the similarity IoU metric as shown in Equation 5.1:

$$IoUY, Y' = \frac{Y \cap Y'}{Y \cup Y'} = \frac{\sum_{i=1}^{n} \min Y_i, Y_i'}{\sum_{i=1}^{n} \max Y_i, Y_i'} \tag{5.1}$$

where A and B are respectively the sets of predicted and ground truth bounding boxes. The results of this preliminary study are available in Table 5.3. Cascade-RCNN obtains the best segmentation results, followed closely by Mobile-RCNN. The latter was selected, since the model is lighter, and thus, it is employed in the part detection module of the architecture.

DCNN Identifier	Cascading	Backbone	AP	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l	Mean-F
Mask-RCNN	No	ResNet-101-FPN	41.2	65.7	47.6	16.3	42.3	42.4	0.70
MobV1-RCNN	No	Mobilenetv1-224-FPN	56.2	76.7	64.9	20.6	54.6	59.7	0.78
MobV2-RCNN	No	Mobilenetv2-FPN	34.6	60.5	35.5	3.8	15.4	43.6	0.63
Cas-Mob-RCNN	Yes	Mobilenetv1-224-FPN	28.5	43.3	33.3	20.0	13.3	35.4	0.61
Cas-RCNN	Yes	ResNet-101-FPN	61.3	76.4	72.6	35.1	61.8	65.4	0.84

Table 5.3: IoU values of different networks evaluated on our test set. Cascade-RCNN and MobV1-RCNN seem to be the best performing ones [Yil+20].

As mentioned earlier, the standard metrics for pixel to pixel segmentation are mainly the COCO [Lin+14] average precision (AP) metrics: AP is average precision of multiple IoU values. As reported in Table 5.3, MobV1-RCNN and CAS-RCNN achieve the best segmentation performances (resp. 0.78 and 0.84). These results are achieved after 500 epochs of training on Google Colaboratory [Bis19] environment using Tensorflow [Aba16] 1.15.2. Figure 5.21 shows a sample output of the part segmentation process.

The following conclusions are made from the experiments. MobileNetV2 has stabilization issues, since the model is not able to predict on every scene (after a certain amount of training). It was observed in the training history that the loss becomes undetermined and the model remains unstable. It could be argued that the reason is the dataset (i.e. the classes are not uniformly distributed). On the other hand, cascading lighter backbones such as MobileNetV1 does not ensure a better feature extraction property.

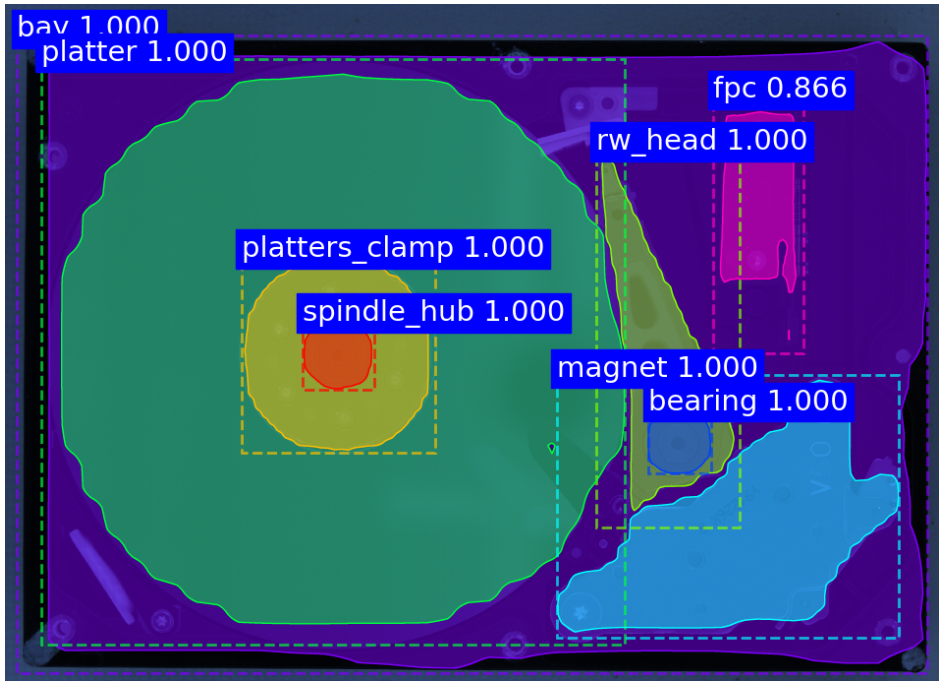

Figure 5.21: Output of the part segmentation module powered by MobV1-RCNN. Each color represents a different part predicted by the network whose names and prediction values are given. In this case, all parts and boundaries are correctly detected [Yil+20].

In fact, the modeling suggests that it actually decreases efficiency due to usage of depthwise convolutions. However, this situation validates the choice of backbone with residual nets that is clearly done in CBNet [Liu+19], Cascade-RCNN [CV18]. In short, it was determined that cascading should only be done with models that have residual backbones, such as ResNet and its variants. During our investigation we noticed that increasing cascading (with more detection target and logits loss) destabilizes the model. The use of DenseNet [Hua+17] may increase accuracy. However, with non-distributed class data as in our case, stabilization remains an issue. This is due to the fact that the last layers in the network cannot handle class-logits when the feature data is non-distributed. In the presence of multi-GPU

systems, it is therefore recommended ResNext-152 [Xie+17] be investigated for higher accuracy.

MobV1-RCNN was chosen due to the fact that it remains stable in the presence of high resolutions, unlike Cascade-RCNN that is unstable despite having high accuracy. Since the setup uses high-resolution images, MobV1-RCNN is therefore employed as the component segmentation scheme in the pipeline. However, it is noted that combining images of different resolutions in a dataset creates problems, as the model is not stable during training.

5.7 Wire Segmentation

Wire detection problem is scientifically interesting since it is challenging to address detection of wires due to their various physical properties and occlusions found in devices. Some of these wires are barely visible (due to the design of the device), posing a great challenge to finding features with high accuracy, and to automate a possible wire cutting action, which must be carried out by automated systems in high precision.

The design scheme of an automated disassembly routine may vary. Here, we consider a top-down view of the device from a camera that is part of the setup. While this view allows acquisition of images covering most of the visible device, it limits the possibility of discovering the entire wire (e.g., due to occlusions). Hence, in many cases, partial detection of wire plays a pivotal role. As there is no fixed physical size for wires in a device, the features to be caught in order to detect the wires vary greatly. A DVD player will have wide connectors as well as thin wires where both have to be detected. On top if that, sometimes the color of wires found in the device may be too challenging for them to be detected over the parts that have the same color (e.g., green wire over a green PCB). All in all, it becomes really difficult to capture all the necessary information with effective precision without using a several-thousand-dollar high-end sensor engineered for this small specific task. Moreover, the amount of wire data available is making it a less desirable domain for many to investigate. Eventually, users are left with a limited amount of annotated data from their own domains, while seeking a universal, generalizable and extensible wire detector that segments the wires found with high precision as illustrated in Figure 5.22.

In this section, a visual wire detection scheme based on deep learning

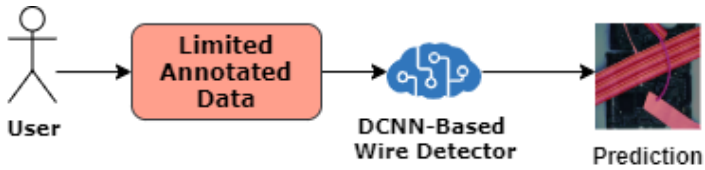

Figure 5.22: We present a DCNN-Based wire detector scheme that requires limited annotated data from the user and delivers accurate predictions for robotic manipulation tasks.

methods empowered with heavy augmentation is proposed, whose pipeline is presented in Figure 5.22. Similar to the other sections, here a comprehensive study is conducted to choose the top-scoring DCNN that yields the best accuracy using the two metrics found most suitable for the evaluation of wire detection. In order to account for the variety of wires as well as the background, heavy augmentation is used, producing massive amounts of training data generated out of a limited set of images collected. This mitigates the problem of finding specific datasets for wires and/or devices, and ensures high accuracy for the network.

The most significant contribution of this scheme is the augmentation routine and the deep learning model chosen by the comparative evaluation of state-of-the-art models. This thesis (at the time of writing), is the first to propose a scheme to detect any kind of wire found in the disassembly environment, given very limited image sets.

The first investigation found in the literature dates back to early 2000s where the authors collected images of urban areas, and used traditional image processing methods and Support Vector Machines (SVMs) [EP99] to spot the wires [Kas02] in aerial navigation tasks. The dataset was generated synthetically, combining real backgrounds with computer generated wire images. Although the work laid out the foundations and complexity of the wire detection problem, it offers no real-time application for any robotic disassembly today. This is due to the fact that the wires found in disassembly environments could not be detected simply by combining Gaussian derivatives and SVMs. Problems such as occlusion, changing brightness and background are too difficult to tackle with the paradigm suggested by this work.

An interesting attempt [Par13] came in 2013 where tangled objects (e.g.,

ropes, pipe/hose, threads) were detected using classical computer vision methods such as blurring, thresholding and Gaussian derivatives. This particular work used no learning paradigm, and achieved 74.9% accuracy. However, one must underline the fact that the work did not detect any wires, rather tangles. Therefore, it is unusable for autonomous disassembly purposes, where the body of a wire has to found and not only the tangled part of it.

One work [MMS17], which grabs attention, was the continuation of the work presented [Kas02], aiming to find wires in aerial navigation. This work is interesting because it is the first work to use convolutional neural networks in order to address the problem. The authors render synthetic wires using a ray tracing engine, and overlay them on 67000 images from flight videos available on the internet. This synthetic dataset is used for pre-training the models before fine-tuning on real data. The work achieved 73% of precision, however, it suffers from the fact that the dataset generation requires expert knowledge in ray tracing engines. Many times these software are not straightforward to use, thus, making methods based on them less desirable. Additionally, requiring a specific software endangers the future of data generation as the software may not be free after a point in the future, or stop providing support. The work nevertheless investigates available network architectures that could be used to detect wires, and plays a pivotal role in this research field. The authors report that dilated convolutional layers [YK15] were chosen since they provide a simple and effective way to gather context without reducing feature map size.

After looking into the literature, it is concluded that there is still a substantial lack in generalizable, device and wire-independent detection methods that can be used in disassembly processes without requiring massive image datasets or expert knowledge of a specific third party software.

The proposed scheme has the main blocks: data generation or *datagen* and model training. The Datagen block aims to heavily augment the limited number of user-annotated raw images and generate massive amounts of augmented images along with their annotations, for the training of the deep neural network model that is used.

5.7.1 Data Generation

It is usually very difficult or even impossible to find a dedicated dataset for specific visual tasks such as wire detection. This inevitably forces one to deal with a limited number of annotated traning data. In order to succeed under this condition, one has to enrich the amount of limited annotated images. To this end, it was advisable to use heavy augmentation routines and generate massive amounts of annotated training data from the existing dataset which contains only 100 annotated wire images gathered from online search engines. Figure 5.23 illustrates the scheme in which the datagen block plays a pivotal role.

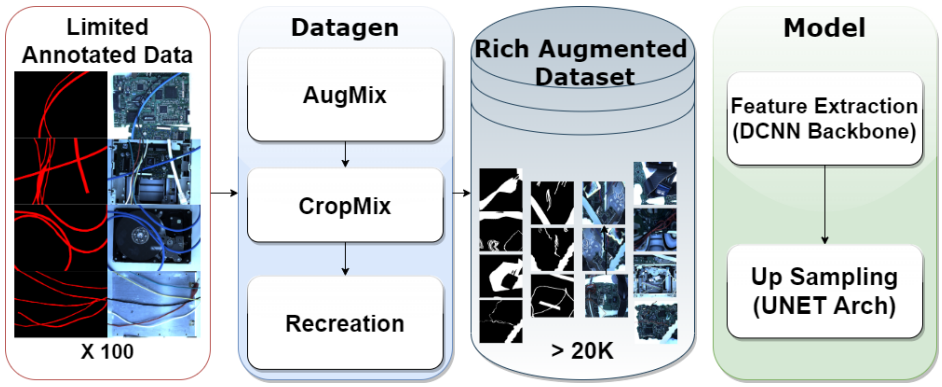

Figure 5.23: The scheme is composed of a data generation block which generates a large number of augmented images using a limited number of annotated images. An EfficientB7 [TL19] model trains on the massive number of generated augmented images.

For the data augmentation block, a 3-step routine was chosen. The augmentation library used and the functions applied are shared; the exact parameters of the augmentation functions can be found in the published source code.

Datagen flow begins with *Augmix* [Hen+19], a popular augmentation library with many options. In this step, however, only shift scale rotation, blur, elastic transform, and optical distortion are applied. In the second step, it continues with *CropMix* [TMU19], which allows manually cropped augmentations rather than automated ones to ensure variety and mask precision based on regions. Here, a 4-segment crop (per image) is preferred

to apply rotation, flipping along with generation ratio as 0.001 and dataset occupancy ratio (training) as 0.488. In the final step, custom made augmentation routine is applied, which creates a new image using 4 images applying rotation, mirroring and flipping, along with dataset occupancy ratio (training) as 0.416 and clustering as sample based. The end result is that the datagen block creates approximately 20000 images for the training data out of 100 collected raw images.

5.7.2 Model Investigation

Similar to the investigation of other schemes, it is found out that there are several classifiers in the literature one could look into when considering the task. These are InceptionV3 [Sze+15], InceptionResnetV2 [Sze+16a], Densenet201 [Hua+16], and EfficientNetB7 [TL19], which all achieve over 93% top-5 accuracy on the well-known Imagenet dataset [Den+09]. One can clearly see that the EfficientNet-B7 model stands out among the rest as illustrated in Figure 5.24. The other models picked follow a similar selection criteria. However, there is one more criterion that is taken into account, which is the number of hyperparameters. Clearly, one does not prefer to pick a good performing model that requires an enormous amount of hyperparameters such as SENet [HSS18]. Hence, the aforementioned models were finalized.

In order to conduct the investigation, a use case including digital entertainment devices (e.g DVD players, gaming consoles, etc.) was chosen. To this end, 100 top-down images of open DVD players from online search engines were collected, the visible wires were annotated by hand. Following this, approximately 20000 augmented and annotated images were generated, which served as training data for the models we investigated.

Throughout the study, only two metrics to evaluate the scheme were considered. The standard metrics for pixel-to-pixel segmentation are mainly the COCO [Lin+14] average precision (AP) metrics: AP is average precision of multiple IoU values seen. The definition of IoU between a known segmentation of n pixels, Y, and a similar set of predicted segmentation, Y' (in the binary case, i.e. where $Y_i, Y_i' \in \{0, 1\}, \forall i \in 1, n$ is as illustrated in the equation below:

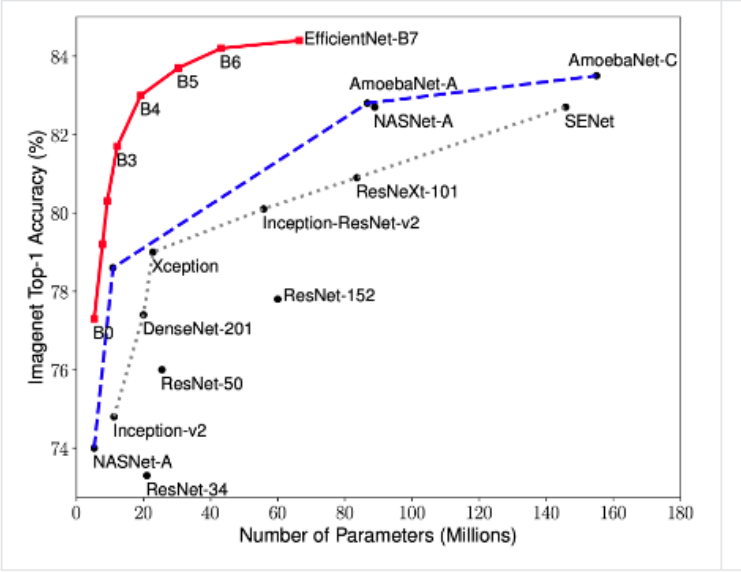

Figure 5.24: Model Size vs. ImageNet Accuracy [TL19].

$$IoUY, Y' = \frac{Y \cap Y'}{Y \cup Y'} = \frac{\sum_{i=1}^{n} \min Y_i, Y_i'}{\sum_{i=1}^{n} \max Y_i, Y_i'} \tag{5.2}$$

However, it was decided to not to only consider IoU alone, as it may be misleading in case of occluded or tangled wires. Therefore, the SSIM (Structural Similarity Index) [Wan+04] metric, known for measuring the objective image quality was also considered. The metric is based on the computation of three terms, namely the luminance term (l), the contrast term (c) and the structural term (s). The overall index is a multiplicative combination of the three terms as it is seen in Equation 5.3 as follows:

$$SSIMx, y = lx, y^{\alpha} \cdot cx, y^{\beta} \cdot sx, y^{\gamma} \tag{5.3}$$

where

$$lx, y = \frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1} \tag{5.4}$$

$$cx, y = \frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2} \tag{5.5}$$

$$sx, y = \frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3} \tag{5.6}$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. If $\alpha = \beta = \gamma = 1$, and $C_3 = C_2 2$ (default selection of C_3) the index simplifies to Equation 5.7 seen below.

$$SSIMx, y = \frac{2\mu_x\mu_y + C12\sigma_{xy} + C2}{\mu_x^2 + \mu_y^2 + C1\sigma_x^2 + \sigma_y^2 + C2} \tag{5.7}$$

All of the experiments conducted were evaluated based on these two metrics.

	Metric					
	SSIM			**IoU/F1**		
Model	**Max.**	**Mean**	**Min.**	**Max.**	**Mean**	**Min.**
EfficientNetB7	**0.956**	**0.877**	**0.761**	**0.988**	**0.952**	**0.897**
InceptionV3	0.944	0.863	0.758	0.977	0.947	0.894
InceptionResNetV2	0.941	0.859	0.743	0.983	0.943	0.886
DenseNet201	0.940	0.862	0.747	0.978	0.945	0.891

Table 5.4: Evaluation of the state-of-the-art models. EfficientNetB7 is proven to be the most suitable model with a high SSIM and IoU score.

Table 5.4 shows the evaluation of the state-of-the-art models based on the aforementioned metrics. One can clearly notice that the model EfficientNetB7 [TL19] scores slightly better in both metrics. The results clearly indicate that EfficientNetB7 is doing a better job at feature extraction from the given images (even with low resolution, low-feature conditions).

Having noticed the slightly better accuracy provided by the EfficientNetB7, inferences on the images of a different device were made, such as a gaming console that is open for disassembly purposes, revealing the wires it has inside. The model, although trained on DVD player wires, was able to make good predictions on the gaming console wires, marking their locations for a possible robotic manipulation action. Figure 5.25 illustrates this use case.

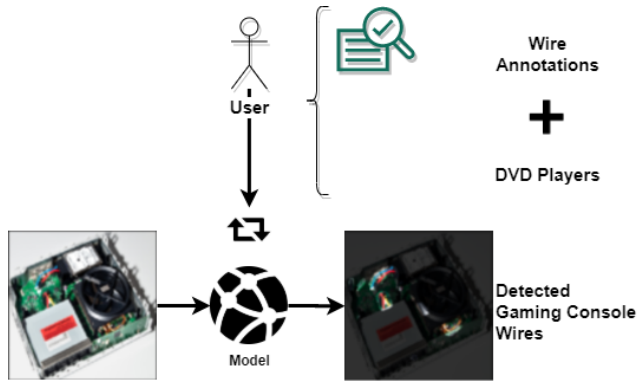

Figure 5.25: We investigated the state-of-the-art models' capabilities on limited amount of data. After training the models for wires of DVD players, we infer on gaming console wires. Both devices belong to the same family of devices, despite of having different inner layouts.

We used the state-of-the-art UNET [RFB15a] model as our up-sampling backbone for all our feature extractor models, which is illustrated with Figure 5.26.

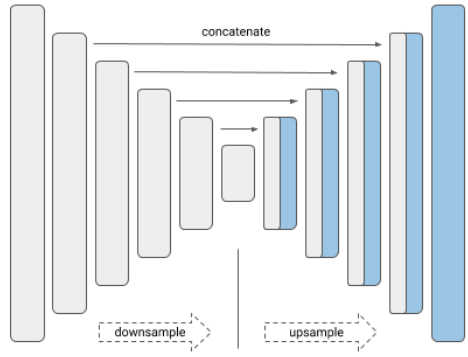

Figure 5.26: UNET architecture used for upsampling and generation of the segmentation mask.

5.7.3 Training the Model

After choosing the model, a new dataset of high (4K) resolution top-down images of wires that could be found in disassembly environments were collected. Various wires taken out of devices, including those that connect devices to power supplies. As background, the most commonly encountered ones such as PCBs, device lids and bays, as well as the work station surface were chosen. Having collected 130 various wires images, 100 of them were split for training, 10 for validation and 20 for testing. Below the details of the experimental evaluation are provided and the final model is presented.

This work has been developed on the Google-provided TPU v2 for training the model via Colaboratory [Bis19] environment which has 64 GB High Bandwidth Memory (HBM) and provides 180 TFlops computing power. The validation set was taken to be 1/10 of the training dataset and training was done with early stopping enabled callbacks so that the model did not overfit. The model was trained with 4 generations of data each time taking one fourth of the total data (i.e- the augmentations are tuned in such a way that every time about 50 percent of training data is completely new every time). No transfer learning was used and the model was trained from scratch. For the final model, however, the best of these 4 weights were taken to train on the whole training dataset. As the MSE loss graph shows in Figure 5.27, the model reaches stable losses very quickly and converges to a final point where a plateau stage can be encountered before stopping. Similarly, in Figure 7.18 of Appendix 7.1 the change of SSMI through epochs is presented.

Summary of the experimental results, with regards to performance of each classifier against the validation set, can be seen in Table 5.4.

Model	Metric	Min.	Max.	Mean
EfficientNetB7	SSIM	0.80	0.98	0.91
EfficientNetB7	IoU/F1	0.89	0.99	0.97

Table 5.5: Evaluation of the trained EfficientNetB7 model on the test data, using SSMI and IoU metrics.

From the collected results in Table 5.5, one can conclude the following: EfficientNetB7 out performs any other state of the art models that are

capable of semantic segmentation. The model outputs show high similarity and attain a good IoU score as well.

It was observed that the resolution of the images inferred must also be of the same resolution of training data, because lowering the resolution creates completely different feature maps compared to what the model was trained on. Figure 5.28 illustrates some of the detections on the test set by our detector. The scheme can handle delicate cases such as wires with tangles, and partial occlusions, which are frequently encountered cases during the disassembly of an electronic device. The model proves to be robust, handling such delicate situations while maintaining high accuracy. Minimum SSIM and IoU of 0.80 and 0.89 are reported, respectively.

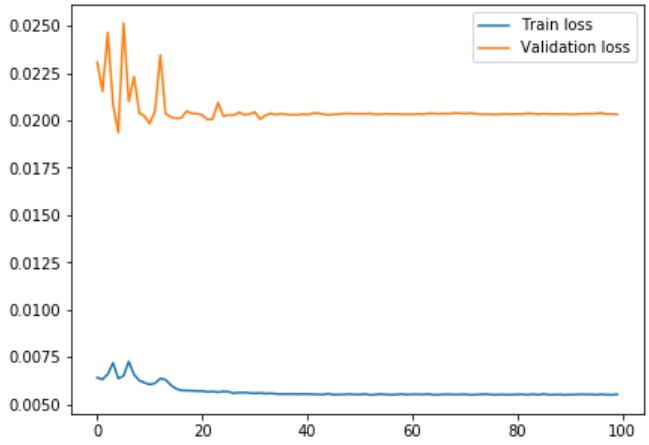

Figure 5.27: Loss in MSE of our model with the pre-trained weights from the K-Gen dataset.

Last but not least, the trained model was tested on devices that the network has never seen before, such as heat cost allocators. Figure 5.28 illustrates the found wires in these devices, as well as other cases where wires were laid out by a human over hard drives. The results prove that the model can be used in disassembly environments where the target device was not seen before, which is usually the case.

Wire detection is a fundamental problem in disassembly environments. The problem itself is a challenging one, since wires have variable physical properties such as geometry, color and thickness, and not every electronic

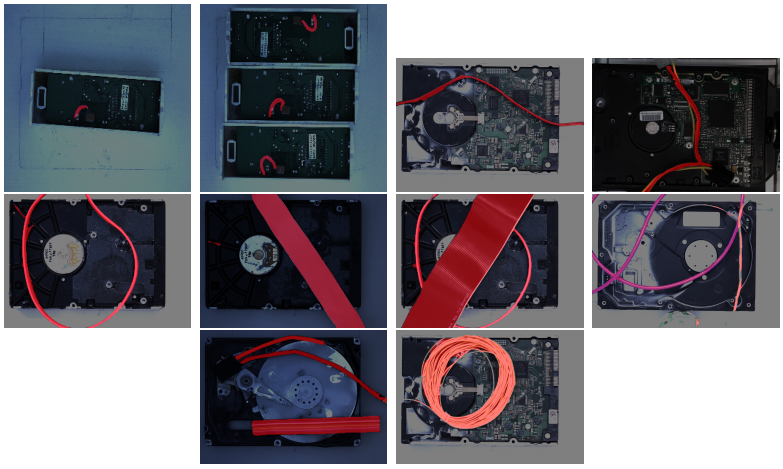

Figure 5.28: The model has a clear generalization ability, since it is able to detect wires found in devices that it has not seen before. It was tested on the back side of heat cost allocators, as well as both sides of hard drives with arbitrarily added wires on them.

device has the same type of wires. It is pointed out that these were the challenging features, the reasons why the previously developed methods were not useful as a general solution to this problem. A model was proposed, which is based on the heavy augmentation and deep convolutional neural networks. The scheme easily lets the users operate the system for any device of their choice, as long as the user manages to collect a limited number of hand-annotated images, which is determined to be approximately 100 for accurate detection. An investigation with the-state-of-the-art models was conducted and EfficientNetB7 was picked based on the results of a use case we selected for the evaluation of these models. After choosing the model, a limited amount of dedicated data was collected using real wire backgrounds that can be found in disassembly environments.

After generating a massive amount of data via the datagen block and training the model one more time, one could demonstrate that the system achieves high accuracy on scenes with random wires as well as on a new device (thermostat) that was not seen by the network before. The evaluation was quantified with testing images of disassembly scenes, containing different models and sizes of wires. Experimental evaluation results of the

scheme was documented. The dataset, as well as the implementation are published to facilitate further research.

5.8 Gap Detection

The approach for the proposed gap detection strategy was partially developed as a Bachelor's thesis project [Bri20] in the neuroscience group of University of Göttingen. Later on, with gradual contributions, it became the proposed gap detection strategy. It consists of a computer vision pipeline to process point cloud information constructed by stereo cameras. There is no machine learning paradigm used, rather, a six-stage pipeline with each stage processing the acquired point cloud to spot the gaps among the data points. In the first step, a passthrough filter is applied to find the ROI. Following this, the point cloud is denoised, a threshold is found and used to find possible gap points. The possible gap points are then clustered to find individual gaps. A convex hull is calculated for each gap to estimate their volume and filter extremely small gaps. Figure 5.29 illustrates the pipeline.

5.8.1 Passthrough filter

In the first processing step, the point cloud is filtered by a passthrough filter to find the region of interest. The region of interest should only contain the device to apply the detector to. Insignificant points (e.g the surface the device is lying on or other objects close to the device) have to be cut away. The filter works by setting lower and upper limits for each of the three axes. Points below the lower limit and points above the upper limit for the specified axis are discarded. Finding these limits or thresholds is done manually in the implementation which shows the real-time result of the passthrough filter. While the passthrough filter cuts away unimportant points, noise created from the stereo cameras persist. To remove the noise, denoising is applied in the next step.

5.8.2 Denoising

Gaps are usually defined by a cluster of points with points belonging to the same gap having only points of this gap in its neighborhood. Far outliers

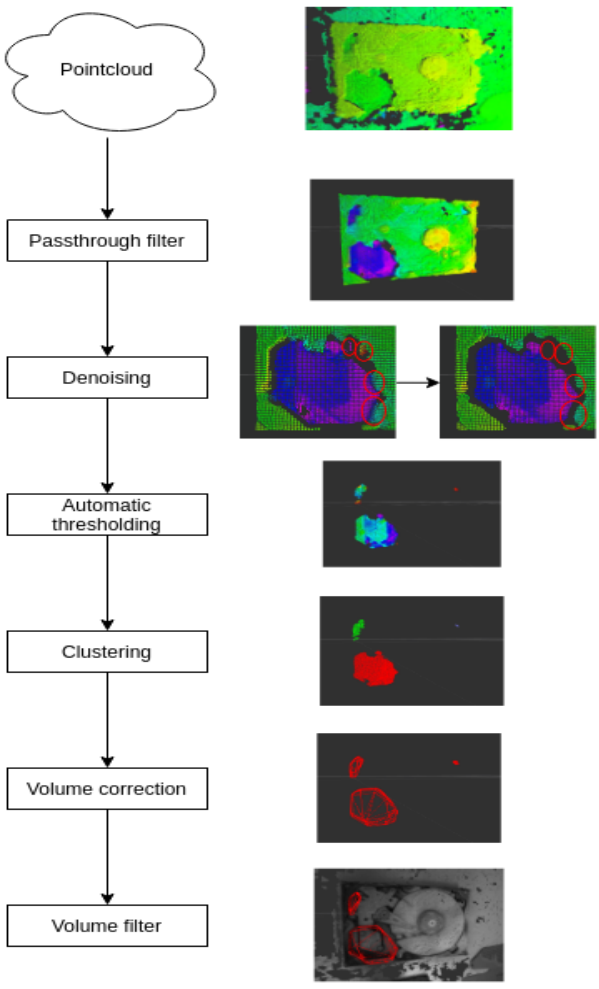

Figure 5.29: Proposed gap detection pipeline [Yil+20].

from these clusters are often noise created by the stereo cameras. This noise can conflict with the volume correction step later in the pipeline, creating a convex hull that is greater than the real gap size. The noise can also shift the threshold found by the automatic thresholding step. The automatic thresholding is finding a midpoint between the distribution of surface points and gap points. If there are enough outliers on one side, the

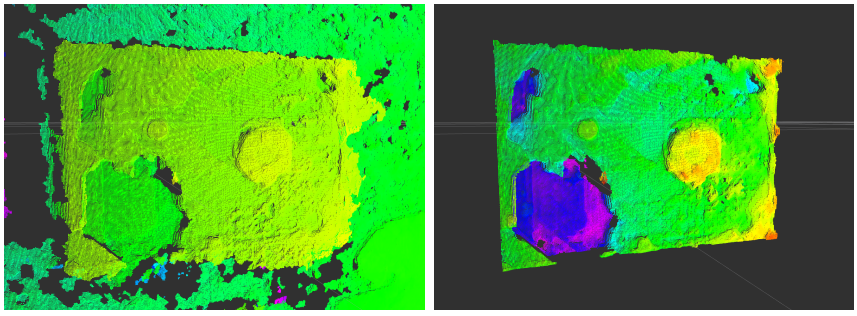

Figure 5.30: Result (right) of the passthrough filter on the input (left) point cloud [Bri20].

threshold can shift in this direction increasing the number of false positive gap points. Therefore, it is important to denoise the point cloud before further processing [Bri20].

The detector is using sparse outlier removal [RC11] (also called statistical outlier removal) to remove the outlier noise. It detects noisy outlier points based on their distance to other points. This is done by calculating the mean distance for every point to the nearest k neighbors [Bri20]. The resulting distribution is assumed to be Gaussian with a mean μ and a standard deviation σ. Based on this distribution, a global distance mean interval $[\mu - \alpha * \sigma, \mu + \alpha * \sigma]$ is computed, where α depends on the size of the analyzed neighborhood. A point will be trimmed if its mean distance is outside this interval [Bri20].

In figure 5.31 a comparison before and after denoising is given. In this close up of a hard drive point cloud, we can see points with a bigger distance to the camera colored in blue and violet, while green points indicate a smaller distance. Sparse outlier removal deleted noise points in the upper right, as well as points that are connecting lower parts of the hard disk drive with higher elevated points [Bri20]. While these "bridges" between low and high parts are not always noise, a clear separation between areas makes clustering easier.

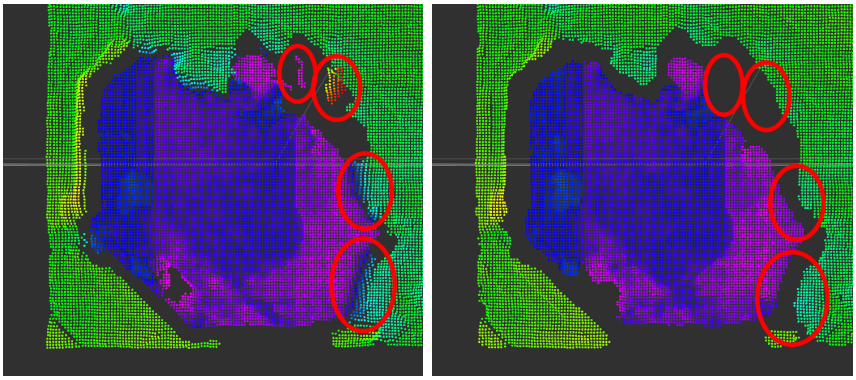

Figure 5.31: Close up of a HDD point cloud before and after sparse outlier denoising [Bri20].

5.8.3 Thresholding

With a denoised point cloud of the device, one can start to filter out points that do not belong to a gap. An important property of each point is its depth, given by the depth axis. Depth refers to the height on which a point is, looking at the device from a side view (shown in figure 5.32). The stereo setup is capturing the device from above, therefore the depth axis is the axis going from the camera, through the surface of the device, to the bottom of it [Bri20]. The depth of a point is important because points belonging to a gap are at the bottom of the device. These points will have a higher depth than points belonging to the surface. Using a well-chosen threshold, one can split the point cloud based on the depth of points into potential gap points and insignificant surface points. In the sense of traditional thresholding, where an image is threshold by its intensity values, we threshold the point cloud into a foreground of surface points and a background of potential gap points [Bri20].

Setting this threshold is tedious and sometimes has to be done again for a new device [Bri20]. If we order the points into a histogram based on their depth, we can observe a bimodal distribution (shown in Figure 5.33) with two modes, one for the surface points and one for the gap points.

On this distribution we can apply automatic thresholding methods such as Otsu Thresholding to determine a threshold between the two modes.

Figure 5.32: The depth axis of a device for a setup where the camera is above the device [Bri20].

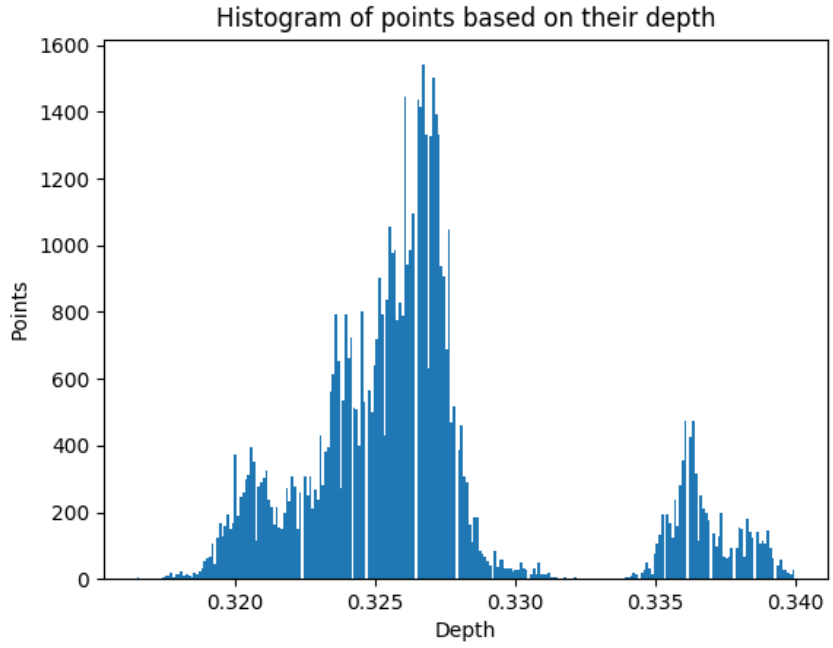

Figure 5.33: Histogram for the depth of points from a hard disk drive [Bri20].

For each mode or class Otsu's Binarization minimizes intra-class variance, or equivalently, maximizes inter-class variance between them to find a threshold [Bri20]. This threshold is then used to split the point cloud, such that we get all points with a high depth that are candidates for a gap.

A similar algorithm [Gla93; PM66] achieved even better results. Instead of maximizing the inter-class variance, the minimum algorithm takes the minimum in the valley of the histogram between the two modes. The corresponding value is then used as the threshold.

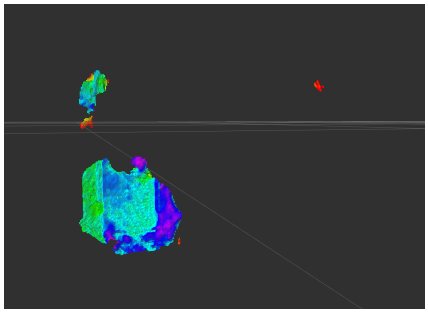

Figure 5.34: Point cloud after automatic thresholding [Bri20].

5.8.4 Clustering

Currently, our data only consists of single points without information about how these points form gaps. Points belonging to a gap often lie in close proximity to each other, opening an opportunity for a clustering approach to label points. Since devices often have multiple gaps, this would allow us to separate points belonging to individual gaps [Bri20].

There are a lot of algorithms that could be used for clustering. However, for our problem an algorithm has to fulfil three main properties:

- Number of clusters: Since the gap detection is used on different devices with different gap counts, the algorithm has to find the number of clusters/gaps only by the data structure.
- Scalability: After the filtering and denoising steps, there are still several thousand points. The clustering algorithm should be scalable to the number of points, as well as the number of clusters.
- Robustness: Gaps can vary greatly in size. A clustering algorithm should be able to find gaps with variable sizes.

The first algorithm to try is often K-Means [MRS08]. It is simple and it runs with a runtime of $\mathcal{O}ni$ (for constant number of dimension and clusters), where n is the number of samples and i the number of iterations until convergence. Additionally, it is fast and it scales well with the number of samples. The problem with K-Means, however, is that it needs the number of clusters K as a parameter. For the gap detector, this means the number of gaps has to be known in advance. This is not possible, since the number of gaps varies from device to device. Thus, K-Means would be an appropriate choice for the problem, since for every device the number of gaps has to be manually entered [Bri20].

Besides K-Means, two algorithms have been proven to be especially applicable for the problem: DBSCAN and its extension HDBSCAN. DBSCAN and HDBSCAN are density-based algorithms, viewing clusters as densely packed regions of points. Instead of choosing a fixed number of clusters such as K-Means, these algorithms estimate a number of clusters from the data structure. Both algorithms scale well to the number of points with a time complexity of $\mathcal{O}n \log n$ [MHA17b; Tan+18]. Due to their structure, HDBSCAN and DBSCAN are classifying points into clusters as well as noise. Points classified as noise are deleted from the result, which gives these algorithms built-in denoising.

DBSCAN's most important parameter is the *eps* radius. This radius is used to determine which points are in a neighborhood of a point, by which, DBSCAN classifies points into clusters [Bri20]. The *eps* radius is also the main difference between DBSCAN and HDBSCAN. For DBSCAN, one has to specify an *eps* radius before clustering, HDBSCAN estimates this distance automatically.

HDBSCAN, on the other hand, does not only estimate *eps*, but it also uses different radii for different clusters [Bri20]. It must be underlined that DBSCAN does not do this, it rather only uses a single radius. This has disadvantages for clusters with varying densities where in one cluster points have a high density and lie close together, and in another cluster points have a low density and lie farther away [Bri20]. For such data structures DBSCAN often faces problems with the fixed *eps* radius. If the radius is too small, clusters with a lower density are detected as noise. If the radius is too big, clusters are fused together. HDBSCAN solves this problem by using different *eps* radii for each cluster. Naturally, this increases the

robustness of the algorithm on different data sets. Therefore, for the gap detector, it could improve the detection results and improve its usability by getting rid of an additional parameter [Bri20].

Although HDBSCAN looks to be the best choice for clustering, in practice DBSCAN has advantages in certain situations over HDBSCAN. Both algorithms can be used in the gap detector. An extensive review of the results of these algorithms on multiple devices is given in Chapter 6. More information about K-Means, DBSCAN and HDBSCAN is also found in Chapter 4.

Algorithm	Scalability	Number of clusters	Denoising	Robustness
K-Means	$\mathcal{O}ni$	parameter	none	-
DBSCAN	$\mathcal{O}n \log n$	automatic	build-in	-
HDBSCAN	$\mathcal{O}n \log n$	automatic	build-in	variable cluster densities

Table 5.6: Comparison of the different clustering algorithms.

In Figure 5.35 one can see the three algorithms on the points found by thresholding. The used hard drive has three gaps that have to be identified. The cluster algorithms have to determine a label for each point, such that points of the same gap are clustered together. Both DBSCAN and HDBSCAN achieved the same desired result [Bri20]. K-Means, even with a given number of clusters, is not able to separate the points belonging to different gaps.

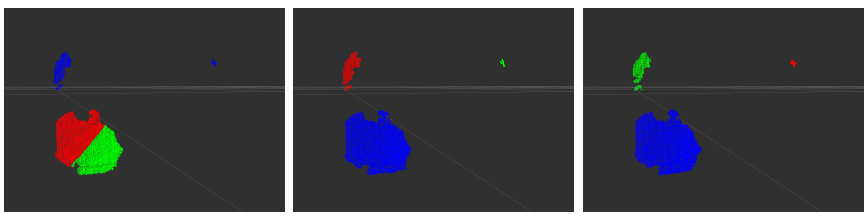

Figure 5.35: K-Means, DBSCAN and HDBSCAN cluster results.

5.8.5 Volume Correction and Convex Hull Calculation

As the clustering step identifies individual clusters as gaps, it finishes the identification of gap points. It is then known where each gap lies, and one could extract information from them.

One type of information is the volume of the gap. Since the disassembly system needs some space for the levering process, we can filter out gaps that are too small based on their volume. To find a good estimate of the volume we need to find the outline of the gap to then calculate the volume inside [Bri20]. A convex hull algorithm can find the convex hull for a set of points and also calculate the volume of the convex hull. Using the identified gap points from clustering, we can find a convex hull for each gap using a convex hull algorithm. The result of this operation is seen in Figure 5.36. Since the gap points only cover the bottom of the gap, the found convex hull also only covers the bottom of the gap. Generally, the volume of the gap should stretch to the surface of the device. Therefore, we need to correct the convex hull to reflect the full volume [Bri20].

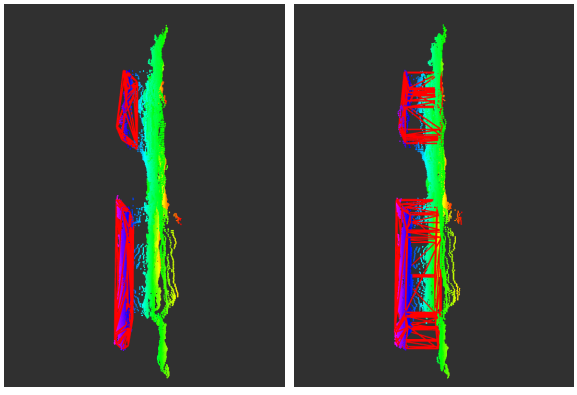

Figure 5.36: Side view of the point cloud with and without volume correction [Bri20].

Correction is done by adding artificial points to the set of gap points [Bri20]. The convex hull algorithm will use them as the new outline points to create a better estimate. In a first pass, the convex hull algorithm is used on the identified gap points to get the boundary points, marking the shape of the gap bottom part. The convex hull has to stretch until the height of the surface to get the full volume space [Bri20]. The height of the surface is estimated by taking a median of the surface points. Surface points are all points cut away in the thresholding process. Given the height of the surface, we can now duplicate the boundary points from the first pass and set them to the surface height. In a second pass of the convex hull

algorithm, a corrected convex hull is found that stretches to the surface of the device [Bri20].

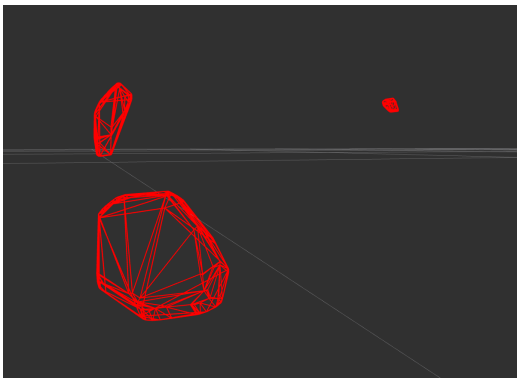

Figure 5.37: Result after the volume correction and convex hull calculation [Bri20].

5.8.6 Volume filter

Given the convex hull and the volume of a gap, we can now filter based on the volume. A simple threshold filter discards gaps with a volume below the threshold, such that the threshold can be set individually for different devices or levering operations.

The volume filter concludes the detection part of the pipeline. For each detected gap we now have a volume and the boundaries from the convex hull [Bri20]. Additionally, the detector also calculates a center for each gap. A gap center is defined by using the vertices of the gap boundaries and calculating the midpoint between the minimum and maximum coordinate for each axis. Finally, these information are bundled and made available for the robot.

5.8.7 Evaluation

Gap detection module has been evaluated by comparing the found gaps with the user annotated point clouds. Twelve hard drives were used in the experiments. Many parts of the hard drive have reflective surfaces which disrupt the stereo imagery and create noise or holes in the point clouds

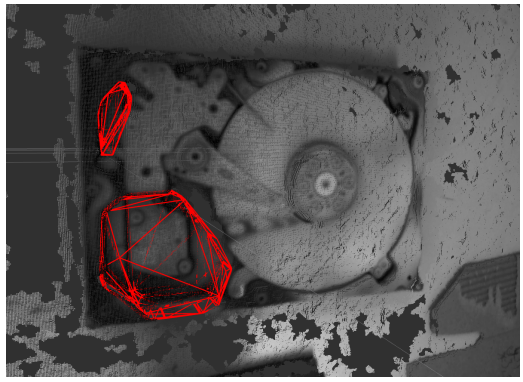

Figure 5.38: Detected gaps after volume filtering [Bri20].

[Bri20]. For this reason, these parts were sprayed with an anti-reflective spray, improving the results of the point clouds taken. Furthermore, to simulate different stages of the disassembly process, parts such as the platter, the read/write arm, the spindle, and other small parts were removed in some hard drives. In addition, some hard drives had the lid still on, resulting in a situation with no gaps [Bri20]. Ideally, the detector should identify these situations and output no gaps. The point clouds were then processed by using the gap detector with manually set region of interest boundaries. For the statistical outlier, denoising the number of neighbors to analyze for each point was set to 150 and the standard deviation multiplier to 30. Both DBSCAN and HDBSCAN were used on the hard drives. DBSCAN used a *eps* distance of 0.006 and HDBSCAN's minimum cluster size parameter was set to 15 [Bri20]. The volume filter was set to filter gaps smaller than a minimum volume of 0.1cm^3 and gaps bigger than 50.0cm^3.

All point clouds taken by the stereo cameras were manually annotated using a Semantic Segmentation Editor. These ground truth annotations consist of pointwise segmentations of each gap in a device. To get comparable point numbers, statistical denoising was run on the cloud before segmentation [Bri20]. Resulting point clouds had on average 51000 points, with the smallest having only 37418 points and the biggest point cloud having 63283 points. Based on the annotations, the volume correction approach mentioned was used to estimate a volume. Given the estimated volume and the number of points for the annotations, the gap detector

was used to process the same point clouds. For all devices the same set of parameters was used [Bri20]. Tuning these parameters on a device-to-device basis could improve the results and correct misclassifications. However, since the gap detector should have a degree of generalization, the parameters are fixed for every device. Additionally, the detector was used twice on each point cloud, once using DBSCAN in the clustering step of the approach and once using HDBSCAN. For each identified gap, the detector outputs the total number of points the gap has, as well as the volume and the center. These identified gaps were then matched to the annotation gaps by calculating the euclidean distance between their centers and discarding gaps that had a bigger distance than 0.025. If multiple gaps remained, the one with the highest volume was chosen. The detector was then evaluated by the detection rate of gaps, as well as the quality of correctly identified gaps [Bri20]. This is motivated by the fact that the system should not merely find the gaps but also estimate their boundaries and volume. Finding perfect center points for every gap does not matter if the estimated boundaries are completely off. An automated disassembly system has to have information about the location of gaps, as well as whether there is enough space for levering.

DBSCAN vs. HDBScan

For the 12 hard drives (with a total of 21 gaps) used during evaluation, the gap detector identified 18 gaps using DBSCAN and 22 gaps using HDBSCAN. A closer look at the precision reveals how HDBSCAN produced more gaps than DBSCAN. For DBSCAN, out of 18 gap predictions, 15 were true positives and 3 false positives, or a precision of 83%. In comparison, out of the 22 gaps found by HDBSCAN, 16 were true positives and 6 false positives resulting in a precision of 72.73% [Bri20]. An explanation for the lower precision of HDBSCAN can be seen looking at the individual results. HDBSCAN split up single gaps into smaller clusters. Table 5.7 demonstrates such a situation. The annotation has one gap with a volume of 3.417 cm^3. Using DBSCAN the detector correctly identified this gap with a volume difference of 0.327cm^3. Using HDBSCAN in contrast resulted in three gaps. In this case, HDBSCAN split up the gap into three clusters instead of producing one single cluster.

Figure 5.39 shows the volumes found using DBSCAN and HDBSCAN,

	Gap	Points	Volume
Annotation	1	456	3.417 cm^3
DBSCAN	1	492	3.09 cm^3
HDBSCAN	1	289	2.132 cm^3
	2	94	0.558 cm^3
	3	46	0.156 cm^3

Table 5.7: Example hard drive where HDBSCAN splits up the gap into three smaller gaps [Bri20].

as well as the annotated 13 gaps. It must be stated that considered gaps consist only of gaps that were identified correctly by DBSCAN and HDBSCAN. Thus, gaps that were only found by one method or by neither were not considered. Volumes of these gaps range from the smallest gap with 0.74 cm^3 up to the biggest one with 46.2cm^3.

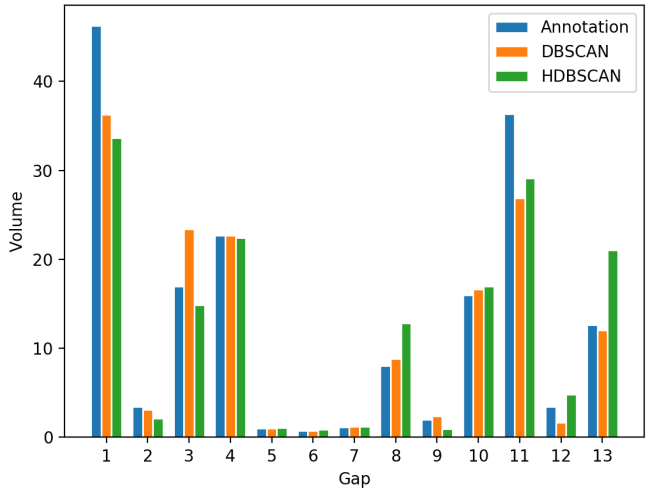

Figure 5.39: Volume comparison of gaps between the detections of DBSCAN, HDBSCAN and the user annotated gaps [Bri20].

A difference between the two algorithms is also noticeable in the quality of the correctly identified gaps. Table 5.8 lists the difference in the total

number of points and volume found using DBSCAN and HDBSCAN against the annotated data. It was found that DBSCAN misses 11.18% of points while HDBSCAN only misses 3.11%. Similarly, gaps identified using DBSCAN had 9.77% less volume than the annotation. HDBSCAN was slightly better with 5.09% [Bri20].

		Points	Volume
DBSCAN	Annotation	43378	174.27cm^3
	Difference	-11.18%	-9.77%
HDBSCAN	Annotation	42120	172.27cm^3
	Output	40754	163.37cm^3
	Difference	-3.11%	-5.09%

Table 5.8: Difference in total volume and total points against the annotated data. The experiment is conducted only on the correctly found gaps [Bri20].

Table 5.9 shares the error rates found through the experiments conducted with DBSCAN, while Table 5.10 shares the error rates for the experiments conducted with HDBSCAN. An error was defined as

$$|1 - \frac{detector\_output}{annotation\_output}| \tag{5.8}$$

to compare the results. Since the gap detector should output good estimates for any gap, the error per device is averaged to summarize the detector error. Furthermore, the standard deviation $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i - \mu^2}$ and variance σ^2 are calculated. Not detected devices, marked in Table 5.9 and 5.10 with dashes below the average error, were not used in the calculations towards accuracy.

On average, DBSCAN produced an error of 23.98% in the number of points with a standard deviation of 26.70% and a variance of 712.7 [Bri20]. In terms of volume, an average error of 24.30% was produced with a standard deviation of 28.14% and a variance of 791.8. Despite the lower total differences shown before for HDBSCAN, the average error of HDBSCAN is higher, with an error of 25.88%, standard deviation of 25.92%, and variance of 672.1 in the number of points. In terms of volume,

the average error was 27.41% with a standard deviation of 23.14% and a variance of 535.57.

	Annotation		Detector		Error	
Gap	Points	Volume in cm^3	Points	Volume in cm^3	Points	Volume
1	9464	46.243	7382	36.3	22.00%	21.50%
2	456	3.417	492	3.099	7.89%	9.31%
3	2568	16.984	2711	23.414	5.57%	37.86%
6	6414	22.712	6532	22.714	1.84%	0.01%
7	343	0.99	394	0.99	14.87%	0.00%
9	287	0.746	319	0.725	11.15%	2.82%
10	277	1.159	317	1.196	14.44%	3.19%
11	3322	8.032	3440	8.854	3.55%	10.23%
12	350	1.974	426	2.421	21.71%	22.64%
15	4967	15.99	5372	16.622	8.15%	3.95%
16	1399	2.797	103	0.136	92.64%	95.14%
17	9729	36.339	8446	26.874	13.19%	26.05%
18	1227	3.45	528	1.677	56.97%	51.39%
19	2237	12.663	1978	12.043	11.58%	4.90%
20	340	0.773	88	0.189	74.12%	75.55%
				Average Error	23.98%	24.30%
4	347	1.443	-	-	-	-
5	182	0.386	-	-	-	-
8	110	0.111	-	-	-	-
13	191	0.438	-	-	-	-
14	266	0.453	-	-	-	-
21	134	0.125	-	-	-	-

Table 5.9: Comparison of the proposed detector using DBSCAN and the annotated data [Bri20].

In conclusion, the detector has a slightly lower gap detection rate with 15 out of 21 using DBSCAN against the 16 out of 21 of HDBSCAN. However, there is a trade-off. The higher detection rate of HDBSCAN also produces a higher number of false positives [Bri20]. The detection pipeline produced gaps with an accuracy of 72.73% using HDBSCAN against the 83.33% using DBSCAN. Furthermore a detection pipeline using DBSCAN was able to produce higher quality gaps, in terms of difference to annotations, with an average error of 23.98% in the number of points and 24.30% in volume, whereas the pipeline using HDBSCAN had a higher error of 25.88% in the number of points and 27.41% in volume. Therefore, the evaluation

	Annotation		Detector		Error	
Gap	Points	Volume in cm^3	Points	Volume in cm^3	Points	Volume
1	9464	46.243	6904	33.668	27.05%	27.19%
2	456	3.417	289	2.132	36.62%	37.61%
3	2568	16.984	2191	14.895	14.68%	12.30%
4	347	1.443	363	1.249	4.61%	13.44%
6	6414	22.712	6467	22.449	0.83%	1.16%
7	343	0.99	422	1.045	23.03%	5.56%
9	287	0.746	356	0.87	24.04%	16.62%
10	277	1.159	324	1.204	16.97%	3.88%
11	3322	8.032	3450	12.854	3.85%	60.03%
12	350	1.974	196	0.951	44.00%	51.82%
13	191	0.438	54	0.124	71.73%	71.69%
15	4967	15.99	5283	16.952	6.36%	6.02%
17	9729	36.339	8494	29.151	12.69%	19.78%
18	1227	3.45	1413	4.793	15.16%	38.93%
19	2237	12.663	4484	21.029	100.45%	66.07%
21	134	0.125	118	0.133	11.94%	6.40%
				Average Error	25.88%	27.41%
5	182	0.386	-	-	-	-
8	110	0.111	-	-	-	-
14	266	0.453	-	-	-	-
16	1399	2.797	-	-	-	-
20	340	0.773	-	-	-	-

Table 5.10: Comparison of the proposed detector using HDBSCAN and the annotated data [Bri20].

concluded that the detection module using DBSCAN is preferable over the one using HDBSCAN. Thus, the accuracy of gap detector for hard drives is taken as 0.83 [Bri20].

Last but not least, some delicate conditions have been observed. Out of the twelve test drives in the annotation, three drives did not have any notable spaces that could be called gaps. These drives are definitely more challenging for the gap detector to operate with, because the depth values acquired do not form a clear bimodal histogram. This disrupts the automatic thresholding and results in misclassified points [Bri20]. These situations can occur in the beginning, when the lid is not yet separated, as well as when the backside of the hard drive is shown instead of the front side(where the gaps are located). A similar situation occurs when the drive

is completely disassembled. Without the parts inside, the hard drive is left with a single big gap. While one could specify this as a gap, it is not important for the levering process anymore, since there are no parts left for levering. This kind of big gaps are still picked up by the thresholding process. However, they are filtered out by the volume filter. For the three tested drives, the detector could deal with all of them, resulting in zero identified gaps [Bri20].

5.9 Bookkeeping

In the context of disassembly, a bookkeeping mechanism aims to register the status of every recognized component in the scene. This is carried out by analysing the predicted pixel-level changes between component boundaries found by the component segmentation block explained in Section 5.6. The mechanism accounts for multiple situations that are explained below. It also allows user to specify the sensitivity of the change it should consider before registering. This is a required feature since, every change is detected by conducting pixel-wise comparison of component boundaries and regions in consecutive frames, meaning that a larger device (and components) may require a less sensitive analysis of changes, as a few pixels of change in large component's boundaries may not exactly mean a misplacement or failed action. If the EOL device is large, then a little touch on its part is not worth registration. On the other hand, if the EOL device is a small one (as in the case of hard drives), then a few pixels may mean more, given some parts such as a *spindle hub* are relatively small, meaning that every pixel should count towards the change threshold.

- **Difference in List of Parts:** When the lists of parts are different between consecutive frames, introduction or removal of those parts are registered.
- **Difference in Locations of Parts**: When the parts appear to be in different locations (if their center points moved more than the user-specified margin) between consecutive frames, this change is registered.

The bookkeeping mechanism has a timer, additionally allowing the system to not consider a frame that is acquired beyond the user-specified

time interval. This is a needed feature as well, since the registration should not occur between a frame from a previous system run. This user-specified parameter is set to 5 minutes by default, considering a frame as consecutive if and only if the frame is acquired within this time. If not, it registers the frame as the primary frame, and starts the timer for another 5 minutes, as explained in algorithm above.

5.10 Implementation

This section is a small overview of the implementations of the above algorithms. It is divided into subsections explaining the significant technical details regarding each vision block introduced in this chapter, illustrated in Figure 5.8. First, however, the algorithm to collect predicates has to be mentioned.

As the algorithm below illustrates, collection starts by ensuring that the input point cloud and RGB image are acquired by tilting the mechanism to get the desired angle. The point cloud is analysed through the `detectGaps()` function, followed by the functions `segmentParts()` and `detectScrews()`, yielding the center points of the required entities. It must be kept in mind that the `detectScrews()` function inherently handles the classification task, and thus, is not illustrated in the algorithm. Finally, calibration information is utilised and a common frame of reference is agreed while merging all entity information. Acquired predicates are then ready for further use. As mentioned previously, the underlying communication is carried out by ROS facility, with each functionality encapsulated as ROS nodes or ROS services, depending on the computational load.

5.10.1 Gap Detection Node

The gap detection node consists of two modules: the preprocessing module and the detection module. They interact with each other through the ROS ecosystem. The preprocessing module is written in C++ and takes care of defining a region of interest and denoising the incoming data. The denoised data is then handled by a Python-based detection module which performs the automatic thresholding, clustering and postprocessing work. Usage of C++ and Python is due to the libraries used in each module.

Algorithm 1 Proposed algorithm to collect predicates using vision blocks in State Estimation.

$c_p, b_p, m_p :=$ ▷ part centers, boundaries, masks
$c_s, b_s, m_s :=$ ▷ screw centers, boundaries, masks
$c_g, b_g, v_g :=$ ▷ gap centers, boundaries, volumes
$I, P := NULL$ ▷ I: Input Monocular Image, P: Input Pointcloud
$C_m, C_s := NULL$ ▷ C_m: Monocular Calibration Info, C_s: Stereo Calibration Info
$predicates =$
procedure COLLECT PREDICATES
    if $hddTable.State = 0$ **then**
        $hddTable.changeStateangle = \theta_{Stereo}$
        $P \leftarrow getPointcloudP$
        if $P \neq NULL$ **then**
            $c_g, b_g, v_g \leftarrow detectGapsP$
        end if
        $hddTable.changeStateangle = \theta_{Monocular}$
        $I \leftarrow getRGBImageI$
        if $I \neq NULL$ & $hddTable.State = 0$ **then**
            $c_p, b_p, m_p \leftarrow segmentPartsI$
            $c_s, b_s, m_s \leftarrow detectScrewsI$
        end if
    end if
    $C_m, C_s \leftarrow getCalibrationInfo$
    $predicates \leftarrow mergeAllInfoI, P, C_m, C_s, c_p, b_p, m_p, c_s, b_s, m_s, c_g, b_g, v_g$
    return $predicates$
end procedure

The preprocessing module is mainly using PCL written in C++, while the detection method uses methods from the SciPy library stack, which are only accessible by Python. The SciPy ecosystem [Vir+20] is a collection of Python-based software packages for scientific computing. SciPy includes many of the most popular Python software packages including NumPy, matplotlib, pandas or Jupyter. NumPy [Har+20] is used especially in many scientific applications because it offers fast n-dimensional array structures and a multitude of functions on them. Additionally, SciPy features so-called *scikits*(SciPy Toolkits) that are standalone libraries offering additional functionality in a specific area. For instance, the scikit-learn [Ped+11] and scikit-image [Van+14] libraries are used in the gap detector to perform clustering and thresholding to the point cloud.

Preprocessing module

The preprocessing module implements the first two parts of the approach: the passthrough filter and the denoising. Since the gap detector is running in the ROS ecosystem, the preprocessing module first subscribes to the topic of the Nerian stereo vision system. Subscribing is done by assigning a callback method to the topic. This callback method will be called each time a point cloud is sent through the topic. The callback method first applies a passthrough filter from PCL to the cloud. It filters the point cloud on each axis based on a lower and an upper limit that are given by the user from the GUI. The filtered point cloud is then denoised by using Statistical Outlier Removal, resulting in the desired preprocessed point cloud. The intermediate filtered cloud and the result cloud are published as topics *filtered_cloud* and *denoised_cloud*. Publishing the output to topics allows other ROS modules to subscribe to these clouds. In our application, this is useful to visualize the results of the ROI (Region Of Interest) filtering and denoising by using the RVIZ module, which can subscribe to a point cloud and show them in a three-dimensional viewing tool. It also allows the detection module to subscribe to the *denoised_cloud* topic and find the gaps of the device. Besides the main callback, the module implements the dynamic reconfigure module with a separate callback. Dynamic reconfigure allows to reconfigure parameters of the detector from a GUI without recompilation, as illustrated in Figure 7.16 of Appendix 7.1. This makes

the process of selecting the ROI easier because the user can adjust the filter limits directly through the GUI with visual feedback from RVIZ.

Detection module

The main function of the detection module is to identify the gaps in a given point cloud. It subscribes to the denoised point cloud topic and processes the point cloud by means of automatic thresholding, clustering and convex hull calculations. The SciPy libraries offer efficient implementations of these methods. Thus, this node is written in Python. Similar to the preprocessing module, the detection module has a callback method that is called every time new data arrives from the preprocessing module through the *denoised_cloud* topic. The first step is to find an automatic threshold using the scikit-image filter module [Ped+11]. Given the thresholded point cloud, clustering is implemented by the methods in scikit-learn [Van+14] and the HDBSCAN module. The results of both thresholding and clustering are published as topics *thresholded_cloud* and *clustered_cloud* to RVIZ.

Following the volume correction approach, a convex hull is computed by using the SciPy method wrapper of the Qhull library [BDH96]. The callback method ends with applying the volume filter and finally creating visualization markers of the identified gaps. These markers are used by RVIZ to display the outlines in the three-dimensional viewer.

In addition to the main callback, the module implements a second callback for a gap detection service, as well as a dynamic reconfigure callback. The gap detection service can answer requests with information about the found gaps such as the gap center coordinates, vertices of the boundaries, or the volume of the gaps. This allows other applications to query gap information, which can be used by a robot to manipulate the device or fused with other information.

5.10.2 Component Segmentation

Segmentation of components is carried out by a standalone Python 3.x-based module that is externally called by the main ROS loop using Python's `os.system()` function, which runs a console command in the background. It facilitates the execution of a standalone Python program by starting a new process, allowing parallel execution. While the ROS main loop

awaits for the completion signal, the standalone Python program conducts the segmentation by loading the trained weights onto GPU and finally delivers the predictions. Acquired component information is then written to a .JSON file to be read by the planner and the association engine, as illustrated in Figure 3.3.

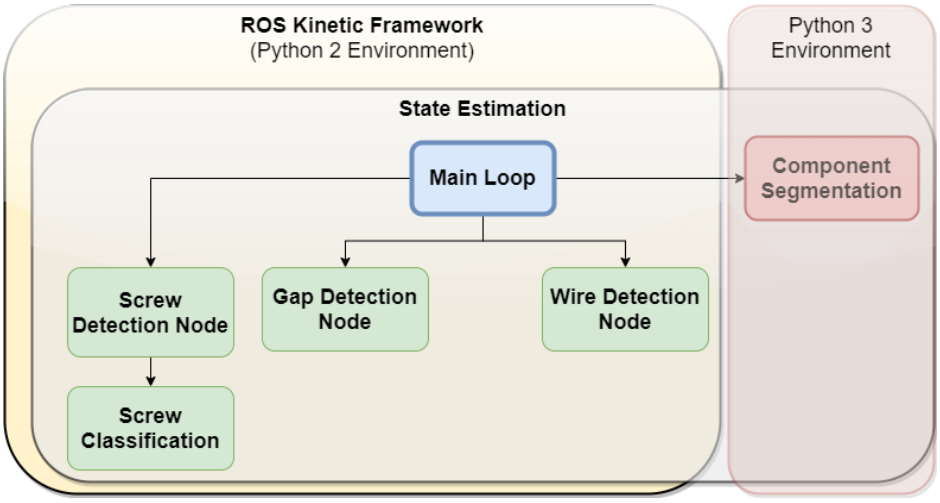

Figure 5.40: Component segmentation is a standalone Python 3 module, unlike other vision blocks that are nodes on ROS framework.

The reason segmentation capability was programmed as a standalone Python module instead of a ROS node was due to the incompatibility[56] of Python 3.x with the ROS version *Kinetic* (the latest available by the time this project started), and the technical decisions were already taken. Given that a possible ROS version is compatible with Python 3.x, then this module could also be wrapped as a ROS node. Figure 5.40 illustrates a graphical representation of ROS nodes and the standalone segmentation module.

56 Foote, T., Conley, K. (n.d.). Target Platforms. Retrieved February 23, 2021, from https://www.ros.org/reps/rep-0003.html#kinetic-kame-may-2016-may-2021

5.10.3 Screw Detection Node

A screw detection node is implemented in a way that it detects and classifies the screws in the scene. More importantly, the node launches with the main loop, and continuously outputs the predictions. That said, the trained weights are loaded only once on the main loop execution, and sufficient GPU memory is reserved until the main loop stops. Additionally, relevant ROS services are available for other components to request specific information regarding the screws.

The reason the screw detection node always reserves a specific amount of GPU memory is because, unlike the component segmentation, screw detection is conducted on small cropped images due to the inherent Hough circle finder. Therefore, the features are never sought in a 4K image, rather approximate sizes of 55×55.

The node also launches a relevant dynamic reconfiguration panel, allowing the user to adjust the parameters of the screw detector and classifier. This panel is convenient on the first run of the system, since the changes in the environment (such as illumination) may require the user to adjust the parameters of Hough circle finder (e.g., the lower and upper limits of the inner thresholding function). Besides, depending on the disassembly strategy, one could lower the confidence threshold to obtain more predictions with the cost of increased false positives, or vice versa.

5.10.4 Wire Detection Node

A wire detection node segments the detected wire pixels, groups them into structures called *wire segments*, and directs them further to predicate creation. It is important to underline the fact the segmentation is not an instance-aware one. It is not possible to obtain a specific instance of a wire, since often there is great amount of discontinuity among the wire pixels due to occlusions, tangles, and simply the wires entering and leaving the region of interest. Similarly, different wires could be detected as one segment, as shown in Figure 5.41. This situation, however, does not cause any ambiguity to any module of the system. Since the relevant action on wires is *cutting*, it does not matter if the wire belongs to a specific instance or not, it could be cut nevertheless.

Figure 5.41: Mask of a detected wire segment made of two intersecting wire instances, yet perceived as a single wire segment due to continuity of neighboring wire pixels.

5.10.5 Underlying Framework and Communication

As previously mentioned, ROS Kinetic is preferred to handle inter-component communication through its services, publishers and subscribers. As soon as the IMAGINE system boots, all nodes start according to an agreed priority. State estimation loop starts with a service call from the IMAGINE system, loading the necessary weights of its modules, occupying enough GPU memory and conducting its checks (i.e. HDD Table's angle). Afterwards, the algorithm described in Section 5.10 is followed through.

6 Experimental Evaluation

6.1 Evaluation Strategy

Evaluation of the proposed visual intelligence scheme is done by assessing the overall capabilities of the system on the estimation of the state. These capabilities are mainly the detection, localization, and identification of the visible entities and their types. Ideally, the scheme should also be evaluated on a second EOL product, so that its capabilities are proven to be robust and generalizing enough for an industrial use. For this purpose, another computer piece -GPU- was chosen. In total, 8 GPUs from various brands and models were collected. This chapter, therefore, documents the results of the benchmarking conducted on the HDDs (which the visual intelligence is trained for) and GPUs (which the visual intelligence has never seen). Additionally, to understand how well the system performs with its capabilities in the presence of limited training data, the system was also evaluated with the user annotated data of entities of the collected GPUs. Collected results are shared in Tables 6.3 , 6.4 and 6.5 below.

6.2 System Evaluation

Table 6.6 presents the results of the series of experiments E1, E2 and E3. Experiment E1 denotes that the acquired accuracy involves no re-training, meaning that the visual intelligence has no awareness of the second device (GPU). In these experiments, there was no annotated training data (e.g., GPU screws, GPU components, GPU wires) provided to retrain the screw detection, screw classification, component segmentation and wire detection networks. They were optimally trained to perform their tasks by default. However, they were trained with the default data that mostly consists of HDDs. It must be mentioned that the biggest entity on a GPU is its PCB. Additionally, entities such as bay, fan, sockets, screws and optionally wires exist in various colors and types. For instance, PCBs found in GPUs

Experiment	Data	Re-training	Test Device
E1	E.D.	No	HDD, GPU
E2	E.D. + 50% C.D.	Yes	HDD, GPU
E3	E.D. + 100% C.D.	Yes	HDD, GPU

Table 6.1: Evaluation scheme to be used through the experiments. "E.D" refers to existing data, whereas "C.D." refers to collected data.

vary in colors of black, blue and green, whereas for HDDs they are green by a very large margin. Nevertheless, by conducting E1, the ability of generalization is evaluated, and, thus, the question of "To what extend could the proposed scheme generalize, given absolutely no new training data?" is answered. Experiment E2 and E3 on the other hand, evaluate the scheme's capabilities after retraining it with 50% and 100% training data, respectively. By conducting E2 and E3, the question of "How does retraining with limited data affect the performance of the scheme on the second device?" is answered. Note that the gap detector was not evaluated on GPUs in any experiment, since there are no gaps significant to the disassembly of the device.

Tables shared in this chapter use acronyms for the names and accuracy of the modules to save space in presentation. Therefore, acronyms **S.D.A**, **S.C.A**, **C.S.A**, **W.D.A**, **G.D.A** correspond to *Screw Detection Accuracy*, *Screw Classification Accuracy*, *Component Segmentation Accuracy*, *Wire Detection Accuracy* and *Gap Detection Accuracy*, respectively.

Table 6.2 shows the experimental data in numbers. For every module, the data was split into training, validation and test sets in ratios of 70%, 20% and 10%, respectively. Evaluation scheme is illustrated in Table 6.1. None of the training strategies is subject to change (e.g., early stopping). Weights were reset between E1, E2 and E3 experiments to prevent learning of repetitive features and introducing bias.

6.2.1 Experiment E1

Experiment E1 is conducted on each module, testing each one's capabilities on performing visual tasks on raw HDD and GPU images without re-training. Table 6.3 reports the accuracy per module. For screws, the

Module\Data	Existing Data (E.D.) RGB Images	Collected Data (C.D.) RGB Images
Component Segmentation	600	100
Screw Detection	20000	2000
Screw Classification	20000	2000
Wire Detection	100	100

Table 6.2: Existing data consists of RGB images of HDD images, whereas collected data consists of RGB images of GPU images. Since the modules were already trained optimally with the existing data, the collected data used was intentionally kept limited.

description of the entity is largely the same (e.g., circular shape, feature in the center) therefore there is no drastic significant drop in accuracy of screw detection network, scoring 0.91. There is an insignificant drop from the original accuracy 0.99 [YW19] due to the fact that GPUs have black and dark gray screws which the network misses from time to time.

As for the screw classifier, the weighted average was found to be 0.94. Here as well, an insignificant drop was observed due to the aforementioned reason. Nevertheless, it was able to find what it was trained on when the learned color was present. Figure 6.1 illustrates such an example. The image above contains screws that have the ordinary silver-metallic color. Note that the screws in/on HDDs were of this color. Thus, the network has the learned features from the images of these HDDs. On the image below, however, it is noticeable that the detection network misses more. Since all the screws found on that particular GPU were of dark gray or black color, the accuracy is naturally lower. However, even in this case, the classifier nevertheless correctly identified the found screws as "ph1" and "torx6 as illustrated. It must be remembered that the classifier only classifies once the detector detects an instance. If there were only two classifications, it is due to the fact that there were only two screws detected.

Component segmentation requires specific user annotation and identification of the components of the device, which was only done for HDDs so far. E1 evaluated the segmentation module therefore, on an unseen device of GPU, and found out that the proposed scheme's segmentation module reports 0.71 Mean-F score, which a bit lower than 0.78 (the Mean-F score calculated for the original network). Figure 6.2 illustrates a case where

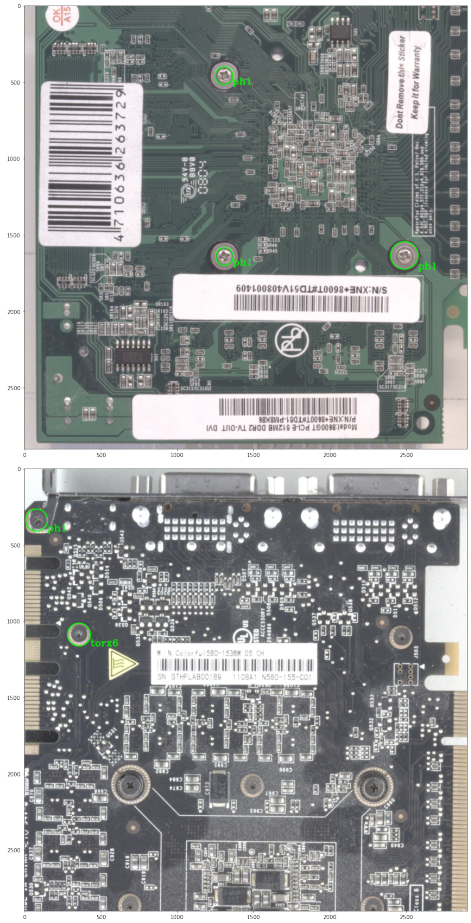

Figure 6.1: Correctly detected and classified screws when the learned color is present (above), missed and incorrectly classified screws when the learned color is different (below).

metallic bay partially occludes the PCB, thus the network is misidentifying PCB as bay as well. This is due to fact that the PCBs that were in the HDD image dataset had nothing on them, contrary to the GPU, where it is very likely to be a metallic bay and/or cooling unit over the PCB, and making PCB features less dominant. Similarly, in the same figure, the lower image shows a correctly identified and segmented PCB. Since most

of the PCB was visible. The network was able to associate it with the learned PCB features.

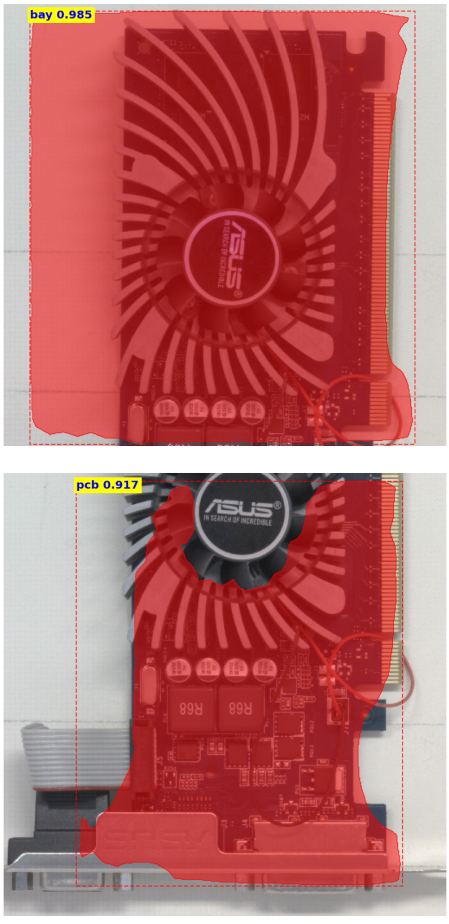

Figure 6.2: Predicted mask by the component segmentation performed on a GPU. Above image depicts an incorrectly identified component, whereas the below image depicts a correctly identified one. Portion of PCB pixels play a pivotal role in segmentation of the PCB.

57 Screw Detection Accuracy
58 Screw Classification Accuracy

Experiment	S.D.A.[57]	S.C.A.[58]	C.S.A.[59]	W.D.A.[60]	G.D.A.[61]
E1	0.91	0.94	0.71	(0.89, 0.88)	X

Table 6.3: Accuracy of the modules without any retraining. The networks only knew the learned features from the HDD training data.

The wire detection network was found to be the critical one here. Although it performs remarkably well on the GPU wires as illustrated in Table 6.3, features that resemble wires are also detected as wires. Since the context information is not there, the wire detector considers non-wire pixels as wires, as illustrated in Figure 6.3, where cooling pipes of the GPU are considered as wires.

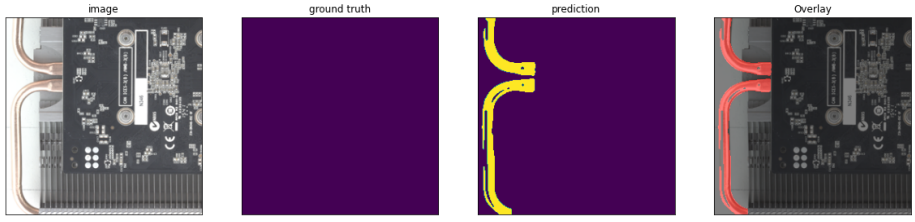

Figure 6.3: Predicted mask by the wire detection, incorrectly marking wire-like looking objects as wires. Metallic pipes are one example of such a situation.

6.2.2 Experiment E2

Experiment E2 is conducted on each module, testing each one's capabilities on performing visual tasks on raw HDD and GPU images with re-training. The training data used is set to 50% of the entire GPU training data (in addition to the existing HDD data). Table 6.4 reports the accuracy per module.

For screws, 50 new cropped images of screw heads belonging to GPUs were included in the dataset. The screw detector and screw classifier scores peaked, ensuring an accurate detection and classification. Therefore, it is

59 Component Segmentation Accuracy
60 Wire Detection Accuracy
61 Gap Detection Accuracy

concluded that 50 new images for screw detection and screw classification networks are sufficient for the GPU. There is a different case for the component segmentation network. This one started to predict meaningful masks for the PCB, as it was trained with extra 50 images of annotated GPU components (bay, PCB). Figure 6.4 shows a sample output where correctly identified PCB borders are following the correct edges of the PCB component. Note that the network is trying to avoid predicting on the irrelevant or unexpected pixels that correspond to the white plastic attachment found on the PCB. The module reports 0.78 Mean-F score, which is equal to the original 0.78 but higher than the E1 score of it, 0.71. Note that the original network was trained on 600 HDD images. Therefore, newly introduced 50 GPU images do make a difference in terms of generalizing.

Experiment	S.D.A[62]	S.C.A[63]	C.S.A.[64]	W.D.A.[65]	G.D.A.[66]
E2	0.99	1.0	0.78	(0.91, 0.93)	X

Table 6.4: Accuracy of the modules after training the networks with 50% of GPU training data.

Additionally, it was observed that the wire detection network accuracy changes positively on insignificant levels.

6.2.3 Experiment E3

Experiment E3 tests each module for its capabilities on performing visual tasks on raw HDD and GPU images with re-training. The training data used is set to 100% of the entire training data (in addition to the existing HDD data): 100 new annotated GPU images, plus 1000 screw images (cropped from these GPU images). Table 6.5 reports the accuracy per module. After re-training the networks, it was noted that there is a substantial improvement for the component segmentation module, where the network was observably learning the features encountered in GPUs

62 Screw Detection Accuracy
63 Screw Classification Accuracy
64 Component Segmentation Accuracy
65 Wire Detection Accuracy
66 Gap Detection Accuracy

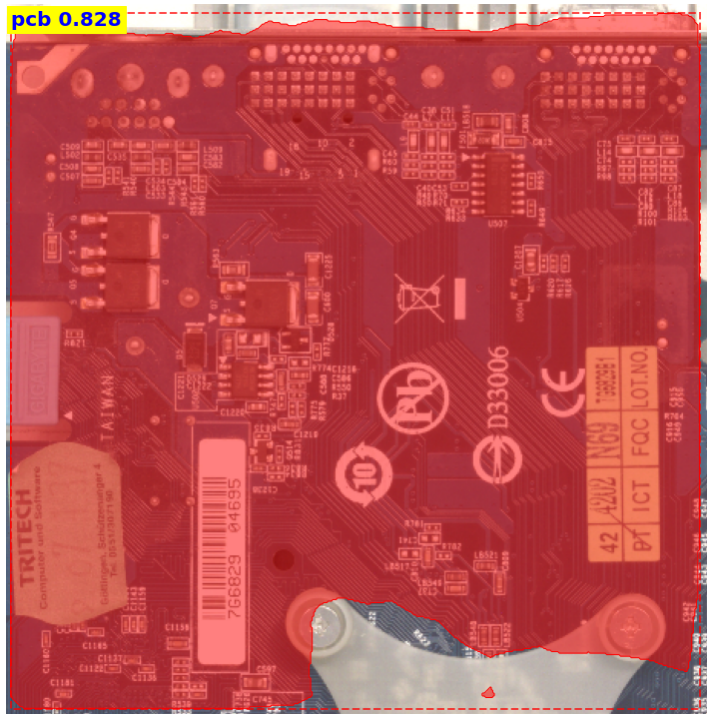

Figure 6.4: Predicted mask by the component segmentation performed on a GPU. The model has been trained with 50% of GPU training data.

and showing the ability to generalize. Figure 6.5 depicts an example where the entire PCB was correctly identified (with a prediction score of 0.846) and segmented accordingly.

Similarly to E2, highly accurate screw detection and classification abilities were observed. Features that were learnt enough for the network to capture the screws. It must be remembered that the mentioned data augmentation functions in the screw classification module generates synthetic data out of limited images and fills in the gaps in data. Therefore, it is observed that re-training the screw detection and classification networks with a small number of images is quite possible.

67 Screw Detection Accuracy
68 Screw Classification Accuracy
69 Component Segmentation Accuracy

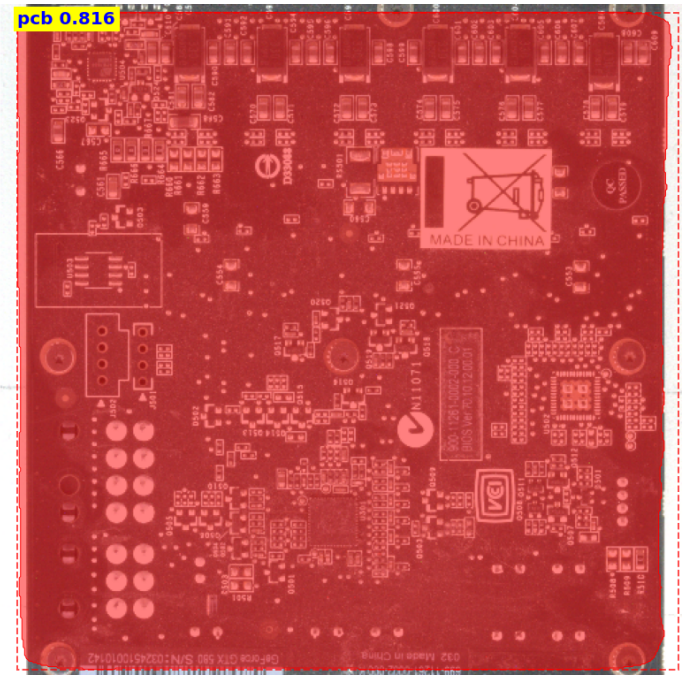

Figure 6.5: Predicted mask by the component segmentation performed on a GPU. The model has been trained with 100% of GPU training data.

Experiment	S.D.A[67]	S.C.A[68]	C.S.A.[69]	W.D.A.[70]	G.D.A.[71]
E3	0.99	1.0	0.84	(0.92, 0.93)	X

Table 6.5: Accuracy of the modules after training the networks with 100% of GPU training data.

Wire detection network accuracy was almost the same with the previous experiment's, no change observed in behaviors either. This is due to the fact that not all GPU models had wires and thus, the newly introduced GPU images had either no wires, or wires that were very easy to detect as illustrated in Figure 6.6.

70 Wire Detection Accuracy
71 Gap Detection Accuracy

Module	Dataset (RGB Images)	Baseline Accuracy
Component Segmentation	600	0.78
Screw Detection	20000	0.99
Screw Classification	20000	0.97
Wire Detection	130	0.91, 0.97

6.3 Generalization

After conducting series of experiments to assess the generalization capability of the scheme, it can be concluded that the scheme generalises the learnt knowledge to an unseen device by acceptable margins. It was found out that visual commonalities (similar features) play a big role in generalization. Experiment E1 proved that PCB components in both GPU and HDD were mostly identified and segmented correctly, and drew the aforementioned conclusion. Some of the incorrect identifications and segments were there due to the fact that the PCBs were occluded by bays. Experiment E2 proved that introducing training data by 50% (on top of the existing data) definitely increases the accuracy of segmentation on PCBs, as illustrated in Figure 6.4. Screw related capabilities were remarkably improved even in E2, with less data. Experiment E3 showed that the component segmentation is the module that reacts to the training data most. This was associated with the fact that other modules have plenty of training images, whereas the original image dataset for the component segmentation consisted of around 600 annotated images. Therefore, introducing 100 new images does make a difference for the retraining. The fine prediction of the edge features were noted as shown in Figure 6.5, as well as more correct identification of components.

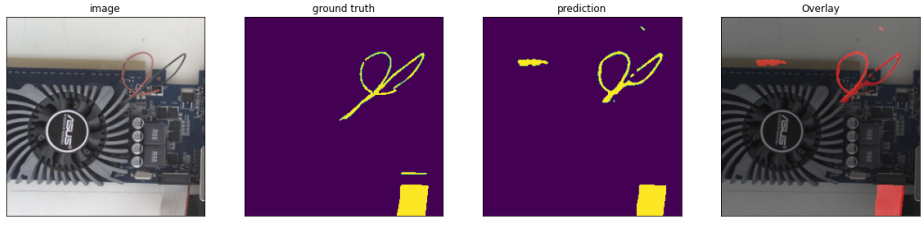

Figure 6.6: Wire Detection output during the experiment E3. All wires were correctly identified.

It is acknowledged that the improvement could only be observed for each module as shown in Table 6.6. Wire detection proved itself to be extremely robust, scoring high in E1, and obviously in E2 and E3. Gap detection had to be skipped for the experiments involving GPUs as there is no gap entity in this EOL device. It must be also noted that not all collected images were able to contain the entire view of the GPU, since the camera lens and the setup height were initially chosen for operating with HDDs. Therefore, the acquired results are the reported predictions on images that partially contain GPUs in their view, as illustrated in the referred figures. While this is not an issue, the optimal scheme would have to operate with a view that contains the chosen EOL device from a reasonable height, proportional to the dimensions of the device.

Experiments	S.D.A.[72]	S.C.A.[73]	C.S.A.[74]	W.D.A.[75]
E1	0.91	0.94	0.71	(0.89, 0.88)
E2	0.99	1.0	0.78	(0.91, 0.93)
E3	0.99	1.0	0.84	(0.92, 0.93)

Table 6.6: Accuracy of each module through experiments E1, E2, and E3.

72 Screw Detection Accuracy
73 Screw Classification Accuracy
74 Component Segmentation Accuracy
75 Wire Detection Accuracy

7 Conclusions & Outlook

This chapter summarizes the proposed system functionalities, discusses significant details, limitations and addresses any future work. It must be underlined that the conclusions made are specific to the state estimation package of IMAGINE, on which this thesis is written. The rest of the project modules and schemes are not taken into the account.

7.1 Discussion

This thesis proposed a visual intelligence scheme that could be used for automating the disassembly routines. Proof of concept was done by evaluating the proposed scheme on two EOL devices, namely hard drives and graphical processing units of computers. The proposed scheme's capabilities were also demonstrated as a part of state estimation package in IMAGINE project, which is a part of EU Horizon 2020 initiative. As of the time this thesis is written, the code has not been publicized. However, there have been published papers regarding a few modules (as mentioned in the thesis). These publications provided the publicized code of the individual components.

The proposed scheme was utilised in a setup with a monocular and stereo camera system, although technically, the only input data required is RGB-D data. In fact, the scheme was tested with an Intel Depth Camera D435 [76] (which provides RGB-D data) instead of the default 2-camera setup. It was noted that the quality of the point cloud acquired was found more promising than the one provided by the stereo camera (see Figure 7.1. However, due to the insufficient resolution of RGB images acquired (2400×2400), the component segmentation and screw detection modules worked with less accuracy. Since in many cases the entities are small and look similar to artefacts, having high resolution images to distinguish key

76 https://www.intelrealsense.com/depth-camera-d435/

features is essential. Therefore, the 2-camera setup is decided to be the default setup, where monocular camera provides 4K images.

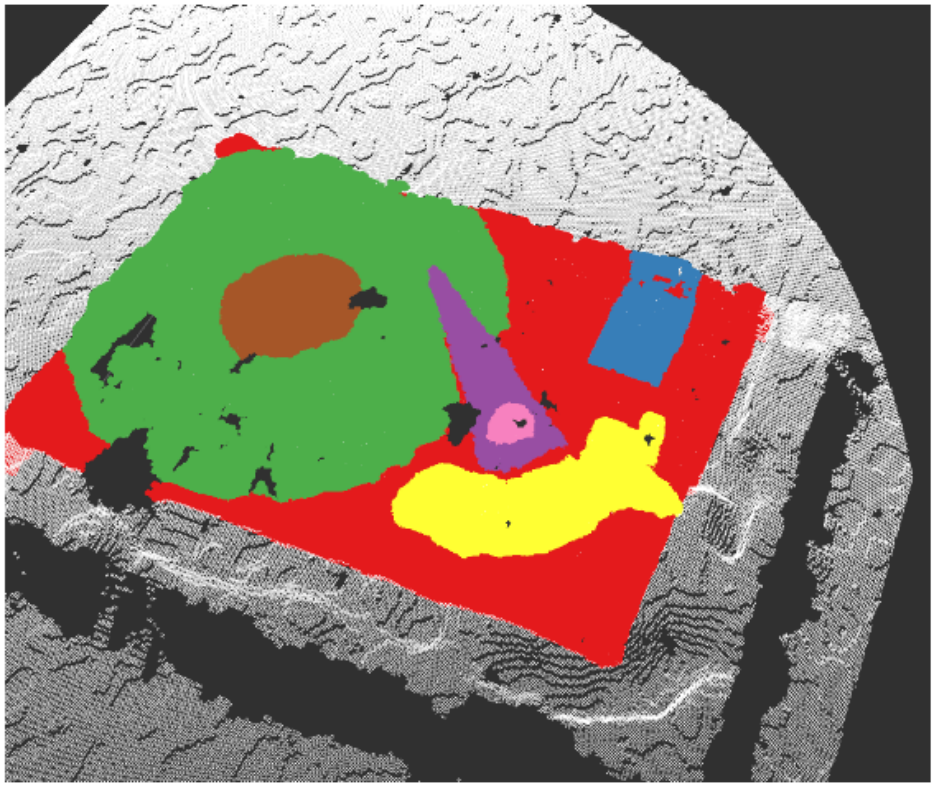

Figure 7.1: A point cloud of the hard drive acquired by the Intel Depth Camera D435. Due to a minor calibration error, the segmented masks are not precisely projected on the point cloud.

Another effort worth to mention here is the 3D model generation out of post-processed clouds. In the IMAGINE pipeline, the generated cloud of the HDD is passed to a 3D model generator, which outputs the 3D model constructed out of depth data (Figure 7.2). This model is later compared with the 3D models in a model database. If the model is found, it can be simulated. This has been tried using the Intel depth camera mentioned above, and the hard drive cloud was obtained. Then, using a 3D mesh generator, Voxblox [Ole+17], the 3D mesh was acquired. The complete

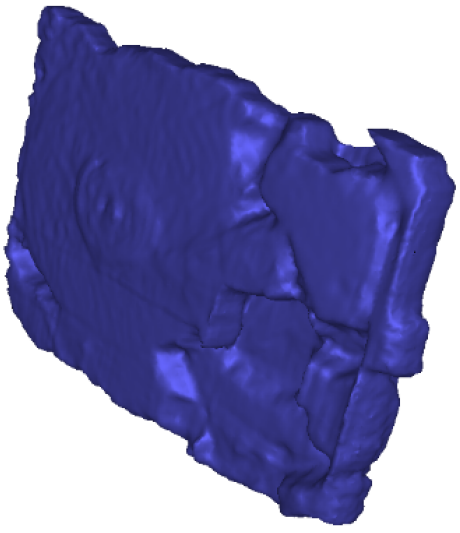

Figure 7.2: A mesh obtained by using Voxblox [Ole+17] converting clouds into 3D meshes.

pipeline for this effort can be found in 7.19 in Appendix 7.1. Even the small individual parts' models were acquired out of segmented clouds (as shown in Figure 7.3). However, the acquired point cloud only had the surface points. Thus, the non-visible parts of the HDD (bottom) were registered as void, as the cloud did not contain a surface point from there. This made the generated model impossible to use in any post-processing, since the 3D model comparison mechanism would not provide healthy results. It was noted that a good heuristic algorithm is required to perform 3D filling of the void parts. This is a separate topic studied under computer graphics.

Although this was a promising effort, having void data on the non-visible parts of the device makes it unusable in the IMAGINE pipeline. Instead, the collected point clouds are directly passed into an algorithm which is based on ICP (Iterative Closest Point) [Che+02], and involves heuristic filling. This effort was taken over by the partner university INSA (The Institut National des Sciences Appliquées de Renne), where Associate Professor Maud Marchal[77] continues her work with her team.

77 http://people.rennes.inria.fr/Maud.Marchal/Research.htm

Figure 7.3: A 3D mesh of a HDD magnet acquired from a point cloud. The process starts with component segmentation module that segments the magnet on a 2D plane. Later, the acquired point is used to project the found 2D pixels on the 3D point cloud obtained by the stereo camera. Lastly, using Voxblox [Ole+17], the point cloud is processed into the 3D mesh.

In conclusion, the proposed scheme is designed to complete the required objectives. To our knowledge, it is the first visual intelligence scheme that has the demonstrated capabilities for automated disassembly. Therefore, the novel contribution of this work is promising for recycling plants that are likely to use robotic systems. As of this writing, there is a prototype developed and demonstrated [78] as one of the milestones of the IMAGINE project. The State estimation package of the demonstrated prototype is engineered with the methods described in this thesis.

7.2 Limitations

It could be concluded that the proposed scheme's limitations largely stem from the absence of new annotated data. Since the initial proof of concept is done by training the system for HDDs (and for GPUs to a certain degree), the obvious limitation of the system is its inability to perform as expected on an unseen device that has no common components with the devices the system trained for. Another likely case is the confusion between two different parts of two different devices. Consider a magnet of a hard drive and a similar looking component in another device. If the system does not have any labeled data from the second device, it won't correctly identify its components. However, it was also found in this thesis that different

78 https://www.youtube.com/watch?v=m8aEZnSdiCA

devices with common components are definitely within the reach of a wide generalization capability.

Having the optimal lighting conditions for the sensor setup a technical limitation. Since a few major classical computer vision methods, such as Hough Transform, require tuning of certain parameters due to their threshold functions. For this reason, the proposed scheme should be provided with fixed and optimal lighting with correctly tuned parameters. If the setup changes (e.g., light source moves or changes) even slightly, the system is no longer guaranteed to perform as expected. Additionally, through evaluation, it was found that the accuracy of Hough Transform is below 0.8 by nature, even with the optimal parameters. Therefore, the screw detection and classification modules are limited to that accuracy. This is due to the fact that the networks are predicting on the region of interest provided by the Hough Transform method. While this task could also be taken over by a separate DCNN which detects circles, it should be kept in mind that Hough Transform requires no training data and works in real-time, unlike the DCNNs, which require annotated data (marked circles in the images) and are computationally more expansive than Hough Transform.

Another limitation is the fact that devices with shiny and/or reflective parts cause the employed gap detector to fail due to the noise in the point cloud. Since reflecting points cannot be represented correctly, the found gaps are either too large or too small. A practical way to deal with this problem is to apply transparent anti-reflection spray over the devices surface, which guarantees tolerably fine point clouds. A possible integration of spraying action into the robotic system could be a promising solution.

One significant point to mention is the fact that the proposed scheme demands high-end computers to fulfill its responsibilities. Since the chosen DCNNs were state-of-the-art at the time of this thesis, they could only be trained using the cloud services providing cutting-edge TPUs and GPUs. It could be agreed that the hardware required for the system makes it less desirable in an economical context.

Finally, the ROS implementation of the proposed scheme requires more than a minute to complete the scene analysis. This, could have been different if the system was not overloaded by the intercommunication between different packages of IMAGINE. Once allocated enough memory,

DCNNs are quite fast for static analysis processes such as this. The process that is computationally expansive is the process of communication between different modules of a complex system involving multiple regular computers over ROS interface. If tested standalone, the functionalities presented in the scheme work fast enough to be used in an industrial setup.

7.3 Future Work

In order to compensate the shortcoming of hardware, the proposed system could utilize cloud storage and computing services such as Google Drive and Google Colab. If a suitable GUI is developed to grant user the abilities of scene visualization, data collection and preparation, re-training of the system modules, then the system could indeed use cloud TPUs to train the modules. At this time, the IMAGINE project is doing so. Since it is only done as a proof of concept, the planned GUI (as illustrated in Figure 7.4) is thought to allow the user to crop and label screw images, prepare the training data and infer locally, whereas the user is assumed to be directed to Google Colab service for training and eventually automatic downloading of the new weights, through identification via a Google account.

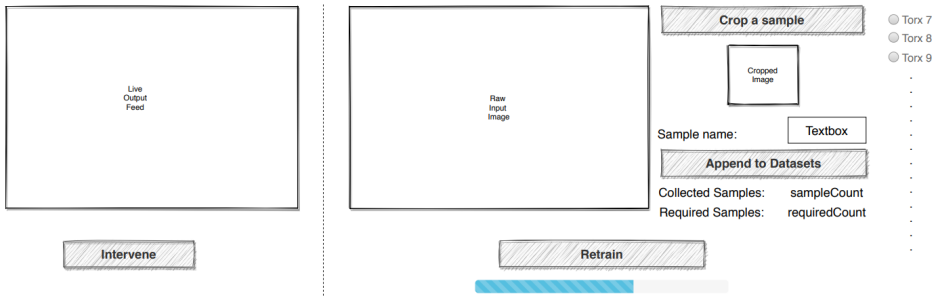

Figure 7.4: Sketch-up of the planned GUI to enable local and cloud functionalities.

The planned GUI allows the user to intervene in the presence of wrong detection or classification, when the user sees it on the visual display. It is also designed to import the new weights downloaded to the local computer. A preliminary flowchart of user's involvement upon this via the planned GUI for re-training is represented in Figure 7.5.

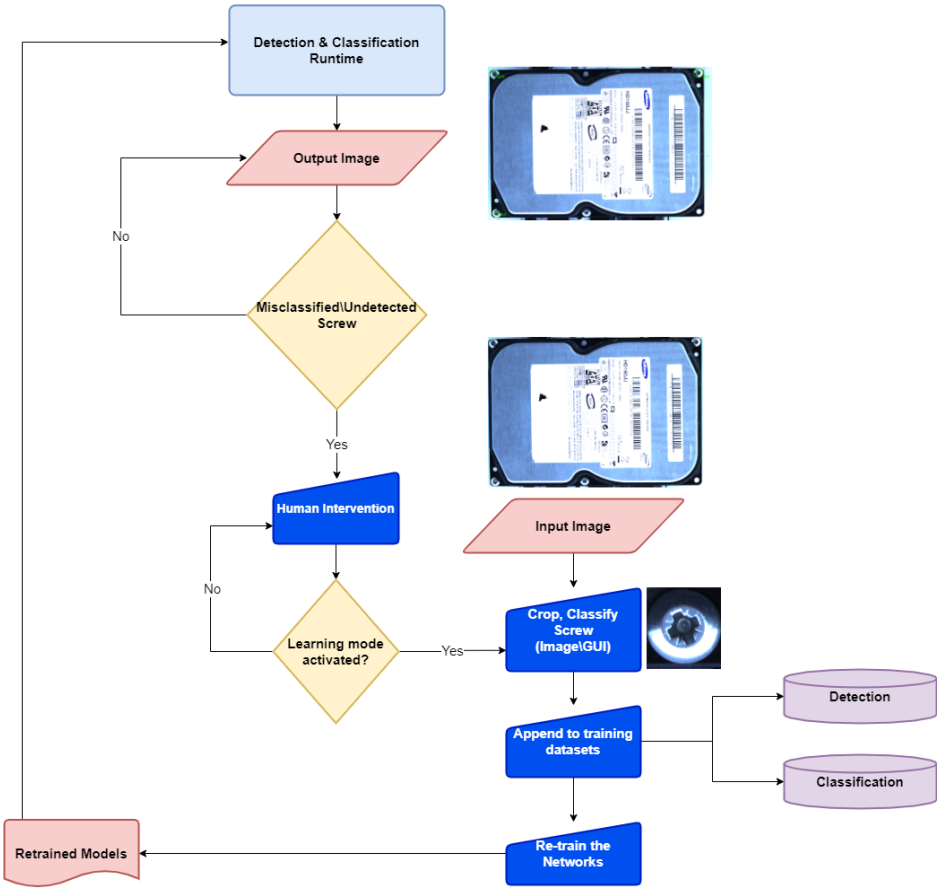

Figure 7.5: Flowchart of the planned re-training capability.

Apart from the learning ability, an interesting future work could be done on expanding the capabilities of the existing modules in the system. This research effort could also motivate the researchers to create image and cloud datasets for the modules to train on different devices. Doing so, an accumulated knowledge base could be used in development of newer methods.

Another topic to look into is the 6D pose estimation using deep learning. Although initially considered, the existing state-of-the-art 6D pose estimation networks are unable to provide a convenient replacement for

2D imagery with deep learning methods. There have been several publications [BV20; Su+21; Tre+18; Xia+18; XLC21; Zha+21] and an NVIDIA solution [Tre+18] to this problem. However, the proposed schemes were too complicated to be used in recycling plants by non-experts. 6D pose estimation networks demand a massive amount of annotated training data, which is an effort that requires an expert knowledge of computer graphics. The data generation process involves a manual synthetic data generation step, where the end user has to operate with tools such as NVIDIA Data Synthesizer [79].

This work also motivates the cooperation between the industry and academy to develop visual intelligence schemes for recycling plants. Every module described in this thesis could be conveniently re-factored to satisfy the requirements of similar systems. In this thesis, the partner recycling plant was Electrocycling GmbH [80] from Goslar, Germany.

79 https://github.com/NVIDIA/Dataset_Synthesizer
80 https://www.youtube.com/watch?v=23ZipmJq-n8

Appendix

7.1 Figures

The next set of figures are used in the chapters specified below.

- Figures 7.2, 7.1 are supplements to Chapter 1: Introduction.
- Figures 7.3, 7.4, 7.5, 7.6, 7.8 are supplements to Chapter 2: Related Work.
- Figures 7.17 is a supplement to Chapter 3: Overview.
- Figures 7.9, 7.10, 7.11 are supplements to Chapter 4: Background.
- Figures 7.12, 7.13, 7.14, 7.15, 7.16, 7.18 are supplements to Chapter 5: Approach.
- Figure 7.19 is a supplement to Chapter 7: Conclusion.

81 https://www.dailymetalprice.com/metalprices.php
82 https://www.dailymetalprice.com/metalprices.php
83 https://www.migatron.com/wp-content/uploads/2018/04/RPS-412A_Analog_Voltage_Rev_3.pdf
84 https://www.migatron.com/wp-content/uploads/2017/01/RPS-409A-IS_Data_Sheet_Rev_4.pdf
85 https://www.pfeifer-technology.de/skycnc-wm-serie-pfiffige-grossformat-bearbeitung/

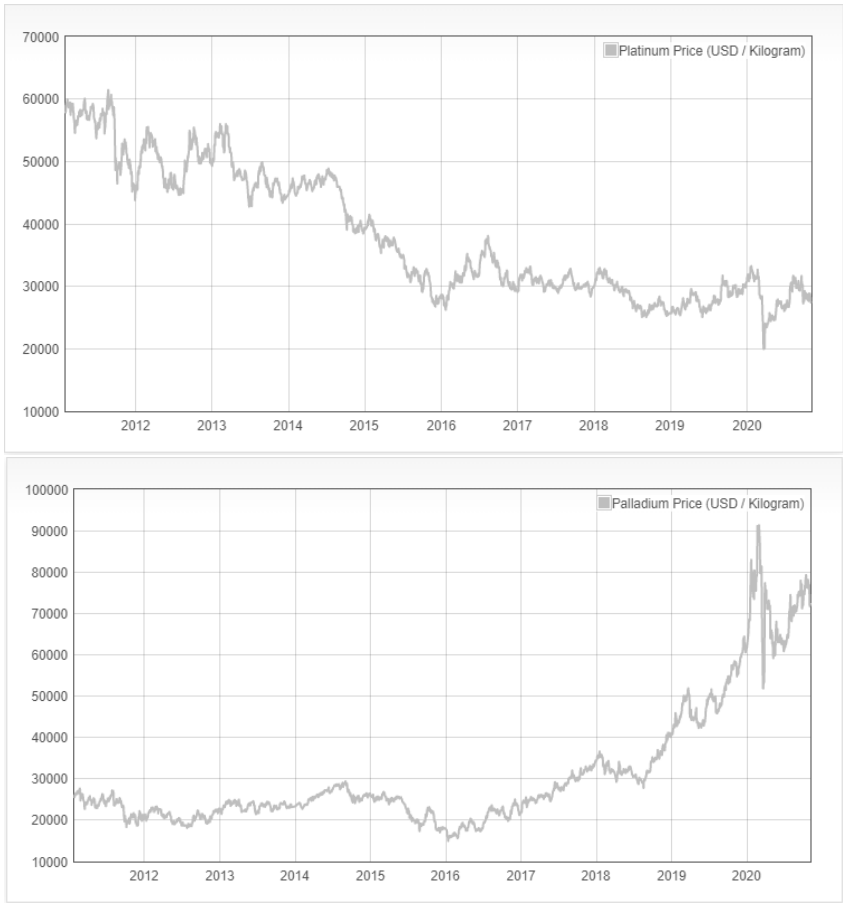

Figure 7.1: Price per Kilogram over the decade for platinum and palladium found in e-waste devices[81].

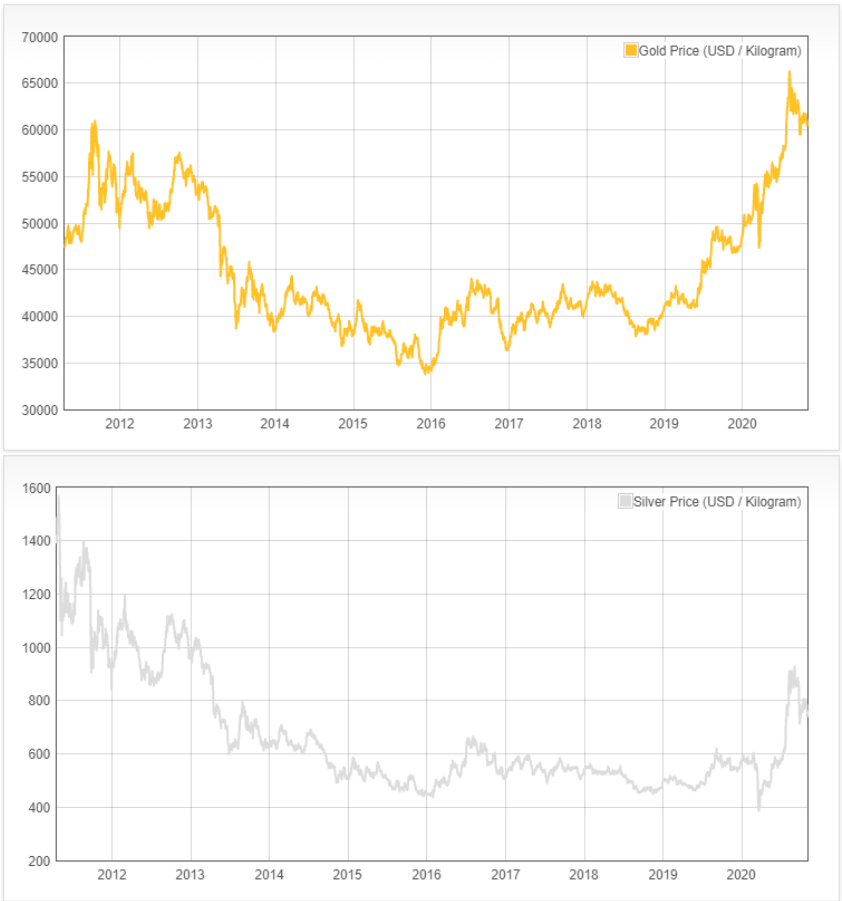

Figure 7.2: Price per Kilogram over the decade for gold and silver found in e-waste devices[82].

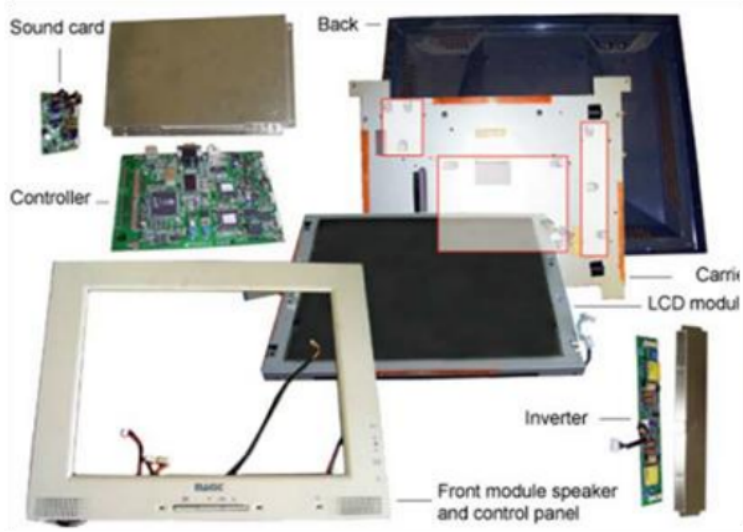

Figure 7.3: Modules in LCD screens[KKS09].

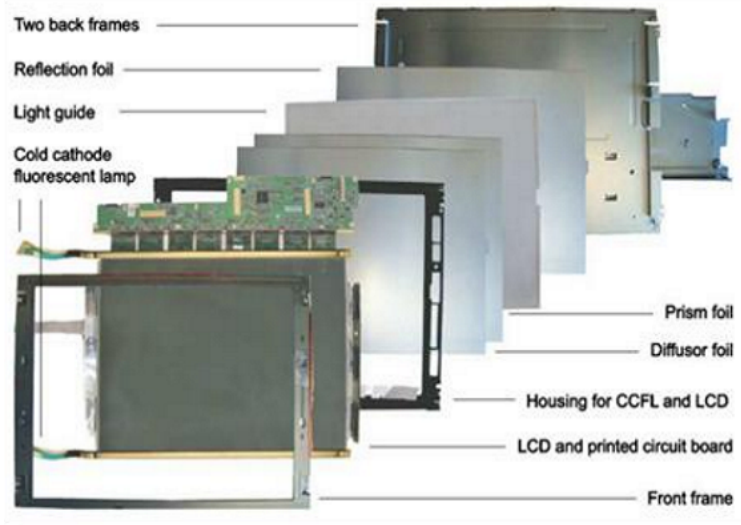

Figure 7.4: Components in LCD screens[KKS09].

Figure 7.5: RPS-412A high accuracy analog ultrasonic sensor. [83]

Figure 7.6: RPS-409A analog ultrasonic sensor. [84]

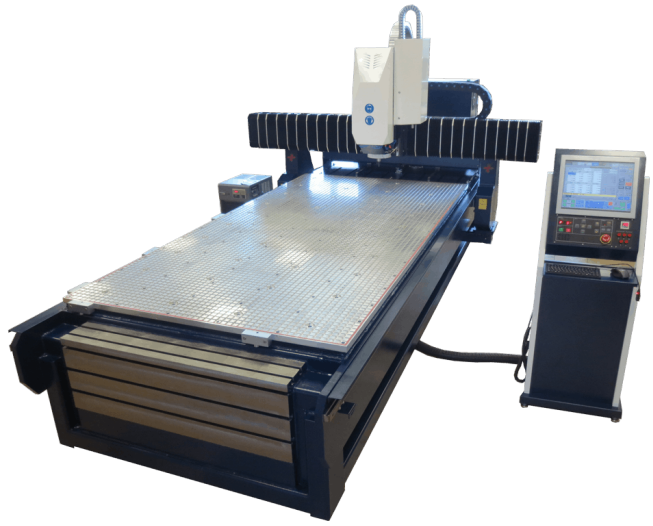

Figure 7.7: Sample CNC/milling machine [85]

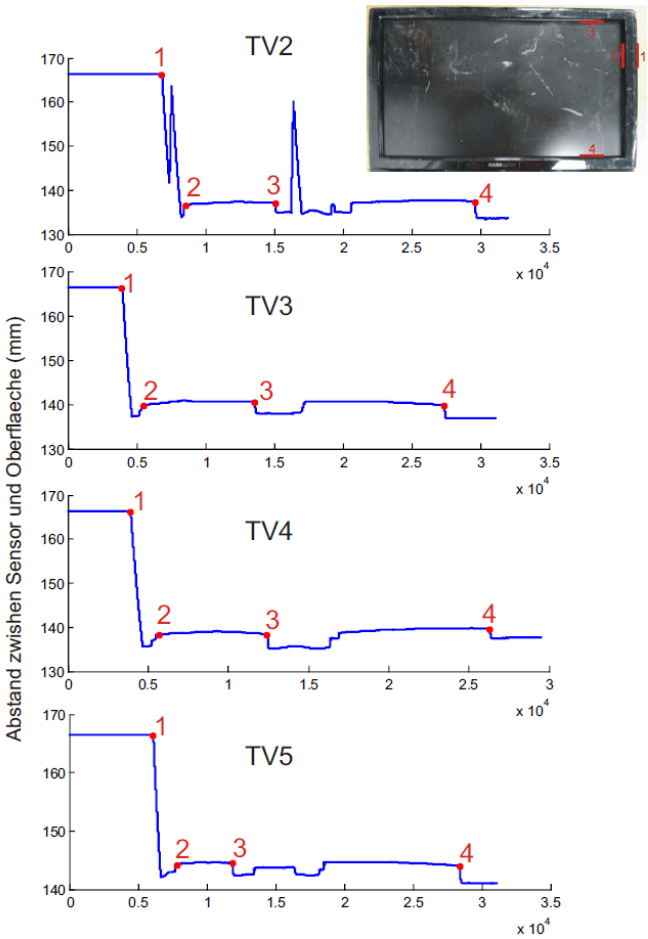

Figure 7.8: Scanning results obtained from the range sensor. Different TVs are considered, which are similar to the one shown in the figure.

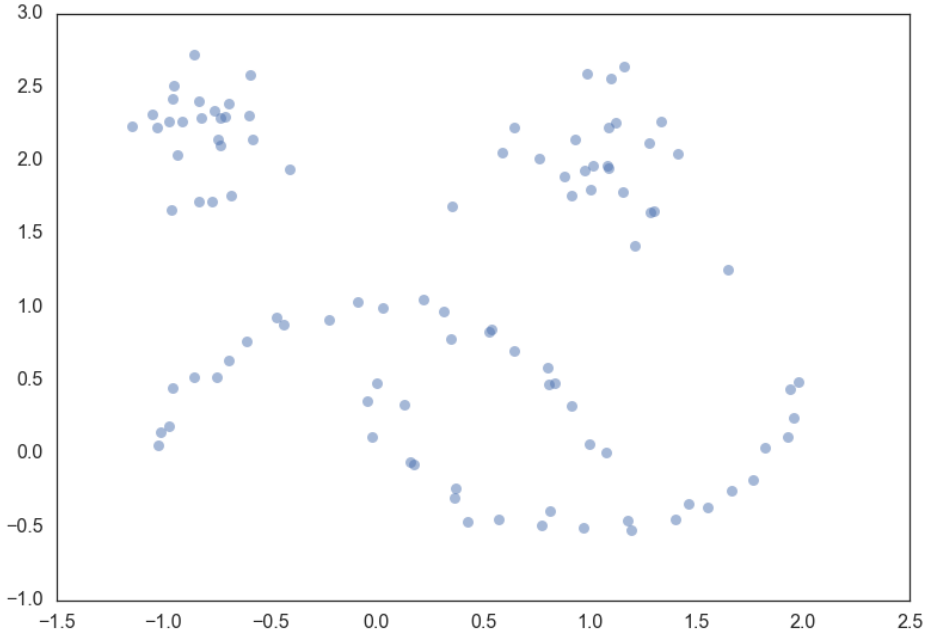

Figure 7.9: Input data points to be clustered.

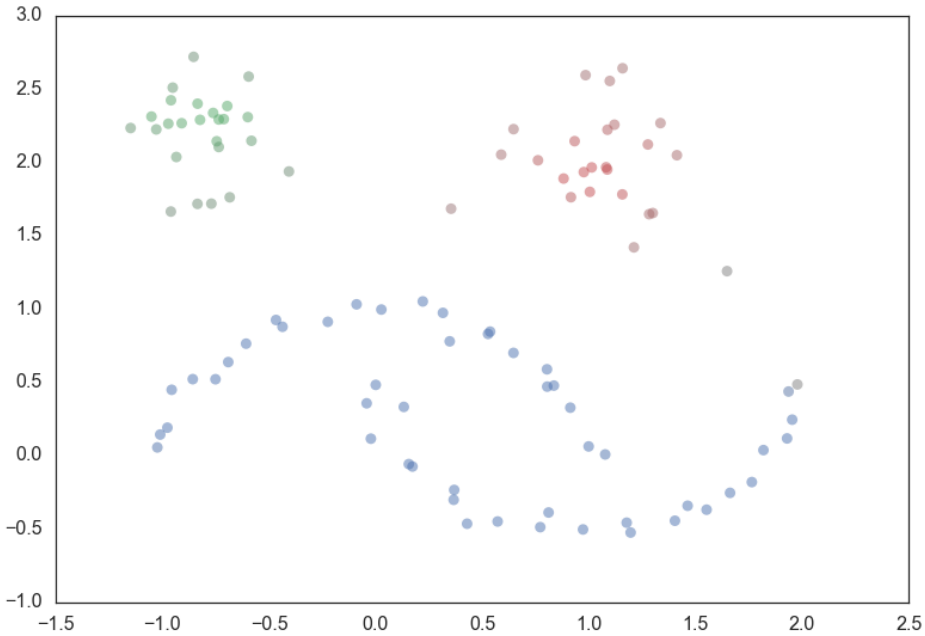

Figure 7.10: Data points clustered by HDBSCAN.

Camera Specifications

Camera modules:	Basler daA1600-60um
Sensor resolution:	1600 x 1200 pixels
Sensor:	e2v EV76C570
Sensor format:	1/1.8"
Lens mount:	C/CS-mount
Chroma:	mono / color
Shutter:	global shutter
Interface:	USB 3.0
Trigger-input:	4-pin Binder M8 connector
Stereo baseline distance:	10 cm / 25 cm
Mounting bottom side:	4× M3 threaded hole 1× 1/4" UNC threaded hole (tripod mount)
Mounting top side:	2× M3 threaded hole
Weight without lenses:	280 g for 10 cm baseline 450 g for 25 cm baseline
Conformity:	CE, FCC, RoHS

Figure 7.11: Specifications of the stereo camera used.

Image Sensor Metrics

Sensor width:	**7.15 mm** (800 pixels)
Sensor height:	**5.36 mm** (600 pixels)
Pixel pitch:	**17.87 µm** (after binning)
ROI width:	**7.15 mm** (400 pixels)
ROI height:	**5.36 mm** (300 pixels)

Lens Metrics

Focal length:	**9.07 mm**
Horizontal angle of view (ROI):	**43.0°**
Vertical angle of view (ROI):	**32.9°**
Diagonal angle of view (ROI):	**52.5°**

Stereo Geometry Metrics

Baseline distance:	**10.0 cm (3.94 in)**
Minimum depth:	**19.9 cm (7.8 in)**
ROI with full disparity range:	**145 × 300 pixels** (30.1° × 32.9°)
ROI with half disparity range:	**273 × 300 pixels** (36.7° × 32.9°)

Depth	Depth Error
0.2 m	0.02 cm
0.5 m	0.12 cm
1 m	0.50 cm
2 m	1.99 cm
5 m	12.6 cm
10 m	51.8 cm
20 m	219 cm
50 m	1634 cm
100 m	9709 cm

Depth	Depth Error
0.65 ft	0.01 in
1 ft	0.02 in
2 ft	0.07 in
5 ft	0.45 in
10 ft	1.83 in
20 ft	7.43 in
50 ft	48.7 in
100 ft	212 in

Figure 7.12: Lens and sensor metrics for the Karmin2.

Sensor

Sensor Vendor	ON Semiconductor
Sensor	MT9F002
Shutter	Rolling Shutter
Max. Image Circle	1/2.3"
Sensor Type	CMOS
Sensor Size	6.5 mm x 4.6 mm
Resolution (HxV)	4608 px x 3288 px
Resolution	14 MP
Pixel Size (H x V)	1.4 µm x 1.4 µm
Frame Rate	7 fps
Mono/Color	Color

Figure 7.13: Specifications of the monocular camera used.

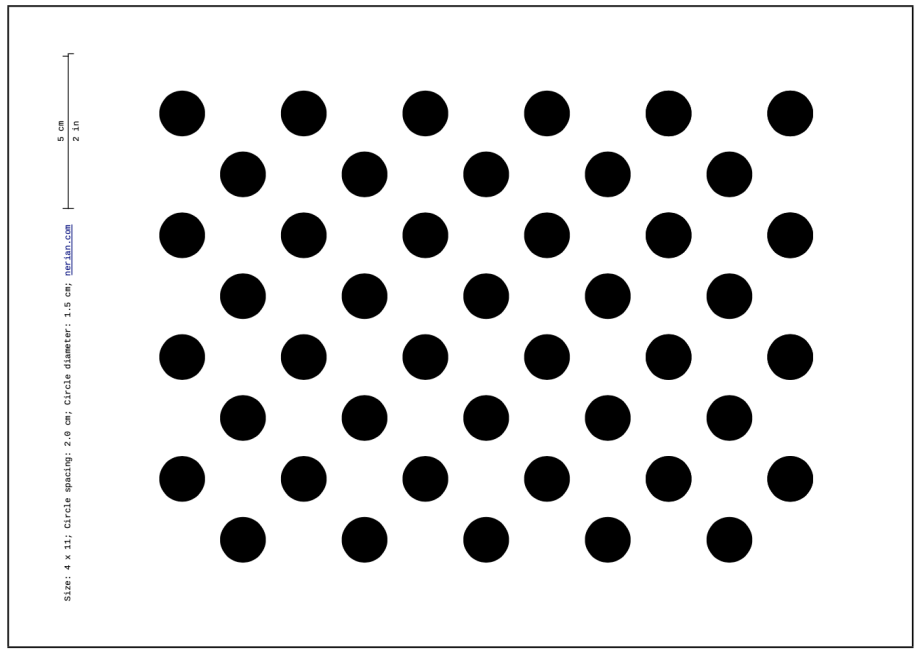

Figure 7.14: Required calibration pattern for the stereo camera calibration using the Nerian software.

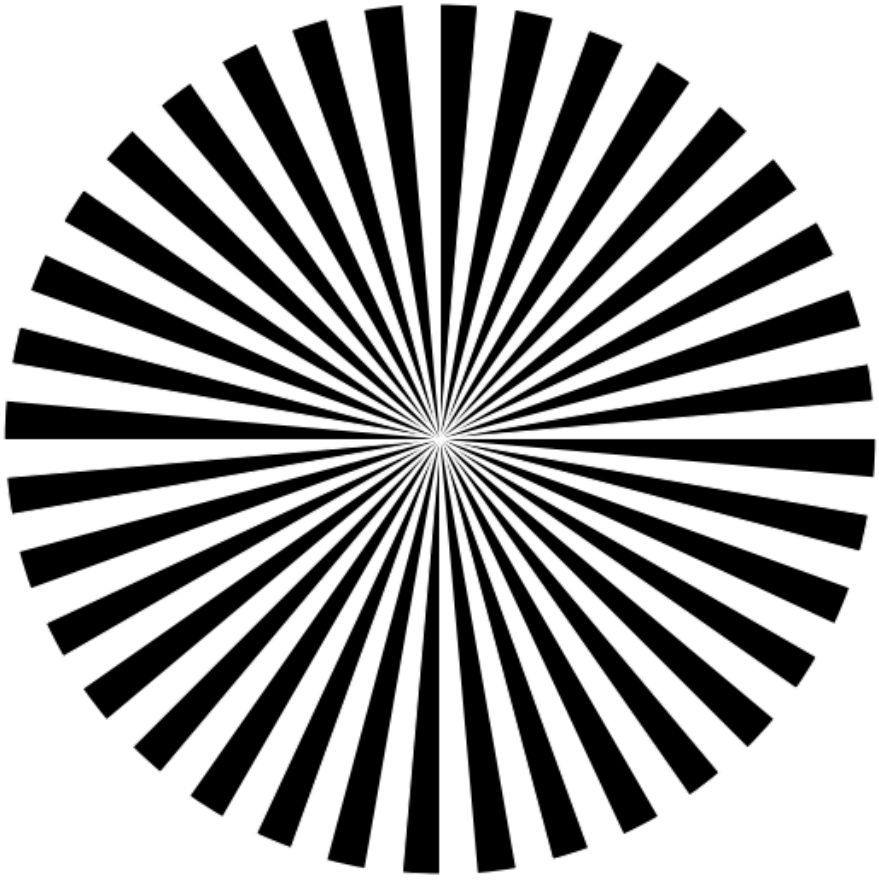

Figure 7.15: Siemens Star used to adjust the focus of Nerian camera system for intrinsic calibration.

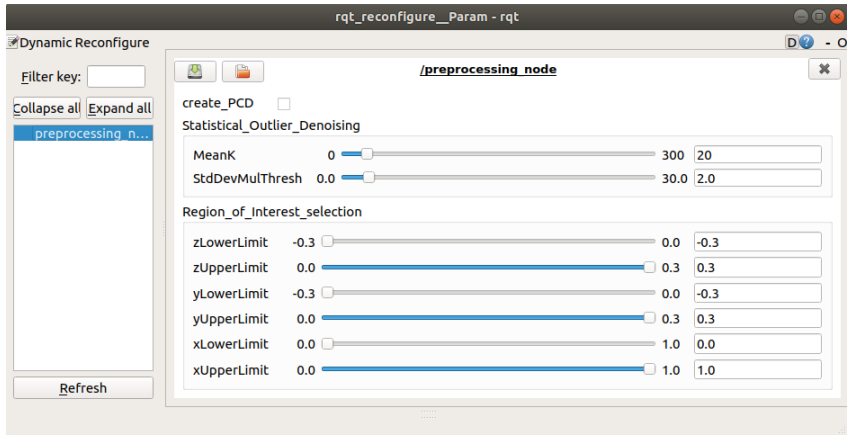

Figure 7.16: The dynamic reconfigure GUI of the preprocessing module

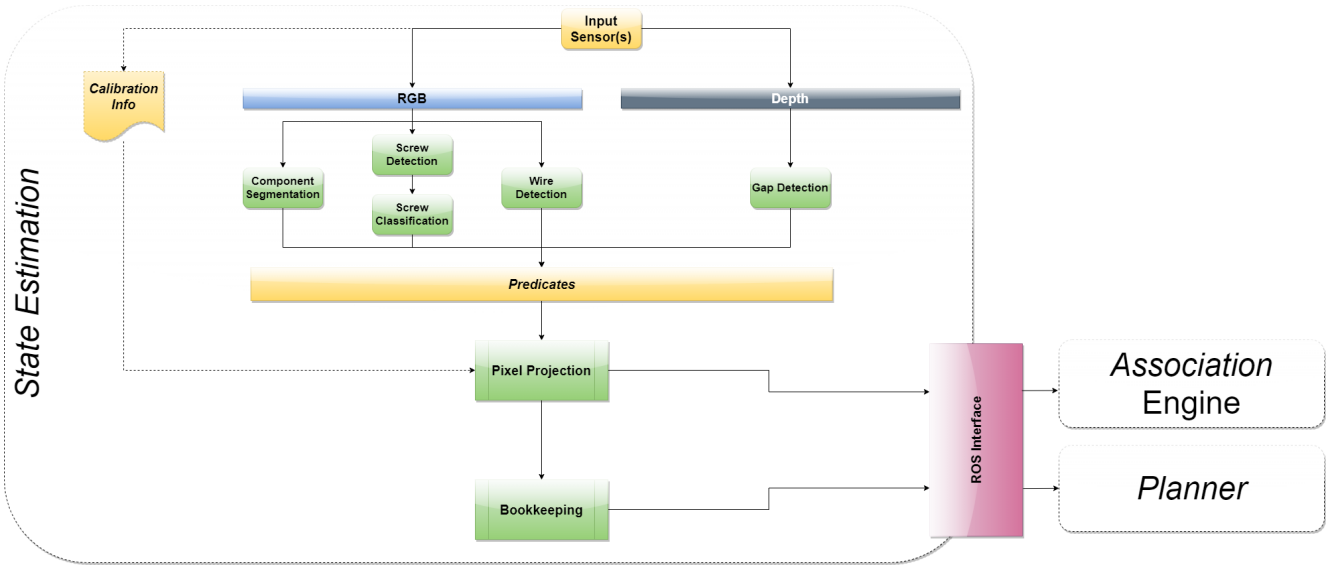

Figure 7.17: Proposed visual intelligence scheme for the State Estimation package of IMAGINE.

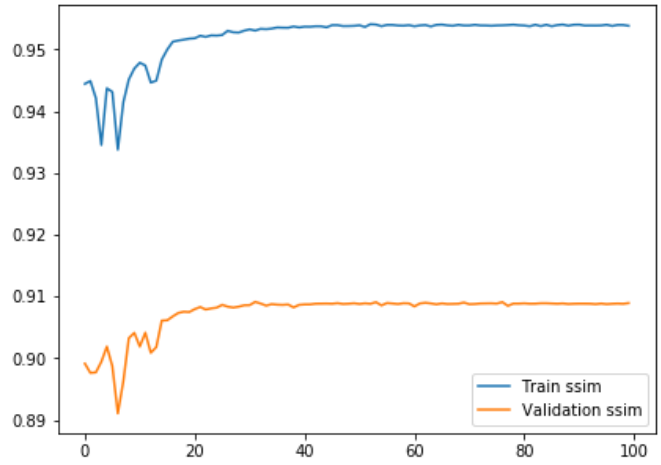

Figure 7.18: Change of SSIM through epochs with the pre-trained weights from the K-Gen dataset.

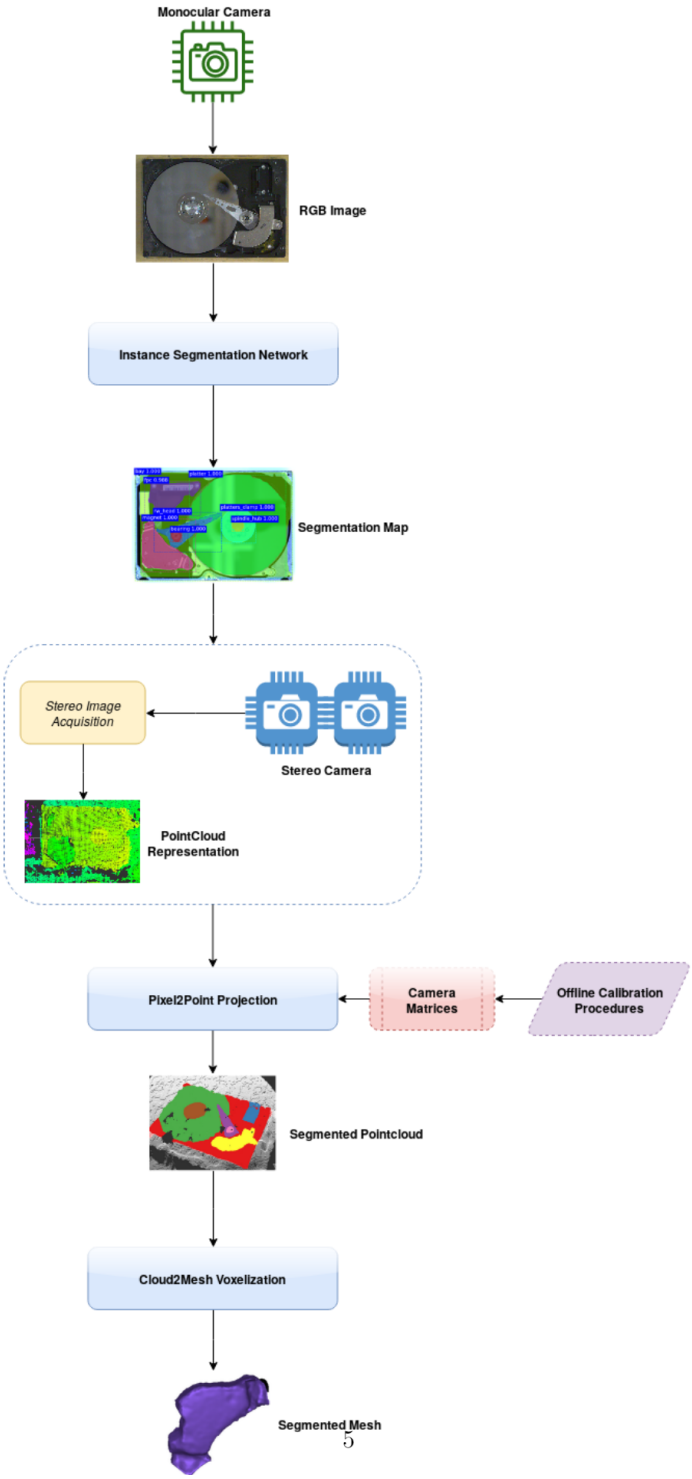

Figure 7.19: Pipeline used to create 3D meshes out of acquired point clouds.

Bibliography

[AA90] A Lynn Abbott and Narendra Ahuja. **Active surface reconstruction by integrating focus, vergence, stereo, and camera calibration**. In: *Proceedings Third International Conference on Computer Vision*. Citeseer. 1990, 489–490 (see page 62).

[Aba16] Martín Abadi. **TensorFlow: learning functions at scale**. In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*. 2016, 1–1 (see page 121).

[ABD16] Grigory Antipov, Sid-Ahmed Berrani, and Jean-Luc Dugelay. **Minimalistic CNN-based ensemble model for gender prediction from face images**. *Pattern recognition letters* 70 (2016), 59–65 (see page 105).

[AHF99] Moumen T Ahmed, ElSayed E Hemayed, and Aly A Farag. **Neurocalibration: a neural network that can tell camera calibration parameters**. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. IEEE. 1999, 463–468 (see page 64).

[Aph] Aphex3. *Typical cnn.png*. [CC BY-SA 4.0, from 16 December 2015, last checked: 16.05.2020, 15:00]. URL: https://commons.wikimedia.org/w/index.php?curid=45679374 (see page 79).

[AV06] David Arthur and Sergei Vassilvitskii. **k-means++: The advantages of careful seeding**. Tech. rep. Stanford, 2006 (see page 70).

[AWZ18] Thangarajah Akilan, Qingming Jonathan Wu, and Hui Zhang. **Effect of fusing features from multiple DCNN architectures in image classification**. *IET Image Processing* 12:7 (2018), 1102–1110 (see page 105).

[Bai02] M. Bailey-Van Kuren. **Automated demanufacturing studies in detecting and destroying, threaded connections for processing electronic waste**. In: *Conference Record 2002 IEEE International Symposium on Electronics and the Environment (Cat. No.02CH37273)*. May 2002, 295–298. DOI: 10.1109/ISEE.2002.1003283 (see page 21).

[Bai06] Michael Bailey-Van Kuren. **Flexible robotic demanufacturing using real time tool path generation**. *Robotics and Computer-Integrated Manufacturing* 22:1 (2006), 17–24 (see page 21).

[Bal+15] Cornelis P Balde, Feng Wang, Ruediger Kuehr, and Jaco Huisman. **The global e-waste monitor 2014: Quantities, flows and resources** (2015) (see page 15).

[BDH96] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. **The Quickhull algorithm for convex hulls**. *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE* 22:4 (1996), 469–483 (see page 154).

[BH96] U Buker and Georg Hartmann. **Knowledge-based view control of a neural 3-D object recognition system**. In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 4. IEEE. 1996, 24–29 (see page 20).

[Bis19] Ekaba Bisong. "Google Colaboratory." In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, 59–64 (see pages 85, 121, 131).

[BK00] Gary Bradski and Adrian Kaehler. **OpenCV**. *Dr. Dobb's journal of software tools* 3 (2000) (see pages 31, 61).

[BK08] Gary Bradski and Adrian Kaehler. **Learning OpenCV: Computer vision with the OpenCV library**. " O'Reilly Media, Inc.", 2008 (see page 104).

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. **Segnet: A deep convolutional encoder-decoder architecture for image segmentation**. *IEEE transactions on pattern analysis and machine intelligence* 39:12 (2017), 2481–2495 (see pages 25, 119).

[Bla+17] CP Blade, V Forti, V Gray, R Kuehr, and P Stegmann. **The Global e-waste Monitor 2017**. *United Nation University (UNU), International Telecommunication Union (ITU) and International Solid Waste Association (ISWA), Bonn/Geneva/Vienna* (2017) (see page 4).

[BPB07] Balakrishnan Ramesh Babu, Anand Kuber Parande, and Chiya Ahmed Basha. **Electrical and electronic waste: a global environmental problem**. *Waste Management & Research* 25:4 (2007), 307–318 (see page 15).

[BPP17] Patricio Benavente, Pavlos Protopapas, and Karim Pichara. **Automatic Survey-Invariant Classification of Variable Stars**. *The Astrophysical Journal* 845 (Aug. 2017), 147. DOI: 10.3847/1538-4357/aa7f2d (see page 71).

[Bri20] Tobias Brinker. *Gap Detection in Hard Drive Disassembly Processes*. Bachelor's Thesis. Georg-August-Universität Göttingen, III.Physics Institute, Biophysics Department, Mar. 2020 (see pages 51, 134, 136–150).

[BRP16] Mohamad Bdiwi, Aquib Rashid, and Matthias Putz. **Autonomous disassembly of electric vehicle motors based on robot cognition**. *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June:July (2016), 2500–2505. ISSN: 10504729. DOI: 10.1109/ICRA.2016.7487404 (see pages 21, 29, 101, 102).

[Buc+12] Matthias Buchert, Andreas Manhart, Daniel Bleher, and Detlef Pingel. **Recycling critical raw materials from waste electronic equipment**. *Freiburg: Öko-Institut eV* 49:0 (2012), 30–40 (see page 33).

[Bük+01] Ulrich Büker, Siegbert Drüe, Nicolai Götze, Georg Hartmann, Björn Kalkreuter, Ralf Stemmer, and Ralf Trapp. **Vision-based control of an autonomous disassembly station**. *Robotics and Autonomous Systems* 35:3-4 (2001), 179–189. DOI: 10.1016/S0921-8890(01)00121-X. URL: https://doi.org/10.1016/S0921-8890(01)00121-X (see pages 20, 29, 31, 101).

[Bus+20] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. **Albumentations: fast and flexible image augmentations**. *Information* 11:2 (2020), 125 (see page 116).

[BV20] Yannick Bukschat and Marcus Vetter. **EfficientPose–An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach**. *arXiv preprint arXiv:2011.04307* (2020) (see page 178).

[Che+02] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. **The trimmed iterative closest point algorithm**. In: *Object recognition supported by user interaction for service robots*. Vol. 3. IEEE. 2002, 545–548 (see page 173).

[Che+19] Cong Chen, Longfei Ma, Yan Jia, and Panli Zuo. **Kidney and Tumor Segmentation Using Modified 3D Mask RCNN** (2019) (see pages 26, 120).

[Che01] Ke-Zhang Chen. **Development of integrated design for disassembly and recycling in concurrent engineering**. *Integrated manufacturing systems* (2001) (see page 16).

[Cho17] François Chollet. **Xception: Deep learning with depthwise separable convolutions**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 1251–1258. DOI: 10.1109/CVPR.2017.195. URL: http://dx.doi.org/10.1109/CVPR.2017.195 (see pages 86, 105).

[CMS13] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. **Density-based clustering based on hierarchical density estimates**. In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer. 2013, 160–172 (see pages 73, 74).

[Cof05] Melanie Cofield. "Digital Imaging Basics." In: *Information Technology Lab, School of Information, The University of Texas at Austin, Summer*. 2005 (see page 53).

[Con16] IMAGINE Consortium. *Robots Understanding Their Actions by Imagining Their Effects*. 2016 (see page 42).

[Cou+19] Vincent Couteaux, S Si-Mohamed, O Nempont, T Lefevre, A Popoff, Guillaume Pizaine, N Villain, Isabelle Bloch, A Cotten, and L Boussel. **Automatic knee meniscus tear detection and orientation classification with Mask-RCNN**. *Diagnostic and interventional imaging* 100:4 (2019), 235–242 (see pages 26, 120).

[Cru+08] S Rolando Cruz-Ramirez, Yasushi Mae, Tomohito Takubo, and Tatsuo Arai. **Detection of screws on metal-ceiling structures for dismantling tasks in buildings**. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, 4123–4129. DOI: 10.1109/IROS.2008.4650975. URL: https://doi.org/10.1109/IROS.2008.4650975 (see page 101).

[CV18] Zhaowei Cai and Nuno Vasconcelos. **Cascade r-cnn: Delving into high quality object detection**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 6154–6162 (see pages 26, 120–122).

[CYV00] S Grace Chang, Bin Yu, and Martin Vetterli. **Spatially adaptive wavelet thresholding with context modeling for image denoising**. *IEEE Transactions on image Processing* 9:9 (2000), 1522–1531 (see page 68).

[Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. **Imagenet: A large-scale hierarchical image database**. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, 248–255 (see pages 105, 113, 127).

[DH71] Richard O Duda and Peter E Hart. **Use of the Hough transformation to detect lines and curves in pictures**. Tech. rep. SRI Iinternational Menlo Park CA Artificial Intelligence Center, 1971 (see page 104).

[DHP04] Joseph K Davidson, Kenneth H Hunt, and Gordon R Pennock. **Robots and screw theory: applications of kinematics and statics to robotics**. *J. Mech. Des.* 126:4 (2004), 763–764 (see page 30).

[DHS15] Jifeng Dai, Kaiming He, and Jian Sun. **BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation**. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 2015, 1635–1643. DOI: 10.1109/ICCV.2015.191 (see pages 25, 120).

[DHS16] Jifeng Dai, Kaiming He, and Jian Sun. **Instance-aware semantic segmentation via multi-task network cascades**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 3150–3158 (see pages 24, 119).

[DM03] Anoop Desai and Anil Mital. **Evaluation of disassemblability to enable design for disassembly in mass production**. *International Journal of Industrial Ergonomics* 32:4 (2003), 265–281 (see page 16).

[Do99] Yongtae Do. **Application of neural networks for stereo-camera calibration**. In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Vol. 4. IEEE. 1999, 2719–2722 (see page 64).

[Drö+14] Klaus Dröder, Annika Raatz, Christoph Herrmann, Kathrin Wegener, and Stefan Andrew. **Disassembly of Electric Vehicle Batteries Using the Example of the Audi Q5 Hybrid System**. *Procedia CIRP* 23 (2014), 155–160. ISSN: 22128271. DOI: 10.1016/j.procir.2014.10.098. URL: http://dx.doi.org/10.1016/j.procir.2014.10.098 (see pages 29, 101).

[Duf+08] Joost R Duflou, Gunther Seliger, Sami Kara, Yasushi Umeda, Aldo Ometto, and Barbara Willems. **Efficiency and feasibility of product disassembly: A case-based study**. *CIRP Annals* 57:2 (2008), 583–600 (see pages 13, 16).

[Dvo+17] Nikita Dvornik, Konstantin Shmelkov, Julien Mairal, and Cordelia Schmid. **Blitznet: A real-time deep network for scene understanding**. In: *Proceedings of the IEEE international conference on computer vision*. 2017, 4154–4162 (see pages 24, 119).

[DZ19] Abhishek Dutta and Andrew Zisserman. **The VIA Annotation Software for Images, Audio and Video**. In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. Nice, France: ACM, 2019. DOI: 10.1145/3343031.3350535. URL: https://doi.org/10.1145/3343031.3350535 (see pages 93, 94).

[ECM] ECMA ECMA. **404: The JSON Data Interchange Format. ECMA (European Association for Standardizing Information and Communication Systems), Geneva, Switzerland, 56 BIBLIOGRAPHY October 2013. URL www. ecma-international. org/publications/files**. Tech. rep. ECMA-ST/ECMA-404. pdf (see page 93).

[Ele03] Waste Electrical. **Directive 2002/96/EC of the European Parliament and of the council of 27 January 2003 on waste electrical and electronic equipment (WEEE)**. *Official Journal of the European Union, L* 37 (2003), 24–38 (see page 15).

[ElS+12] Ahmed ElSayed, Elif Kongar, Surendra M Gupta, and Tarek Sobh. **A robotic-driven disassembly sequence generator for end-of-life electronic products**. *Journal of Intelligent & Robotic Systems* 68:1 (2012), 43–52 (see pages 20, 21).

[Els+12] Ahmed Elsayed, Elif Kongar, Surendra M. Gupta, and Tarek Sobh. **A robotic-driven disassembly sequence generator for end-of-life electronic products**. *Journal of Intelligent and Robotic Systems: Theory and Applications* 68:1 (2012), 43–52. ISSN: 09210296. DOI: 10.1007/s10846-012-9667-8. URL: http://dx.doi.org/10.1007/s10846-012-9667-8 (see pages 29, 101).

[EP99] Theodoros Evgeniou and Massimiliano Pontil. **Support vector machines: Theory and applications**. In: *Advanced Course on Artificial Intelligence*. Springer. 1999, 249–257 (see page 124).

[Est+96a] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. **A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise**. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507 (see page 69).

[Est+96b] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. **A density-based algorithm for discovering clusters in large spatial databases with noise.** In: *Kdd*. Vol. 96. 34. 1996, 226–231 (see page 72).

[Eve+10] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. **The Pascal Visual Object Classes (VOC) Challenge**. *International Journal of Computer Vision* 88:2 (June 2010), 303–338. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4 (see pages 106, 114).

[FKS06] C Franke, S Kernbaum, and G Seliger. "Remanufacturing of flat screen monitors." In: *Innovation in life cycle engineering and sustainable development*. Springer, 2006, 139–152 (see page 17).

[For+20] Vanessa Forti, Cornelis P Balde, Ruediger Kuehr, and Garam Bel. **The Global E-waste Monitor 2020: Quantities, flows and the circular economy potential** (2020) (see page 15).

[FSB19] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. **RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free**. *arXiv preprint arXiv:1901.03353* (2019) (see pages 24, 119).

[GF16] Golnaz Ghiasi and Charless C Fowlkes. **Laplacian pyramid reconstruction and refinement for semantic segmentation**. In: *European Conference on Computer Vision*. Springer. 2016, 519–534 (see pages 25, 120).

[GG93] Ardeshir Goshtasby and William A Gruver. **Design of a single-lens stereo camera system**. *Pattern Recognition* 26:6 (1993), 923–937 (see page 62).

[GG99] Askiner Gungor and Surendra M Gupta. **Issues in environmentally conscious manufacturing and product recovery: a survey**. *Computers & Industrial Engineering* 36:4 (1999), 811–853 (see page 16).

[Gil+06] Pablo Gil, Fernando Torres, FG Ortiz, and O Reinoso. **Detection of partial occlusions of assembled components to simplify the disassembly tasks**. *The International Journal of Advanced Manufacturing Technology* 30:5-6 (2006), 530–539 (see page 19).

[Gil+07] Pablo Gil, Jorge Pomares, S vT Puente C Diaz, F Candelas, and F Torres. **Flexible multi-sensorial system for automatic disassembly using cooperative robots**. *International Journal of Computer Integrated Manufacturing* 20:8 (2007), 757–772 (see page 19).

[Gir+14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. **Rich feature hierarchies for accurate object detection and semantic segmentation**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, 580–587 (see pages 26, 120).

[Gir15] Ross Girshick. **Fast r-cnn**. In: *Proceedings of the IEEE international conference on computer vision*. 2015, 1440–1448 (see pages 26, 120).

[Gla93] Chris A Glasbey. **An analysis of histogram-based thresholding algorithms**. *CVGIP: Graphical models and image processing* 55:6 (1993), 532–537 (see page 139).

[Goo+16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. **Deep learning**. Vol. 1. 2. MIT press Cambridge, 2016 (see page 78).

[GW02] Rafael C Gonzalez and Richard E Woods. **Digital image processing (preview)** (2002) (see page 52).

[GWE04] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. **Digital image processing using MATLAB**. Pearson Education India, 2004 (see pages 52, 66, 68).

[HAN10] Rostam Affendi Hamzah, Rosman Abd Rahim, and Zarina Mohd Noh. **Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application**. In: *2010 3rd International Conference on Computer Science and Information Technology*. Vol. 1. IEEE. 2010, 652–657 (see page 64).

[Har+20] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. **Array programming with NumPy**. *Nature* 585:7825 (Sept. 2020), 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2 (see page 153).

[He+16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Deep residual learning for image recognition**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90 (see pages 86, 113).

[He+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. **Identity mappings in deep residual networks**. In: *European conference on computer vision*. Springer. 2016, 630–645. DOI: 10.1007/978-3-319-46493-0_38. URL: http://dx.doi.org/10.1007/978-3-319-46493-0%5C_38 (see page 105).

[He+17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. **Mask r-cnn**. In: *Proceedings of the IEEE international conference on computer vision*. 2017, 2961–2969 (see pages 26, 120, 121).

[Hen+19] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. **AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty**. *arXiv preprint arXiv:1912.02781* (2019) (see page 126).

[HHT00] Karlheinz Hohm, H Muller Hofstede, and Henning Tolle. **Robot assisted disassembly of electronic devices**. In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*. Vol. 2. IEEE. 2000, 1273–1278 (see page 30).

[Hir05] Heiko Hirschmuller. **Accurate and efficient stereo processing by semi-global matching and mutual information**. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, 807–814 (see page 97).

[Hir11] Heiko Hirschmüller. **Semi-global matching-motivation, developments and applications**. *Photogrammetric Week 11* (2011), 173–184 (see page 65).

[HLL10] Yong Seok Heo, Kyong Mu Lee, and Sang Uk Lee. **Robust stereo matching using adaptive normalized cross-correlation**. *IEEE Transactions on pattern analysis and machine intelligence* 33:4 (2010), 807–822 (see page 65).

[How+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. **Mobilenets: Efficient convolutional neural networks for mobile vision applications**. *arXiv preprint arXiv:1704.04861* (2017) (see pages 26, 120).

[HSS18] Jie Hu, Li Shen, and Gang Sun. **Squeeze-and-excitation networks**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 7132–7141 (see page 127).

[Hua+16] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. *Densely Connected Convolutional Networks*. 2016. arXiv: 1608.06993 [cs.CV] (see page 127).

[Hua+17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. **Densely connected convolutional networks**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 4700–4708. DOI: 10.1109/CVPR.2017.243. URL: https://doi.org/10.1109/CVPR.2017.243 (see pages 105, 113, 122).

[Hui08] Jaco Huisman. **2008 Review of Directive 2002/96 on Waste Electrical and Electronic Equipment (WEEE), Final Report**. *http://ec. europa. eu/environment/waste/weee/pdf/final_rep_unu. pdf* (2008) (see page 15).

[HZ03] Richard Hartley and Andrew Zisserman. **Multiple view geometry in computer vision**. Cambridge university press, 2003 (see page 59).

[JAC96] TM Jorgensen, Allan Weimar Andersen, and Steen Sloth Christensen. **Shape recognition system for automatic disassembly of TV-sets**. In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 2. IEEE. 1996, 653–656 (see page 30).

[Jah+19] Ali Jahanian, Quang H Le, Kamal Youcef-Toumi, and Dzmitry Tsetserukou. **See the E-Waste! Training Visual Intelligence to See Dense Circuit Boards for Recycling**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019 (see pages 26, 27).

[JKS95] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. **Machine vision**. Vol. 5. McGraw-hill New York, 1995 (see page 62).

[Joh18] Jeremiah W Johnson. **Adapting mask-rcnn for automatic nucleus segmentation**. *arXiv preprint arXiv:1805.00500* (2018) (see pages 26, 120).

[Jou+16] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. **Learning Visual Features from Large Weakly Supervised Data**. In: *Computer Vision – ECCV 2016*. 2016, 67–84. ISBN: 978-3-319-46478-7 (see pages 26, 120).

[Jou+17] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. **In-datacenter performance analysis of a tensor processing unit**. In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, 1–12 (see page 84).

[Kaf16] Jefkine Kafunah. *Backpropagation In Convolutional Neural Networks*. [last checked 18.09.2020, 11:19]. Sept. 2016. URL: https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/ (see pages 81, 82).

[Kas02] Huang Y Narasimhamurthy A Pande N Kasturi R Camps O. **Wire detection algorithms for navigation**. *NASA Technical report*: 814 (2002) (see pages 124, 125).

[Kaw+18] Yusuke Kawakami, Tetsuo Hattori, Yoshiro Imai, Kazuaki Ando, Yo Horikawa, and RPC Janaka Rajapakse. **Histogram Analysis Method Based on Gaussian Distribution and Curvature Computation (I)—-Peaks and Valleys Detection—-**. In: *Proceedings of International Conference on Artificial Life and Robotics*. 2018, 367–370 (see page 66).

[KB14] Diederik Kingma and Jimmy Ba. **Adam: A Method for Stochastic Optimization**. *International Conference on Learning Representations* (Dec. 2014) (see page 113).

[KFS09] Sebastian Kernbaum, Carsten Franke, and Gunther Seliger. **Flat screen monitor disassembly and testing for remanufacturing**. *International Journal of Sustainable Manufacturing* 1:3 (2009), 347–360 (see pages 32, 33).

[Kho+17] A. Khoreva, R. Benenson, J. Hosang, M. Hein, and B. Schiele. **Simple Does It: Weakly Supervised Instance and Semantic Segmentation**. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, 1665–1674. DOI: 10.1109/CVPR.2017.181 (see pages 25, 120).

[KHS07] Hyung-Ju Kim, Robert Harms, and Gnther Seliger. **Automatic control sequence generation for a hybrid disassembly system**. *IEEE transactions on automation science and engineering* 4:2 (2007), 194–205 (see page 17).

[KJ00] Björn Karlsson and Jan-Ove Järrhed. **Recycling of electrical motors by automatic disassembly**. *Measurement Science and Technology* 11:4 (May 2000), 350–357. DOI: 10.1088/0957-0233/11/4/303 (see page 21).

[KKS09] H-J Kim, S Kernbaum, and G Seliger. **Emulation-based control of a disassembly system for LCD monitors**. *The International Journal of Advanced Manufacturing Technology* 40:3-4 (2009), 383–392 (see pages 17, 18, 33, 182).

[Kno+02] Reinhard Knoth, M Brandstotter, B Kopacek, and P Kopacek. **Automated disassembly of electr (on) ic equipment**. In: *Conference Record 2002 IEEE International Symposium on Electronics and the Environment (Cat. No. 02CH37273)*. IEEE. 2002, 290–294 (see page 17).

[KP18] Andreas Kamilaris and Francesc X Prenafeta-Boldú. **Deep learning in agriculture: A survey**. *Computers and electronics in agriculture* 147 (2018), 70–90 (see page 78).

[KRK07] Sami Kara, Fatida Rugrungruang, and Hartmut Kaebernick. **Simulation modelling of reverse logistics networks**. *International journal of production economics* 106:1 (2007), 61–69 (see page 14).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. **Imagenet classification with deep convolutional neural networks**. In: *Advances in neural information processing systems*. 2012, 1097–1105 (see pages 25, 119).

[Kum+19] Sameer Kumar, Victor Bitorff, Dehao Chen, Chiachen Chou, Blake Hechtman, HyoukJoong Lee, Naveen Kumar, Peter Mattson, Shibo Wang, Tao Wang, et al. **Scale mlperf-0.6 models on google tpu-v3 pods**. *arXiv preprint arXiv:1909.09756* (2019) (see page 84).

[Kwo04] Soon Kwon. **Threshold selection based on cluster analysis**. *Pattern Recognition Letters* 25 (July 2004), 1045–1050. DOI: 10.1016/j.patrec.2004.03.001 (see page 67).

[Lan+10] Manuel Lang, Alexander Hornung, Oliver Wang, Steven Poulakos, Aljoscha Smolic, and Markus Gross. **Nonlinear disparity mapping for stereoscopic 3D**. *ACM Transactions on Graphics (TOG)* 29:4 (2010), 1–10 (see page 62).

[LG04] A.J.D. Lambert and Surendra Gupta. **Disassembly Modeling for Assembly, Maintenance, Reuse and Recycling** (Dec. 2004). DOI: 10.1201/9780203487174 (see page 14).

[Li+17] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. **Fully convolutional instance-aware semantic segmentation**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 2359–2367 (see pages 24, 119).

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. **Microsoft coco: Common objects in context**. In: *European conference on computer vision*. Springer. 2014, 740–755 (see pages 25, 120, 121, 127).

[Lin+17a] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. **Refinenet: Multi-path refinement networks for high-resolution semantic segmentation**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 1925–1934 (see pages 25, 120).

[Lin+17b] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. **Feature pyramid networks for object detection**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 2117–2125 (see pages 26, 120).

[Lin+19] Di Lin, Dingguo Shen, Siting Shen, Yuanfeng Ji, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. **ZigZagNet: Fusing Top-Down and Bottom-Up Context for Object Segmentation**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 7490–7499 (see pages 24, 119).

[Lit+17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. **A survey on deep learning in medical image analysis**. *Medical image analysis* 42 (2017), 60–88 (see page 78).

[Liu+18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. **Path aggregation network for instance segmentation**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 8759–8768 (see pages 24, 119).

[Liu+19] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. **Cbnet: A novel composite backbone network architecture for object detection**. *arXiv preprint arXiv:1909.03625* (2019) (see page 122).

[LK00] DooHyun Lee and InSo Kweon. **A novel stereo camera system by a biprism**. *IEEE Transactions on Robotics and Automation* 16:5 (2000), 528–541 (see page 62).

[Low10] David Lowe. **The computer vision industry**. *Computer Science Department* 2366 (2010) (see page 22).

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. **Fully convolutional networks for semantic segmentation**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, 3431–3440 (see pages 25, 119).

[LZP16] Kuang Liu, Mingmin Zhang, and Zhigeng Pan. **Facial expression recognition with CNN ensemble**. In: *2016 international conference on cyberworlds (CW)*. IEEE. 2016, 163–166 (see page 105).

[Mac+67] James MacQueen et al. **Some methods for classification and analysis of multivariate observations**. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, 281–297 (see page 69).

[Mad+15] Jorge Enrique Madrigal-Arias, Rosalba Argumedo-Delira, Alejandro Alarcón, Ma Mendoza-López, Oscar García-Barradas, Jesús Samuel Cruz-Sánchez, Ronald Ferrera-Cerrato, Maribel Jiménez-Fernández, et al. **Bioleaching of gold, copper and nickel from waste cellular phone PCBs and computer goldfinger motherboards by two Aspergillus nigerstrains**. *Brazilian Journal of Microbiology* 46:3 (2015), 707–713 (see page 7).

[MAT10] MATLAB. **version 7.10.0 (R2010a)**. Natick, Massachusetts: The MathWorks Inc., 2010 (see pages 31, 61).

[Mat89] Larry Matthies. **Dynamic stereo vision**. PhD thesis. Carnegie Mellon University USA, 1989 (see page 65).

[Mer+10] Munir Merdan, Wilfried Lepuschitz, Thomas Meurer, and Markus Vincze. **Towards ontology-based automated disassembly systems**. In: *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE. 2010, 1392–1397 (see page 21).

[MHA17a] Leland McInnes, John Healy, and Steve Astels. **hdbscan: Hierarchical density based clustering**. *Journal of Open Source Software* 2:11 (2017), 205 (see page 69).

[MHA17b] Leland McInnes, John Healy, and Steve Astels. **hdbscan: Hierarchical density based clustering**. *The Journal of Open Source Software* 2:11 (May 2017). DOI: 10.21105/joss.00205. URL: https://doi.org/10.21105%2Fjoss.00205 (see pages 75, 140).

[Mit+97] Tom M Mitchell et al. **Machine learning** (1997) (see page 75).

[MKS17] Vikash Mishra, Shobhit Kumar, and Neeraj Shukla. **Image Acquisition and Techniques to Perform Image Acquisition**. *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology* 9 (July 2017). DOI: 10.18090/samriddhi.v9i01.8333 (see page 53).

[MKS18] M. Mezghani, J. Kang, and F. Sèdes. **Industrial Requirements Classification for Redundancy and Inconsistency Detection in SEMIOS**. In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. 2018, 297–303. DOI: 10.1109/RE.2018.00037 (see page 70).

[MMS17] Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. **Wire Detection using Synthetic Data and Dilated Convolutional Networks for Unmanned Aerial Vehicles**. *International Conference on Intelligent Robots and Systems* (2017) (see page 125).

[Mot+15] S Motala, S Ngandu, S Mti, F Arends, L Winnaar, E Khalema, M Makiwane, C Ndinda, B Moolman, T Maluleke, et al. **Millennium development goals: Country report 2015** (2015) (see page 1).

[MP13] Durga Prasad Mohapatra and Srikanta Patnaik. **Intelligent computing, networking, and informatics**. *Advances in Intelligent Systems and Computing* 243 (2013), 1047–1054 (see page 51).

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. **Introduction to Information Retrieval**. New York, NY, USA: Cambridge University Press, 2008 (see page 140).

[Nev76] Ramakant Nevatia. **Depth measurement by motion stereo**. *Computer Graphics and Image Processing* 5:2 (1976), 203–214 (see page 62).

[NF15] Martin Naumann and Manuel Fechter. **Robots as enablers for changeability in assembly applications**. In: *15. Internationales Stuttgarter Symposium*. Springer. 2015, 1155–1171 (see page 30).

[Nob+20] Tomomi Nobashi, Claudia Zacharias, Jason K Ellis, Valentina Ferri, Mary Ellen Koran, Benjamin L Franc, Andrei Iagaru, and Guido A Davidzon. **Performance comparison of individual and ensemble CNN models for the classification of brain 18F-FDG-PET scans**. *Journal of digital imaging* 33:2 (2020), 447–455 (see page 105).

[OA05] Abhijit S Ogale and Yiannis Aloimonos. **Shape and the stereo correspondence problem**. *International Journal of Computer Vision* 65:3 (2005), 147–162 (see page 64).

[Ole+17] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. **Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning**. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, 1366–1373 (see pages 172–174).

[Ore+01] Ram Oren, David S Ellsworth, Kurt H Johnsen, Nathan Phillips, Brent E Ewers, Chris Maier, Karina VR Schäfer, Heather McCarthy, George Hendrey, Steven G McNulty, et al. **Soil fertility limits carbon sequestration by forest ecosystems in a CO 2-enriched atmosphere**. *Nature* 411:6836 (2001), 469–472 (see pages 15, 16).

[Ots79] Nobuyuki Otsu. **A threshold selection method from gray-level histograms**. *IEEE transactions on systems, man, and cybernetics* 9:1 (1979), 62–66 (see page 68).

[Par13] Paritosh Parmar. **Use of computer vision to detect tangles in tangled objects**. *2013 IEEE 2nd International Conference on Image Information Processing, IEEE ICIIP 2013* (2013), 39–44. DOI: 10.1109/ICIIP.2013.6707551 (see page 124).

[PAV19] Fábio Perez, Sandra Avila, and Eduardo Valle. **Solo or ensemble? choosing a cnn architecture for melanoma classification**. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019 (see page 105).

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research* 12 (2011), 2825–2830 (see pages 153, 154).

[Peh06] Martin Pehnt. **Dynamic life cycle assessment (LCA) of renewable energy technologies**. *Renewable Energy* 31:1 (2006), 55–71. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2005.03.002. URL: http://www.sciencedirect.com/science/article/pii/S0960148105000662 (see page 13).

[Pet05] AK Peterson. **Introduction to Basic Measures of a Digital Image for Pictorial Collections, Prints & Photographs Division**. In: *Library of Congress, Washington, DC*. 2005, 20540–4720 (see page 53).

[Pin+16] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. **Learning to refine object segments**. In: *European Conference on Computer Vision*. Springer. 2016, 75–91 (see pages 25, 120).

[PM66] Judith MS Prewitt and Mortimer L Mendelsohn. **The analysis of cell images**. *Annals of the New York Academy of Sciences* 128:3 (1966), 1035–1053 (see page 139).

[Pom+04a] Jorge Pomares, Santiago T. Puente, Fernando Torres, Francisco A. Candelas, and Pablo Gil. **Virtual disassembly of products based on geometric models**. *Computers in Industry* 55:1 (2004), 1–14. ISSN: 01663615. DOI: 10.1016/j.compind.2004.03.001. URL: http://dx.doi.org/10.1016/j.compind.2004.03.001 (see pages 30, 101).

[Pom+04b] Jorge Pomares, ST Puente, Fernando Torres, FA Candelas, and Pablo Gil. **Virtual disassembly of products based on geometric models**. *Computers in Industry* 55:1 (2004), 1–14 (see page 21).

[Qia97] Ning Qian. **Binocular disparity and the perception of depth**. *Neuron* 18:3 (1997), 359–368 (see page 65).

[Qui+09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. **ROS: an open-source Robot Operating System**. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, 5 (see pages 31, 46).

[Rai20] Ankush Rai. **Attribute based level adaptive thresholding algorithm for object extraction**. *Journal of Advancements in Robotics* 1:2 (2020), 13–17 (see page 67).

[Rao+20] Sunil Rao, Sameeksha Katoch, Vivek Narayanaswamy, Gowtham Muniraju, Cihan Tepedelenlioglu, Andreas Spanias, Pavan Turaga, Raja Ayyanar, and Devarajan Srinivasan. **Machine Learning for Solar Array Monitoring, Optimization, and Control**. *Synthesis Lectures on Power Electronics* 7:1 (2020), 1–91 (see page 69).

[Ras+97] Wayne S Rasband et al. *ImageJ*. 1997 (see page 31).

[Ras12] WS Rasband. **ImageJ: Image processing and analysis in Java**. *ascl* (2012), ascl–1206 (see page 31).

[RC11] Radu Bogdan Rusu and Steve Cousins. **3d is here: Point cloud library (pcl)**. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, 1–4 (see page 136).

[Ren+15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. **Faster r-cnn: Towards real-time object detection with region proposal networks**. In: *Advances in neural information processing systems*. 2015, 91–99 (see pages 26, 120).

[RF18] Joseph Redmon and Ali Farhadi. **Yolov3: An incremental improvement. arXiv 2018**. *arXiv preprint arXiv:1804.02767* (2018), 1–6 (see page 106).

[RFB15a] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (see page 130).

[RFB15b] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. **U-net: Convolutional networks for biomedical image segmentation**. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, 234–241 (see pages 25, 119).

[Rok09] Lior Rokach. "A survey of clustering algorithms." In: *Data mining and knowledge discovery handbook*. Springer, 2009, 269–298 (see page 69).

[ROL11] Alan Ryan, Liam O'Donoghue, and Huw Lewis. **Characterising components of liquid crystal displays to facilitate disassembly**. *Journal of Cleaner Production* 19:9-10 (2011), 1066–1071 (see pages 32, 33).

[ŞA20] Ayşe Şerbetçi and Yusuf Sinan Akgül. **End-to-end training of cnn ensembles for person re-identification**. *Pattern Recognition* (2020), 107319 (see page 105).

[Sac96] Jonathan Sachs. **Digital image basics**. *Digital Light & Color* 1999 (1996) (see page 53).

[Sam59] Arthur L Samuel. **Some studies in machine learning using the game of checkers**. *IBM Journal of research and development* 3:3 (1959), 210–229 (see page 75).

[San+18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. **Mobilenetv2: Inverted residuals and linear bottlenecks**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 4510–4520 (see page 121).

[SB07] Indranil Sarkar and Manu Bansal. **A wavelet-based multiresolution approach to solve the stereo correspondence problem using mutual information**. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37:4 (2007), 1009–1014 (see page 64).

[Sch15] Timo Schmid. *An Analysis Of Possible Approaches To Boost Existing Solutions*. Master's Thesis. Heidelberg Collaboratory for Image Processing, Computer Vision group, Oct. 2015 (see page 70).

[Sch18] Konstantin Schauwecker. **Real-Time Stereo Vision on FPGAs with SceneScan**. In: *Forum Bildverarbeitung 2018*. KIT Scientific Publishing. 2018, 339 (see page 97).

[SLR13] Robert Spangenberg, Tobias Langner, and Raúl Rojas. **Weighted semi-global matching and center-symmetric census transform for robust driver assistance**. In: *International Conference on Computer Analysis of Images and Patterns*. Springer. 2013, 34–41 (see page 65).

[SS04] Mehmet Sezgin and Bülent Sankur. **Survey over image thresholding techniques and quantitative performance evaluation**. *Journal of Electronic imaging* 13:1 (2004), 146–166 (see page 66).

[SSP00] Rajeev Solomon, Peter A Sandborn, and Michael G Pecht. **Electronic part life cycle concepts and obsolescence forecasting**. *IEEE Transactions on Components and Packaging Technologies* 23:4 (2000), 707–717 (see page 15).

[Su+21] Yongzhi Su, Jason Rambach, Alain Pagani, and Didier Stricker. **SynPo-Net—Accurate and Fast CNN-Based 6DoF Object Pose Estimation Using Synthetic Training**. *Sensors* 21:1 (2021), 300 (see page 178).

[Sug+15] Daisuke Sugimura, Takuya Mikami, Hiroki Yamashita, and Takayuki Hamamoto. **Enhancing color images of extremely low light scenes based on RGB/NIR images acquisition with different exposure times**. *IEEE Transactions on Image Processing* 24:11 (2015), 3586–3597 (see page 54).

[SW95] Andrew J Spicer and Michael H Wang. **A software tool for end-of-life-cycle consideration within a DSS approach to environmentally conscious design and manufacturing**. *Computers & Industrial Engineering* 29:1-4 (1995), 501–505 (see page 16).

[SZ15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 (see page 86).

[Sze+15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV] (see page 127).

[Sze+16a] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV] (see page 127).

[Sze+16b] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. **Rethinking the inception architecture for computer vision**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 2818–2826. DOI: 10.1109/CVPR.2016.308. URL: https://doi.org/10.1109/CVPR.2016.308 (see page 105).

[Sze+17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. **Inception-v4, inception-resnet and the impact of residual connections on learning**. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (see page 105).

[Tan+18] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. **Introduction to Data Mining (2Nd Edition)**. 2nd. Pearson, 2018 (see pages 72, 73, 140).

[TL19] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2019. arXiv: 1905.11946 [cs.LG] (see pages 88, 113, 126–129).

[TMU19] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. **Data augmentation using random image cropping and patching for deep CNNs**. *IEEE Transactions on Circuits and Systems for Video Technology* (2019) (see page 126).

[TN00] Martin Tonko and Hans-Hellmut Nagel. **Model-based stereo-tracking of non-polyhedral objects for automatic disassembly experiments**. *International Journal of Computer Vision* 37:1 (2000), 99–118 (see pages 21, 30).

[Ton+99] M. Tonko, V. Gengenbach, H.-H Nagel, Radu Horaud, and R. Ffi. **Towards the Integration of Object Recognition and Visual Servoing for Disassembly of Used Cars** (Jan. 1999) (see pages 22, 23).

[Tor+04] Fernando Torres, Pablo Gil, ST Puente, Jorge Pomares, and R Aracil. **Automatic PC disassembly for component recovery**. *The international journal of advanced manufacturing technology* 23:1-2 (2004), 39–46 (see pages 19, 23, 24, 30).

[TPA03a] Fernando Torres, Santiago Puente, and Rafael Aracil. **Disassembly Planning Based on Precedence Relations among Assemblies**. *International Journal of Advanced Manufacturing Technology* 21 (Feb. 2003). DOI: 10.1007/s001700300037 (see page 21).

[TPA03b] Fernando Torres, ST Puente, and R Aracil. **Disassembly planning based on precedence relations among assemblies**. *The International Journal of Advanced Manufacturing Technology* 21:5 (2003), 317–327 (see page 19).

[Tre+18] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. 2018. arXiv: 1809.10790 [cs.RO] (see page 178).

[Tro20] Henrik Trommer. *Evaluating Super Resolution Images On DCNN Based Screwhead Classification Task*. Bachelor's Thesis. Georg-August-Universität Göttingen, III.Physics Institute, Biophysics Department, Sept. 2020 (see page 51).

[UIU19] PACE United Nations Envrionment Programme, ILO ITU, and United Nations University UNIDO. **A New Circular Vision for Electronics Time for a Global Reboot** (2019) (see page 6).

[Uki07] Hiroyuki Ukida. **Visual defect inspection of rotating screw heads**. In: *SICE Annual Conference 2007*. IEEE. 2007, 1478–1483 (see pages 21, 101).

[Van+14] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. **scikit-image: image processing in Python**. *PeerJ* 2 (2014), e453 (see pages 66, 67, 153, 154).

[VC15] Supachai Vongbunyong and Wei Hua Chen. "Disassembly automation." In: *Disassembly Automation*. Springer, 2015, 25–54 (see pages 14, 18, 20, 22).

[VD09] Guido Van Rossum and Fred L. Drake. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697 (see pages 74, 75).

[Vig+10] Federico Vigano, Stefano Consonni, Mario Grosso, and Lucia Rigamonti. **Material and energy recovery from Automotive Shredded Residues (ASR) via sequential gasification and combustion**. *Waste Management* 30:1 (2010), 145–153 (see page 16).

[Vir+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. **SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python**. *Nature Methods* 17 (2020), 261–272. DOI: 10.1038/s41592-019-0686-2 (see page 153).

[Wan+04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. **Image quality assessment: from error visibility to structural similarity**. *IEEE transactions on image processing* 13:4 (2004), 600–612 (see page 128).

[Wan+19] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. **Deep learning for sensor-based activity recognition: A survey**. *Pattern Recognition Letters* 119 (2019), 3–11 (see page 78).

[WCH+92] Juyang Weng, Paul Cohen, Marc Herniou, et al. **Camera calibration with distortion models and accuracy evaluation**. *IEEE Transactions on pattern analysis and machine intelligence* 14:10 (1992), 965–980 (see page 64).

[Weg+15] Kathrin Wegener, Wei Hua Chen, Franz Dietrich, Klaus Dröder, and Sami Kara. **Robot assisted disassembly for the recycling of electric vehicle batteries**. *Procedia CIRP* 29 (2015), 716–721. ISSN: 22128271. DOI: 10.1016/j.procir.2015.02.051 (see pages 21, 29, 101, 102).

[Wei+06] A. Weigl-Seitz, K. Hohm, M. Seitz, and H. Tolle. **On strategies and solutions for automated disassembly of electronic devices**. *International Journal of Advanced Manufacturing Technology* 30:5-6 (2006), 561–573. ISSN: 02683768. DOI: 10.1007/s00170-005-0043-8 (see pages 20, 21, 29, 101).

[WWB19] Yu Emma Wang, Gu-Yeon Wei, and David Brooks. **Benchmarking tpu, gpu, and cpu platforms for deep learning**. *arXiv preprint arXiv:1907.10701* (2019) (see pages 84, 88).

[Xia+18] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. *PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes*. 2018. arXiv: 1711.00199 [cs.CV] (see page 178).

[Xie+08] S. Q. Xie, D. Cheng, S. Wong, and E. Haemmerle. **Three-dimensional object recognition system for enhancing the intelligence of a KUKA robot**. *International Journal of Advanced Manufacturing Technology* 38:7-8 (2008), 822–839. ISSN: 02683768. DOI: 10.1007/s00170-007-1112-y (see pages 21, 30, 101).

[Xie+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. **Aggregated residual transformations for deep neural networks**. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 1492–1500. DOI: 10.1109/CVPR.2017.634. URL: https://doi.org/10.1109/CVPR.2017.634 (see pages 105, 123).

[Xie+19] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. *Self-training with Noisy Student improves ImageNet classification*. 2019. arXiv: 1911.04252 [cs.LG] (see page 113).

[XLC21] Feng Xiong, Chengju Liu, and Qijun Chen. **Region Pixel Voting Network (RPVNet) for 6D Pose Estimation from Monocular Image**. *Applied Sciences* 11:2 (2021). ISSN: 2076-3417. DOI: 10.3390/app11020743. URL: https://www.mdpi.com/2076-3417/11/2/743 (see page 178).

[Yan+20] Jennie Yang, Jens Bertram, Thomas Schettgen, Peter Heitland, Damian Fischer, Fatima Seidu, Michael Felten, Thomas Kraus, Julius N Fobil, and Andrea Kaifie. **Arsenic burden in e-waste recycling workers–a cross-sectional study at the Agbogbloshie e-waste recycling site, Ghana**. *Chemosphere* (2020), 127712 (see page 3).

[Yil+20] Erenus Yildiz., Tobias Brinker., Erwan Renaudo., Jakob J. Hollenstein., Simon Haller-Seeber., Justus Piater., and Florentin Wörgötter. **A Visual Intelligence Scheme for Hard Drive Disassembly in Automated Recycling Routines**. In: *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,* INSTICC. SciTePress, 2020, 17–27. ISBN: 978-989-758-479-4. DOI: 10.5220/0010016000170027 (see pages 121, 122, 135).

[YK15] Fisher Yu and Vladlen Koltun. **Multi-scale context aggregation by dilated convolutions**. *arXiv preprint arXiv:1511.07122* (2015) (see page 125).

[YW19] Erenus Yildiz and Florentin Wörgötter. **DCNN-Based Screw Detection for Automated Disassembly Processes**. In: *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2019, 187–192 (see pages 101–112, 117, 161).

[YW20] Erenus Yildiz. and Florentin Wörgötter. **DCNN-Based Screw Classification in Automated Disassembly Processes**. In: *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,* INSTICC. SciTePress, 2020, 61–68. DOI: 10.5220/0009979900610068 (see pages 111–118).

[ZDK01] HARALD Zebedin, KONRAD Daichendt, and PETER Kopacek. **A new strategy for a flexible semi-automatic disassembling cell of printed circuit boards**. In: *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*. Vol. 3. IEEE. 2001, 1742–1746 (see page 18).

[Zha+17] Bo Zhao, Jiashi Feng, Xiao Wu, and Shuicheng Yan. **A survey on deep learning-based fine-grained object classification and semantic segmentation**. *International Journal of Automation and Computing* 14:2 (2017), 119–135 (see page 78).

[Zha+19] Man Zhang, Yong Zhou, Jiaqi Zhao, Yiyun Man, Bing Liu, and Rui Yao. **A survey of semi- and weakly supervised semantic segmentation of images**. *Artificial Intelligence Review* (Dec. 2019). ISSN: 1573-7462. DOI: 10.1007/s10462-019-09792-7 (see pages 25, 120).

[Zha+21] Heng Zhao, Chungang Zhuang, Lei Jia, and Han Ding. "6D Pose Estimation of Texture-Less Object in RGB-D Images." In: *Mechatronics and Machine Vision in Practice 4*. Springer, 2021, 45–58 (see page 178).

[Zha12] Zhengyou Zhang. **Microsoft kinect sensor and its effect**. *IEEE multimedia* 19:2 (2012), 4–10. DOI: 10.1109/MMUL.2012.24. URL: https://doi.org/10.1109/MMUL.2012.24 (see page 102).

[ZW11] Yudong Zhang and Lenan Wu. **Optimal multi-level thresholding based on maximum Tsallis entropy via an artificial bee colony approach**. *Entropy* 13:4 (2011), 841–859 (see page 67).

[ZZ14] Cha Zhang and Zhengyou Zhang. "Calibration between depth and color sensors for commodity depth cameras." In: *Computer vision and machine learning with RGB-D sensors*. Springer, 2014, 47–64 (see page 57).

List of Publications

Articles in Conference Proceedings

[Yil+20] Erenus Yildiz., Tobias Brinker., Erwan Renaudo., Jakob J. Hollenstein., Simon Haller-Seeber., Justus Piater., and Florentin Wörgötter. **A Visual Intelligence Scheme for Hard Drive Disassembly in Automated Recycling Routines**. In: *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,* INSTICC. SciTePress, 2020, 17–27. ISBN: 978-989-758-479-4. DOI: 10.5220/0010016000170027.

[YW19] Erenus Yildiz and Florentin Wörgötter. **DCNN-Based Screw Detection for Automated Disassembly Processes**. In: *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2019, 187–192.

[YW20] Erenus Yildiz. and Florentin Wörgötter. **DCNN-Based Screw Classification in Automated Disassembly Processes**. In: *Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,* INSTICC. SciTePress, 2020, 61–68. DOI: 10.5220/0009979900610068.