Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Engineering - Robotics
30.0 credits

COMPUTER VISION FOR VOLUME ESTIMATION AND MATERIAL CLASSIFICATION

Oliver Lagelius
ols16001@student.mdu.se

Ludwig Wässman
lwn15002@student.mdu.se

Examiner: Masoud Daneshtalab
Mälardalen University, Västerås, Sweden

Supervisor: Ali Zoljodi
Mälardalen University, Västerås, Sweden

Company Supervisor: Meysam Karami Rad
Volvo Construction Equipment, Eskilstuna

12/07/2023

Abstract

Vehicular automation is a rapidly advancing field within robotics. These autonomous machines have the potential to perform burdensome and dangerous tasks that historically have been executed by humans which has been a long-time goal for the industry. This thesis aims to develop a computer vision system to enable volume estimation and material classification of the material inside the bucket of an autonomous wheel loader. This information is crucial for autonomous wheel loaders to make decisions. The system is intended to be self-calibrating to ensure future adaptability to different bucket sizes. A Convolutional Neural Network (CNN) based edge detecting network referred to as Dense Extreme Inception Network for Edge Detection (DexiNed) is proposed to both remove redundant information and enhance desired information. By combining the depth perception from a stereo camera and the information extracted from the DexiNed a proposed solution to estimate the volume is presented. A Simple Linear Iterative Clustering (SLIC) approach is applied to extract the material to enable classification of the material. The estimated volume is compared to an annotated true baseline for validation of the system. The thesis presents the precision of the volume estimation and showcases the result of material extraction using three different segment sizes with the SLIC. Additionally, the thesis presents issues concerning material classification.

Keywords: *Clustering, DexiNed, Image Processing, SLIC Segmentation, Stereo Vision*

Acknowledgements

The authors would like to thank our academic supervisor Ali Zoljodi for his valuable insights. We would also like to thank our company supervisor Meysam Karami Rad for all his help with everything from gaining us access to the Volvo wheel-loaders to proofreading the thesis. We would like to place a special thanks to Volvo Construction Equipment for providing us the chance to sit at their office in Eskilstuna alongside their experienced engineers. We would also like to acknowledge Mälardalens University and thank the institution for the great education in robotics that has enabled this thesis. Last but not least, the authors would like to thank our fellow students Jonathan Fernkvist, Inas Hamzic, Karl Söderlund, and Simon Friberg for their valuable conversations, ideas, and proofreading.

Contents

1. Introduction **1**
1.1 Problem Formulation . . . 2
1.2 Purpose and Goals . . . 2
1.3 Limitations . . . 2

2. Background **3**
2.1 Wheel Loaders . . . 3
2.2 Computer Vision . . . 3
2.3 Supervised Learning & Unsupervised Learning . . . 3
2.4 Stereo Vision . . . 4
2.4.1 Stereo Correspondence . . . 5
2.5 Superpixel Segmentation . . . 5
2.5.1 Simple Linear Iterative Clustering for Superpixel Segmentation . . . 5
2.5.2 Threshold Based Classification for Superpixel Segments . . . 6
2.6 K-Means Clustering . . . 6
2.7 Convolutional Neural Network . . . 7
2.7.1 Convolution . . . 7
2.7.2 Padding . . . 7
2.7.3 Activation Function . . . 7
2.7.4 Pooling Layer . . . 8
2.8 Dense Extreme Inception Network for Edge Detection . . . 8
2.9 Adaptive Histogram Equalization . . . 8
2.10 Contrast Limited Adaptive Histogram Equalization . . . 8
2.11 Contour Detection . . . 9
2.12 Morphological Image Processing . . . 9
2.13 Median Filter . . . 9
2.14 Sklanskys Algorithm for Convex Hull . . . 9
2.15 Green's Theorem . . . 10

3. Related Work **11**
3.1 Superpixel Segmentation . . . 11
3.2 DexiNed . . . 12
3.3 Distance & Volume Measurement using Stereo Camera . . . 12

4. Method **14**
4.1 System Development Research Methodology . . . 14
4.2 Process of Systems Development Research Methodology . . . 15
4.3 Software . . . 15
4.4 Hardware . . . 16
4.4.1 OAK-D PoE . . . 16
4.4.2 Computer . . . 16

5. Ethical Considerations **16**

6. Implementation **17**
6.1 System Overview . . . 17
6.2 Software for Data Acquisition . . . 18
6.3 Data Acquisition . . . 18
6.3.1 Camera Setup One . . . 19
6.3.2 Camera Setup Two . . . 19
6.3.3 Camera Setup Three . . . 20
6.3.4 Dataset . . . 21
6.4 Bucket Extraction . . . 21
6.4.1 Validation of the Bucket Extraction . . . 24
6.5 Material Extraction . . . 25

6.5.1 Validation of Material Extraction 27
6.6 Material Classification 27
6.7 Volume Estimation 29
6.7.1 Surface Reconstruction 29
6.7.2 Lid Construction & Interpolation 30
6.7.3 Volume Calculation 31

7. Results **32**
7.1 Bucket Extraction 32
7.2 Volume Estimation 32
7.3 Material Extraction 33

8. Conclusions **35**

9. Discussion **36**
9.1 Bucket Extraction 36
9.2 Material Extraction 36
9.3 Volume Estimation 36
9.4 Material Classification 38

References **39**

List of Figures

1 Volvo L220 Wheel-Loader . . . 1
2 Stereo system . . . 4
3 Superpixel segmentation process . . . 6
4 K-mean clustering example . . . 7
5 DexiNed architecture . . . 8
6 Sklanskys Part 1 & 2 . . . 10
7 The region D bounded by the closed curve. . . . 10
8 Edge-map predictions with state-of-the-art algorithms and DexiNed, from paper[14] 12
9 Flowchart of a multi-methodological approach to Is research [31] . . . 14
10 A Research Process of Systems Development Research Methodology [31] . . . 15
11 System Overview . . . 17
12 Oak-D firmware configuration . . . 19
13 Camera Setup One . . . 19
14 Camera Setup Two . . . 20
15 Camera Setup Three . . . 20
16 Original bucket image and Dexined fused edge-map . . . 21
17 Disparity image and Dexined fused edge-map . . . 21
18 Threshold binary image and Bitwise AND binary image. . . . 22
19 Detected contours with the convex hulls. . . . 22
20 Largest detected contour with convex hull . . . 23
21 Bucket extraction output . . . 23
22 Ideal bucket extraction. . . . 24
23 Successful extraction. . . . 25
24 Flowchart of the material extraction . . . 25
25 Step-by-Step material extraction . . . 26
26 Masked Edge-Map . . . 27
27 Binary Threshold image and opened binary image . . . 28
28 Material classification . . . 28
29 Surfaces reconstruction plot . . . 30
30 Lid construction plot . . . 30
31 Results Volume Estimation . . . 33
32 Graph of the accuracy of the material extraction algorithm . . . 34
33 Graph of the wrongfully clustered material . . . 34

List of Abbreviations

DexiNed Dense Extreme Inception Network for Edge Detection i

AHE Adaptive Histogram Equalization 8

CLAHE Contrast Limited Adaptive Histogram Equalization 8

CNN Convolutional Neural Network i

PoE Power-over-Ethernet 16

FPS Frames per second 16

MP Mega Pixels 16

SoM System Module 16

VCC Vector Code Correlation 12

BIPED Barcelona Images for Perceptual Edge Detection 12

SLIC Simple Linear Iterative Clustering i

SIFT Scale-Invariant Feature Transform 5

SURF Speed Up Robust Features 5

BSDS500 Berkeley Segmentation Dataset 500 12

RNN Recurrent Neural Networks 1

ECU Electronic Control Units 1

ReLU Rectified Linear Unit 7

1. Introduction

Vehicular automation is a dynamic domain within the field of robotics, continuously advancing at a rapid pace. Among these advancements, automated wheel loaders like the Volvo LX03 are considered the future of modern mining. The ability to have autonomous machines perform burdensome and dangerous tasks that historically have been executed by humans has been a goal for a long time in the industry [1]. Given the potential profitability of having wheel loaders operate continuously at mining sites, investors have shown significant interest in this technology. However, engineers face several challenges that must be addressed in order for these machines to work efficiently. These challenges include various aspects, such as optimising performance, addressing safety concerns, enhancing productivity, and refining the overall operational effectiveness of automated wheel loaders. Solving these issues is critical to fully capitalise on the potential benefits offered by autonomous wheel loaders in the mining and construction industry [2]. Volvo Construction Equipment has produced innovative solutions for wheel-loaders, dumpers, excavators, and other construction equipment for many decades [3]. In recent years the increase in processing power of small embedded computers has spiked an interest in autonomous and intelligent embedded solutions for construction equipment and automobiles in general. Since the 1980s, automation has utilised Electronic Control Units (ECU) for CAN-bus-based embedded sensor connectivity. With the increased processing power of the ECU, more complex sensors, information systems, and control systems have been added [4]. One of these computationally heavy sensors is the *optical sensor*. Optical sensors in combination with AI and image processing algorithms are used to enable computer vision [5]. Computer vision is a field of study within robotic science that enables computers to interpret and understand visual information from the world through images and video. The goal of computer vision is to develop models and algorithms that analyse and recognise information from image frames. Computer vision is influenced by multiple disciplines, including machine learning, image processing, and pattern recognition. Object detection and classification, image segmentation, stereo vision, and scene reconstruction are a few of the key tasks in computer vision [5]. Researchers have developed various methods to extract relevant information from images. CNN have shown success in image classification and object detection, [5] while Recurrent Neural Networks (RNN) have been used for video analysis and image captioning [5]. Computer vision has numerous applications in various fields, including security and surveillance, medical imaging, self-driving cars, and autonomous construction equipment [5].

The main objective of this thesis is to develop and implement an unsupervised machine-learning algorithm that can effectively identify materials with varying dimensions present within a wheel loader bucket, while also estimating the volume of the material in the bucket. The research delves into the current advancements in the field of optical sensors, image processing techniques, and machine learning algorithms specifically designed for the detection of grit and rocks, as well as accurate volume estimation. By leveraging state-of-the-art methodologies and technologies, this thesis aims to contribute to the existing knowledge in the domain of material detection and volume estimation. The focus lies on exploring innovative approaches that combine optical sensors, image processing techniques, and machine learning algorithms to achieve reliable and accurate results.

Figure 1: Volvo L220 Wheel-Loader. This is a similar vehicle to the one used in this project. The image belongs to ©Volvo CE

1.1 Problem Formulation

This thesis aims to explore the process of gathering a diverse dataset that consists of images showcasing a wheel-loader bucket. These images will include various materials and volumes and will be captured at the Volvo CE test facility. The dataset will be used to test an unsupervised machine-learning algorithm that extracts relevant information from images. The relevant information is the volume of the material in the bucket and the material characteristics. The different classes of characteristics will depend on the grit size.

This master thesis will focus on investigating and solving the following research questions:

1. What are the state-of-the-art (SoTA) related to the optical estimation of volume and classification of materials?
2. How can computer vision be utilised to identify the amount of material in a bucket on a wheel loader?
3. How can computer vision be utilised to classify the material in the bucket?

1.2 Purpose and Goals

This thesis aims to facilitate the physical excavation for an autonomous Volvo construction vehicle. This is done by optically sensing the environment, processing the data, and then calculating the fullness of material in the bucket and classifying the data based on material characteristics. The primary objective is to calculate the proportion of material within the bucket as a percentage of its total capacity. This thesis strives to develop a self-calibrating algorithm capable of adapting to buckets of various sizes. The second objective is to cluster the material into two classes: rocks and grit. To achieve this, the algorithm must possess the capability to distinguish the material from the bucket, ensuring that the clustering process focuses solely on relevant material data. In the context of an autonomous wheel loader, both the bucket volume and material information serve multiple purposes. For instance when the autonomous wheel loader engages in excavation tasks, the knowledge of the bucket's fill level becomes crucial in deciding whether the machine should execute a retake or proceed with dumping the material at the intended destination. Additionally, the information pertaining to the material inside the bucket plays a significant role in determining the appropriate dumping location or assessing its relevance. By analysing the material composition, the autonomous wheel loader can make informed decisions about where the material should be deposited or if it possesses any significance for further processing or handling. This allows for efficient and effective management of the excavated material.

1.3 Limitations

The scope of this thesis is focused on a dataset that comprises images captured under controlled and consistent lighting conditions. To ensure reliability and minimise the potential influence of external factors, all images in the dataset were collected on the same day in a riding stable. This approach mitigates any impact resulting from variations in weather conditions, such as rain, snow, fog, or frost. It is important to note that the algorithm developed in this thesis may possess the capability to account for these specific weather-related factors, However, it will not be tested in this thesis. Additionally, the dataset used in this research is specifically limited to a single type of bucket. This deliberate restriction ensures that the dimensions of the bucket are well-defined and consistent throughout the dataset. The algorithm will not be tailored to address the specific characteristics and dimensions of this particular bucket. The System is supposed to be dynamic where different buckets of different volumes can be used. The implementation, testing, and verification of the algorithm are conducted in an offline setting. This means that the algorithm's performance is assessed using pre-captured images and is not designed to perform real-time volume estimation or material clustering. The offline nature of the implementation allows for careful analysis, optimisation, and validation of the algorithm's effectiveness in accurately estimating volume and performing material clustering.

2. Background

This section aims to provide the reader with the necessary information and context to understand the research investigated. By providing this information, the background section aims to help the reader to understand the significance and relevance of the research being presented in this thesis.

2.1 Wheel Loaders

Wheel loaders are machines that are used in miscellaneous industrial and commercial applications. Some of the most common use of wheel loaders is in the mining, agriculture, and construction industry. The standard variant of the wheel loader has a mounted bucket at the front of the wheel loader and with the use of a hydraulic system, the bucket can scoop up different materials such as dirt, gravel, and rocks. In the front of the wheel loader, the cab for the operator is located while the engine is located at the rear. The operator can move and drive the wheel loader and also control the bucket from the cab. Due to the high variety of applications the standard bucket can be changed for various attachments [1]. See Figure 1 for an image of a Volvo Wheel-loader.

2.2 Computer Vision

Computer vision is a technical field that focuses on enabling computers to interpret and understand image data, video, and other types of visual data. It is a multidisciplinary field where algorithms and models are developed in order to perform tasks such as image recognition, object tracking, 3D reconstruction, and scene understanding. The field draws on concepts from mathematics, physics, and computer science [6]. Computer vision often relies on machine-learning techniques to analyse visual data and extract meaningful information from it. These algorithms are typically trained on large datasets with the purpose to recognise patterns and features in images and videos. This field of study has a wide range of applications in various fields, including security, healthcare, and vehicular automation. In security, computer vision may be used for surveillance to identify burglars. In healthcare, it can be used for imagery diagnosis. And in vehicular automation, computer vision may be used to enable self or semi-self driving vehicles [6].

Some key, sub-fields, techniques, and methods used in computer vision are:

1. **Feature Extraction**: This is a fundamental sub-field within computer vision where the goal is to identify distinctive features in images. This can later be used to identify patterns and thus objects.
2. **Object Detection**: This is a sub-field of computer vision where the goal is to identify specific objects within an image. Object detection algorithms often use feature extraction in combination with machine learning algorithms.
3. **3D Reconstruction**: This is a sub-field of computer vision where the goal is to reconstruct a 3D scene from multiple 2D images. Cameras with depth perception are often used.
4. **Image Processing**: Involves techniques to manipulate images with the goal to enhance useful information while redundant information is attenuated. The purpose is to improve image quality with techniques such as filtering.

2.3 Supervised Learning & Unsupervised Learning

Supervised learning is a subcategory of machine learning that requires labelled data during the training, testing, and validation stages. A supervised machine learning algorithm learns patterns and makes predictions based on the provided training data and its associated labels. Commonly by utilising cross-validation techniques, the input data is partitioned into multiple subsets, ensuring a robust evaluation of the algorithm's performance. The primary goal for supervised learning algorithms is to accurately predict the correct labels for new, unseen data based on its previous training phase. The training phase is constructed to optimise the algorithm by adjusting its weights based on the accuracy of its predictions. The iterative process of adjusting the weights continues

until the algorithm achieves a suitable level of accuracy and generalisation. The iterative process of training a supervised machine-learning algorithm aims to find a balance between overfitting and underfitting. If the algorithm performs well on the training data but fails to generalise well to the validation data, the algorithm is overfitted. This occurs when the algorithm is excessively complex and customised to the training data. If the algorithm on the other hand is too simplistic and fails to capture the underlying patterns in the data, the algorithm is underfitted [7],[8]

Another subcategory of machine learning is unsupervised learning. These algorithms do not require labelled training data. They instead find patterns, structures, or clusters in the data without explicit guidance. Unsupervised learning is particularly useful on large and complex datasets where manual annotation is time-consuming or impossible. It is a powerful tool for data analysis and pattern recognition. It may serve a crucial role in the extraction of meaningful and useful information from unstructured and unlabelled data. The performance of an unsupervised algorithm is usually evaluated by metrics such as accuracy, reconstruction error, or visual inspection of the results [7],[8].

2.4 Stereo Vision

A simple stereo system resembles the human perception of depth, our brain is feed information from both our eyes and from that information, the distance to an object is calculated. This is called binocular vision, if we close one eye our depth perception will be affected. Similarly, by using two cameras with a known horizontal relation to each other, it is possible to calculate the depth of individual pixels using triangulation. The fundamental idea is to find the corresponding point in the two images taken from different sources and from there calculate the pixel offset [9]. With a known pixel offset and camera baseline distance and camera angle, it is possible to triangulate the depth of an object that matches the pixel. The pixel offset is referred to as the disparity between points and is defined as the difference in their horizontal pixel locations. The depth is inversely proportional to the disparity. First, a matching scene point must be found in both images, these are the corresponding points. This process is called stereo correspondence and will be further elaborated in section 2.4.1. See 1 for the equation to calculate the depth of the pixel [9].

$$Z = \frac{Bf_x}{(u_l - u_r)} \tag{1}$$

The pixel location of the points is (u_l, v_l) for the left camera and (u_r, v_r) for the right camera. Where u is the horizontal coordinate and v is the vertical coordinate. $(u_l - u_r)$ is the disparity. Z is the depth and the baseline displacement of the two cameras is the variable B [9]. f_x is the focal length of the lens. Figure 2 demonstrates the relation for all variables.

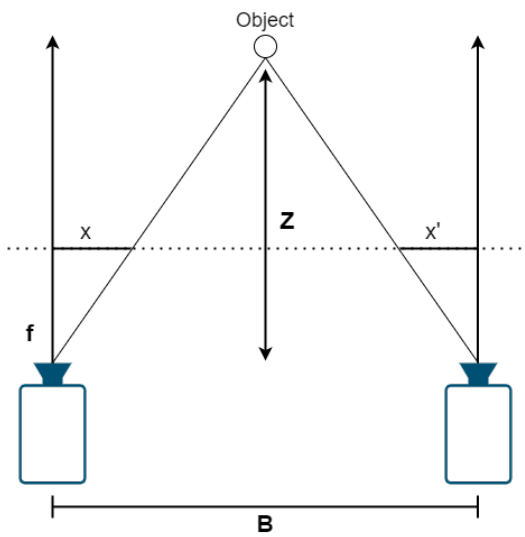

Figure 2: Simple stereo system with two cameras and one object [9].

A large disparity between the pixels corresponds to a short distance to the object and a small disparity corresponds to a long distance. It is important to understand that a longer baseline will give better depth perception if both (u_l, v_l) and (u_r, v_r) exist. If the cameras are placed too far apart in relation to the object, one or both cameras may lose the object/pixel outside the frame. If the object/pixel is outside of one of the frames, stereo matching, and depth calculation will be impossible. Thus it is important to place the cameras in a good position in relation to the object.

2.4.1 Stereo Correspondence

Stereo correspondence is the process of finding corresponding pixels in two or more images taken from different viewpoints, this process is also referred to as stereo matching. As mentioned in the section above, the corresponding pixels need to be matched for a stereo-vision system to calculate the depth to an object in an image [9]. A variety of algorithms may be used for stereo matching, however, they can be grouped into two categories: local methods and global methods.

Local methods, also known as feature-based methods, divide both images into local regions and extract features from these regions. The features are then mapped to establish correspondence. Some traditional feature-based methods are Scale-Invariant Feature Transform (SIFT), Speed Up Robust Features (SURF), and Harris corner detector. Global methods, also known as dense methods compare all pixels in both images. They use a cost function to evaluate the similarity between pixels and from there match pixels from both images. A popular global algorithm is called Block Matching. Block Matching divides an image into overlapping blocks and calculates the similarity between blocks in the images [9]. Both methods mentioned have advantages and disadvantages. Local methods are known to be fast but sensitive to changes in viewpoint, lighting, and scale. While Global methods are robust to these mentioned changes but it is computationally expensive.

2.5 Superpixel Segmentation

Superpixel segmentation is a popular algorithm in the field of computer vision. The algorithm distributes an $N \times M$ image into a subset of regions, where each region is referred to as a superpixel. The boundary of each superpixel is determined by the colour of the pixels in proximity. Thus sharp edges in the image caused by sudden colour changes will tend to be boundary points for the superpixels.

2.5.1 Simple Linear Iterative Clustering for Superpixel Segmentation

There exist various algorithms to generate superpixels for an image, one of the fastest and more common is Simple Linear Iterative Clustering (SLIC) Superpixel Segmentation [10]. The SLIC algorithm takes two inputs, an $N \times M$ image and a K parameter which represents the number of superpixels the image will be divided into. First, the SLIC algorithm will place the center of each superpixel cluster $C_i = [l_i, a_i, b_i, x_i, y_i]$ in a regular grid pattern with a distance of S pixels apart where $S = \sqrt{\frac{N}{k}}$ and N is the number of pixels in the image [10]. Secondly, all centers will be placed in a seed location corresponding to the lowest gradient position in a 3×3 neighbourhood. This step is performed to mitigate the risk of placing a center on an edge or seeding a superpixel with a noise pixel. The third step is known as the assignment step. In this step, all pixels are associated with one cluster center. The SLIC algorithm only evaluates the pixels within a certain distance from each center pixel. This reduces the number of pixel associations that need to be calculated [10]. Each superpixel is approximately $S \times S$ and the region evaluated for each center is $2S \times 2S$. The center is updated for all superpixels once all pixels are assigned to a superpixel. One of the major advantages of the SLIC algorithm is that it can find features in the CIELAB space. Once the superpixel regions are determined, the mean RGB value is calculated and assigned for all superpixels individually [10]. Figure 3 shows the input and output of the superpixel segmentation algorithm.

Figure 3: The first image is a standard image before processing. The second image is the representation of the superpixel regions which are closed and bounded. The third image is the result when each superpixel is assigned its mean RGB value. The total number of pixels for this image is 423×640 and the total number of superpixels are 200. Visually a portion of the edges are saved. Image by Frauke Riether.

2.5.2 Threshold Based Classification for Superpixel Segments

There are numerous methods for extracting information from an image after segmentation. The method used for information extraction is primarily dependent on the developer's prior knowledge of the dataset. For example, if the developer knows that the dataset will contain images of a homogeneous blue sky and blackbirds a simple classification strategy is to assign each superpixel as the background sky or as birds dependent on colour. This is known as threshold-based segmentation. This simple strategy does not take spatial dependence into account which makes its usefulness limited to images with distinct colour discrepancies [11].

2.6 K-Means Clustering

K-means clustering is a popular unsupervised algorithm. It clusters data points into k clusters where each cluster is defined relative to a centroid. When the algorithm is finished, each centroid will be the center of a cluster. The number of clusters k is pre-determined by the user [5],[12].

1. The first step of k-means clustering is initiation. There are many different strategies for this step, however, the most intuitive is a random placement of the centroids.
2. The second step is to assign the nearest data points to each cluster.
3. The third step is to recalculate the centroid by calculating the mean position of the cluster.
4. The second and third step is repeated until the centroid position stops changing.

For a dataset containing Red-Green-Blue pixels, each pixel can be represented by a 3-dimensional point [Red value, Green value, Blue value]. Thus the k-means clustering algorithm when clustering RGB images will account for a 3-dim space [5],[12]. Figure 4 shows the input and output of the K-means clustering algorithm.

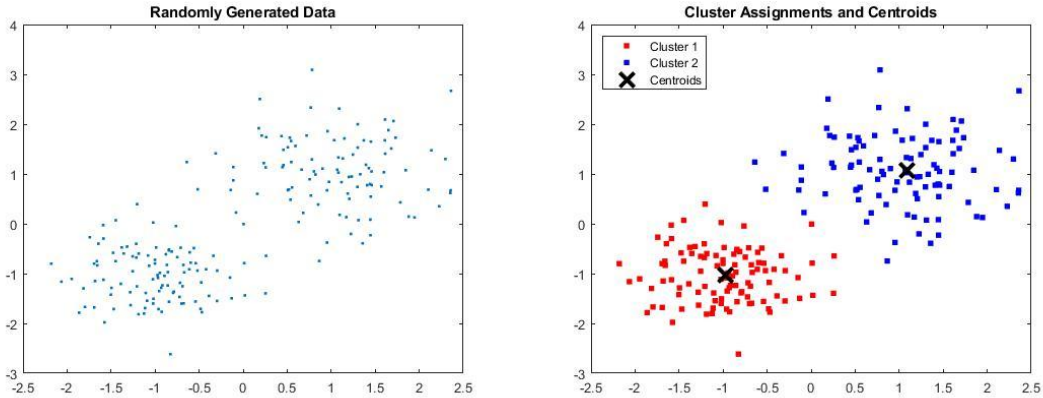

Figure 4: The left image shows a randomly generated two-dimensional dataset. The right image is the same dataset clustered into two clusters ($k = 2$).

2.7 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a class of artificial neural networks that are commonly used for feature extraction in signals and images. It is a deep learning strategy to learn spatial hierarchies of features from input data [13].

2.7.1 Convolution

A CNN works by convolving the inputs of a neural network with filters. A filter is a small matrix used to measure how similar a region of the input matrix is compared to a feature. A feature may be any prominent attribute such as a straight vertical line, arc, or horizontal edge. The filter will then slide across the input matrix and map similar edges. A filter may also be called a kernel [13]. An RGB image consists of three matrices of data, so the convolution requires a three-layered matrix filter. The convolution of the input with multiple filters extracts a broader range of features. For instance, a convolutional neural network can find the horizontal and vertical lines and edges of a house by using multiple filters [13].

2.7.2 Padding

When a $N \times M$ matrix is convolved with an $f \times f$ filter the size of the output matrix will always be $(N-f+1) \times (M-f+1)$. This means that the output matrix will always be smaller than the input matrix if the filter is 2×2 or larger [13]. This results in a loss of information which is a downside of convolution. In order to solve this problem, padding may be used. Padding is a technique to add, often white pixels, to the edges of the image to make up for the loss after convolution [13].

2.7.3 Activation Function

An activation function in a neural network is a function that maps an input between a desired interval. The optimal interval depends on the system. The most common activation function for CNN is a Rectified Linear Unit (ReLU) activation function [13]. The ReLU function can be seen in equation 2. It is a linear function that maps all negative inputs to zero.

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{2}$$

Another common activation function used in neural networks is the Sigmoid activation function [13]. It is a non-linear activation function that produces an output between 0 and 1. The sigmoid activation function can be seen in equation 3.

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{3}$$

2.7.4 Pooling Layer

After an individual convolutional layer and padding, it is common for a CNN to use Max Pooling. Max pooling is an operation that takes a $P \times P$ matrix and maps the max integer to a new matrix. The new matrix has reduced dimensions thus the computational cost for operations on the matrix will be reduced, this is also referred to as downsampling. Max pooling tends to retain sharp edges and features while it removes redundant frames [13]. This makes it a popular pooling algorithm for CNN since CNN primarily identifies features in a dataset by the edges.

2.8 Dense Extreme Inception Network for Edge Detection

Dense Extreme Inception Network for Edge Detection (DexiNed) is a CNN based edge detecting network. The DexiNed has a single training stage and is trained from scratch without the need of pre-trained weights [14]. Different filters are stacked together in order to predict an edge-map with the same resolution as the input image to the DexiNed. DexiNed can be seen as the merging between the two sub networks Dense extreme inception network (Dexi) and the up-sampling block (UB). The two sub networks handle different parts where the Dexi takes the RGB image and the UB the feature maps from each Dexi which results in the DexiNed to generate a thin edge-map [14].

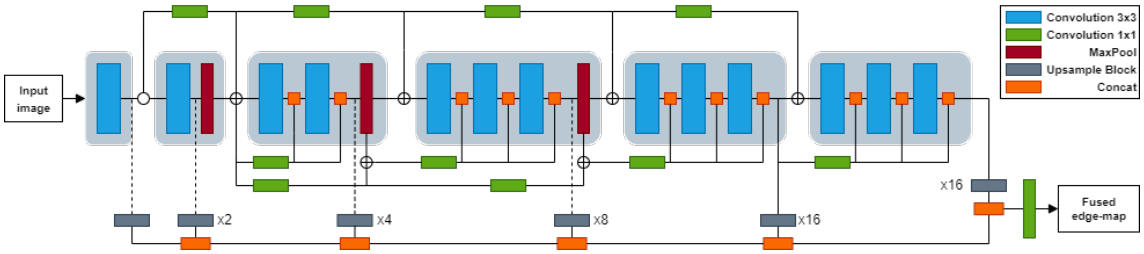

Figure 5: Architecture for DexiNed. The DexiNed includes an encoder which is made up by six main blocks. Each main block is composed by sub blocks where the sub blocks is connected to the output from the previous main block. Through 1x1 convolution the main blocks are linked to each other where the output from the main block is fed into an Upsample Block (showed in dark grey in the figure), the Upsample Block creates an intermediate edge-map to be able to build a Scale Space Volume in order to make a final fused edge-map.

2.9 Adaptive Histogram Equalization

The Adaptive Histogram Equalization (AHE) is a method for contrast enhancement which is proven to be effective and were designed in order to be widely applicable. Some drawbacks of the method are the ability to handle homogeneous regions, where the AHE overenhances the noise in these regions as well as the AHE being slow [15].

2.10 Contrast Limited Adaptive Histogram Equalization

AHE is proven to be effective in contrast enhancing, however as described in 2.9 the noise of the image is being enhanced as well, in some applications the enhancement of the noise can affect the result in a negative manner [15]. One example is for contrast enhancement of images with homogeneous regions, where the enhancement of noise can be disturbing. The limitation of contrast enhancement by the Contrast Limited Adaptive Histogram Equalization (CLAHE) can in these examples give a more desirable result [15].

2.11 Contour Detection

Contours refer to the boundaries or shapes of objects in an image and can be seen as the curve with all the points along the boundary of an object. In image processing and computer vision, the contours are commonly used for object detection, object recognition, and shape analysis [16]. Contour following operations involves tracing the outline boundary of an object by identifying a starting pixel on the edge of the contour and from that starting point following the contour or edge of the object by iterative inspecting the neighboured pixels of the contour pixel until the contour of the bounded area is closed.

2.12 Morphological Image Processing

Morphological image processing is a broad set of image processing techniques that uses the shape and spatial relationship between objects in images. The approach has a wide range of applications, with the most common being image segmentation, noise reduction and feature extraction. By applying a structuring element to probe an input image, the value of each pixel in the output image is based on the comparison of the same pixel in the input image and its neighborhood pixels, where the output image has the same size as the input image [17]. The approach can be applied to both binary and grey-scale images. The operations erosion and dilation are the most common morphological operations, where erosion removes image pixels on the object boundaries and dilation adds pixels on the object boundaries. The shape and size of the structuring element decide how many pixels that is removed or added to the object in the image. The two morphological operations: opening and closing the merging of the previous mention operations. Opening first performs erosion followed by dilation which generally smoothes the contour of an object and breaks narrow interconnections. Closing is the opposite order, in other words, dilation followed by erosion which generally fuses narrow interconnections and fills small gaps in the objects [18].

2.13 Median Filter

A median filter is a digital signal processing technique that is commonly used to remove noise from images. The filter works by replacing each pixel in the input signal with the median of neighbouring pixels. The filter works by sliding a window of size $M \times N$, also called a kernel across the input image. Then it calculates the median value of all pixels within the window and outputs it to the center pixel of the window [19]. The median filter serves as a low-pass filter since it removes impulse noise and low-frequency components, which are caused by sudden spikes or drops in signal intensity. The median filter is also computationally efficient. A potential drawback with the median filter is that it can smooth out details and edges in the input image or signal. The window size of the filter directly affects the performance of the filter. A smaller window size preserves features in the image to a higher degree than a larger window size, however, its ability to attenuate and suppress undesirable noise is lower. In the case of a larger window, noise suppression potential is high but limited to image content preservation [19].

2.14 Sklanskys Algorithm for Convex Hull

A convex hull is a polygon that contains all points in a given plane or coordinate system. It can be described as the smallest shape possible to enclose a set of points without any indentations or protrusions on its edges.

The Sklansky algorithm [20] is an algorithm that finds a convex hull for a set of 2D points. The algorithm was proposed by Jack Sklansky in 1982 and works by iterative adding new points to the convex hull. It is popular for two-dimensional datasets since its complexity of $O(nlogn)$. The algorithm creates a convex hull by incrementally adding points one at a time. The algorithm achieves a log-linear time complexity because each point is processed once and only once. The algorithm can be separated into two parts. In part one, the algorithm first identifies the extreme vertices in the horizontal and vertical directions and creates the square S. All points of the simple polygon are within the square S. Then four triangular regions R_i are created. The idea is to create a monotone chain within all four regions R_i. To create a monotone region, part 2 has to be examined. For this example, we will start at R_1. Suppose that two vertices I & $I + 1$ have

been accepted onto the monotone chain which starts at L. Separate the plane into three regions as shown in Figure 6. The next vertex, $I+2$ is handled as the pseudo-code below:

1. If $I+2$ is in $R_1 \longrightarrow$ discard $I+2$
2. Else If $I+2$ is in $A_3 \longrightarrow$ retain $I+2$ and re-iterate
3. Else If $I+2$ is in $A_2 \vee I+2$ is in $A_1 \wedge I+2$ is above $I+1 \wedge$ a vertex was discarded due to line 3 on the immediately preceding iteration $\longrightarrow$ discard $I+2$
4. Else If $I+2$ is in $A_1 \longrightarrow$ discard $I+1$

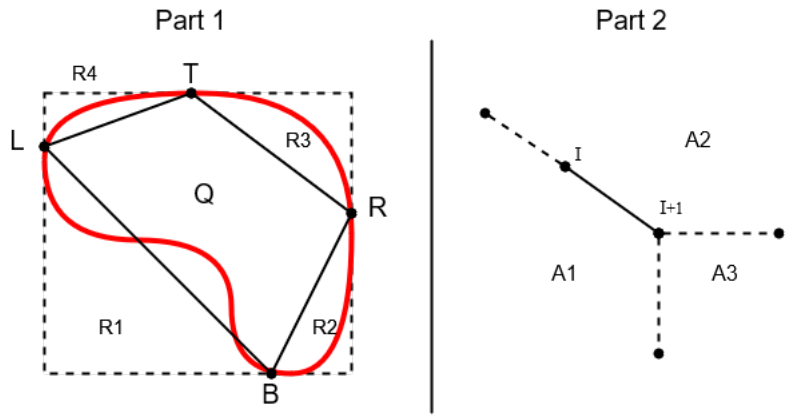

Figure 6: To the left, the image shows Part1 of Sklanskys algorithm and to the rights the image shows Part 2 of the algorithm.

2.15 Green's Theorem

Greens' theorem [21] relates the line integral of a vector field $\vec{F} =< F_1, F_2 >$ around a closed curve C to a double integral over the region D which is bounded by C. Green's theorem is stated in equation 4:

$$\int_C \vec{F} \cdot d\vec{r} = \iint_D \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right) dx\, dy \tag{4}$$

The theorem applies that the curve runs counterclockwise i.e. positively oriented, which can be seen in Figure 7. For curves that run clockwise, i.e. negatively oriented the Greens' theorem can still be applied but with a negative sign.

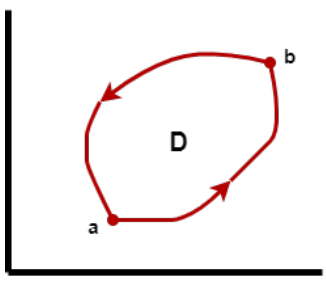

Figure 7: The region D bounded by the closed curve.

The area A of the bounded region D in the linear space can be calculated using equation 5.

$$A = \iint_D dA. \tag{5}$$

3. Related Work

This section will provide context and background information to the research investigated for this thesis. It will demonstrate the current state of the field and the gap in knowledge that this thesis aims to address. Additionally, this section will establish the credibility of the research conducted in this thesis by critically evaluating the relevant literature.

3.1 Superpixel Segmentation

G. Xiaojin et al. [22] implemented an algorithm to detect rocks on the moon. They aimed to use computer vision to help an autonomous rover recognise and avoid rocks as it travels around the moon. To do this, they used a superpixel segmentation algorithm to fragment the image into regions of interest and a threshold-based approach to classify rock segments as foreground and other segments as background. All images had a grey-scale resolution of 512×512, and the algorithm fragmented the images into 500 superpixels. The authors concluded that the superpixel segmentation accurately captured most rock boundaries. They used an energy function to label each superpixel $S = \{s_1, .., s_N\}$ into the labels foreground and background. They evaluated the algorithm using a validation dataset and compared it to manual labelling. The evaluation used three criteria: Region labelling accuracy, boundary localisation accuracy, and Rock detection accuracy. Region labelling accuracy measured the percentage of correctly labelled rock pixels compared to the validation data image, and the algorithm achieved a labelling accuracy of 81.95%. Boundary localisation accuracy evaluated the accuracy of localizing the rock boundaries by averaging the minimum distance for each boundary pixel, with an average localisation error of 13.34 pixels. Rock detection accuracy evaluated the system's ability to identify the number of rocks by comparing the number of false positives and negatives to the true number.

Y. Ren et al. [23] compared a traditional segmentation method to superpixel segmentation in their paper to correctly segment oolites from carbonated rock. They used a traditional image segmentation algorithm, which consists of three steps: the extraction of connected areas of carbonate rocks, k-means clustering, and binary image processing and contour extraction. For the superpixel segmentation, they used the SLIC algorithm [10] and divided the image into 400 superpixels. The image used for evaluation had a resolution of 781×468.

	AP	Precision	Recall	IoU
Traditional	0.83	**0.93**	0.76	0.72
Superpixel	**0.86**	0.84	**0.94**	**0.80**

Table 1: The results of each evaluation index of the two methods [23].

The researchers in [23] compared the SLIC superpixel segmentation algorithm to the traditional segmentation algorithm and concluded that the former was somewhat better. The results of [23] can be seen in table 1.

R. Achanta et al. [10] developed the SLIC algorithm and compared it to multiple segmentation algorithms. The algorithms that they compared their SLIC algorithm to were four state-of-the-art Graph-based algorithms and four Gradient-ascent-based algorithms. They also described in great detail how the algorithm works. The evaluation was performed on a dataset provided by Berkeley [24]. The dataset contains approximately three-hundred 321×481 images and 10 ground truth segmentation for each image that has been human-annotated [10]. The authors concluded that The SLIC algorithm has a complexity of $O(nlogn)$ which makes it the fastest of all superpixel algorithms. It is also the algorithm with the highest memory efficiency. SLIC only requires N floats to store the distance from each pixel to its nearest cluster center [10]. Compared to the second-best algorithm which has a memory efficiency of $5N$.

3.2 DexiNed

From a paper by Xavier Soria et al [14], a deep learning edge detection for thin edge maps is proposed and referred to as DexiNed. A comparison between different state-of-the-art CNN edge detection algorithms is implemented, and the paper concludes that DexiNed is superior in thin edge-map edge detection. The comparison from the paper[14], can be seen in Figure 8 where the comparison is conducted on the Berkeley Segmentation Dataset 500 (BSDS500) which is the standard benchmark for edge detection and Barcelona Images for Perceptual Edge Detection (BIPED) dataset. The BSDS500 datset is divided into 200 images for training, another 200 images for testing and 100 for validation [25]. The BIPED dataset comprises 250 high-definition outdoor images with dimensions of 1280x720 pixels for each image. The dataset is publicly available as a benchmark for evaluating edge detection algorithms and provides 200 images for training and 50 images for testing, with the split already prearranged [14].

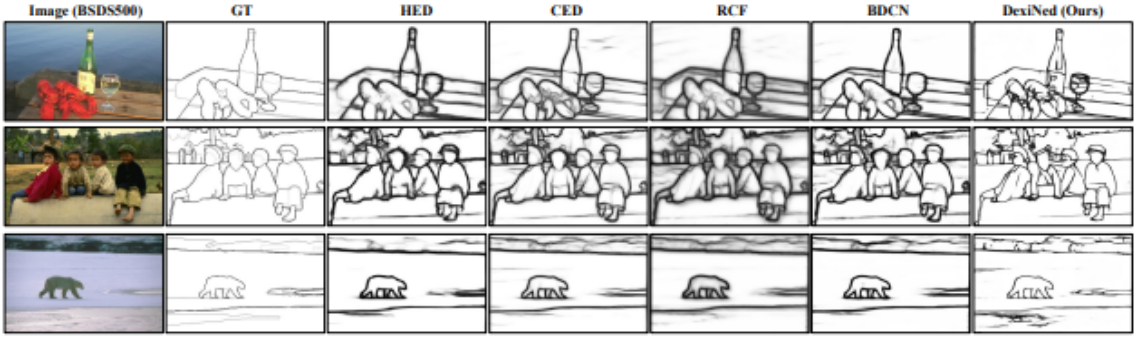

Figure 8: Figure from the paper [14]. The paper compares different state-of-the-art algorithms used on three BSDS500 images and the DexiNed used on three BIPED images.

Haijie Li et al. [26] used a modified DexiNed to estimate the particle size of rocks to affect the crusher's efficiency in a crushing plant. The work from the paper is divided into four different parts, the first part being preprocessing the image with grey scaling the image and then applying CLAHE, due to the edge features of rocks being very similar. In the second part of the [26] paper, they performed a modified DexiNed. Instead of using the six main blocks as in the original DexiNed which can be seen in Figure 5, they used a DexiNed consisting of five main blocks on the preprocessed image to avoid loss of information in the final fused edge-map. In the third part of the work from [26], they performed contour segmentation and filtering by applying a binary filter on the fused edge-map from the DexiNed followed by the morphological operation applied on the binary image. The built-in OpenCV function `findContours()` was used to determine the contours to perform contour segmentation and filtering later. The last part of the [26] paper is the distribution calculations. By calibrating the area of the contours they later estimated the size and mass of the materials. The paper proves that the demonstrated method which includes the DexiNed can detect the edges of rocks and distinguish rocks from nearby fine regions[26] as well as estimate the size of the materials from the pixel areas.

3.3 Distance & Volume Measurement using Stereo Camera

In a paper written by Mizuchi et al [27], they propose a distance measurement system using a stereo camera. The paper is for automatic berthing control for ships with the goal to measure the distance between the ship and an onshore/offshore target. The method combined object detection and the triangulation method in order to get the calculated distance to the object. For object detection, the paper used Vector Code Correlation (VCC), which is used for template matching and is a correlation calculation algorithm that finds the region with the maximum correlation in images [28]. The proposed measurement system from the paper [27] had an accuracy within $1m$ at a range between $20 - 100m$.

N. Bándi et al. [29] propose a method to create a dense point cloud of a 3-dimensional object with multiple images from different angles. Their proposed method can be divided into four steps. The first step is to estimate the camera position based on the feature points found on the images using a technique known as bundle adjustment [30]. The second step is to use the camera position to generate a dense cloud onto which a Poisson surface and convex hull are fit. The third step is to determine the intersection of the surfaces. And for the fourth and last step, the volume of the closed and bounded body can be calculated and converted to metric scale. [29] concluded that the method is robust and performs a 90% volume estimation accuracy for high-quality objects if the algorithm is given an adequate number of images and resolution. The number of images needed to accurately estimate the volume of a dense 3D model is approximately 30 for their method. With images of adequate number and resolution the average compiling time is 221 seconds. They identify that their solution can be improved with object recognition and segmentation to exclude redundant information in the background. This is to robustly estimate the volume in the case of an image with a complex background.

4. Method

The research method provides a systematic approach to investigating a problem or research question. The method is used to gather and analyse data and to draw conclusions or make recommendations based on that data. There are numerous research methods available, such as quantitative methods, qualitative methods, system development research, and others. The research methodology chosen is determined by the problem and the research questions. The research method chosen helps to ensure that the research is conducted in a rigorous and systematic manner. It also contributes to the reliability and validity of the results.

4.1 System Development Research Methodology

This thesis will apply a system development research method approach in order to answer the research questions. The method is inspired by an iterative approach from Nunamaker and Chen [31], see Figure 9.

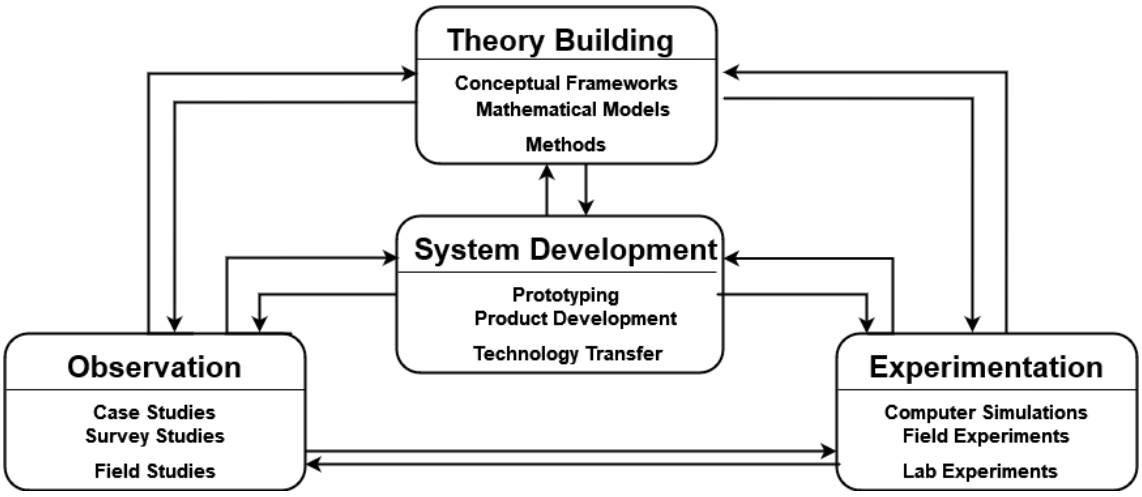

Figure 9: Flowchart of a multi-methodological approach to Is research [31]

1. `Theory Building` refers to constructing a conceptual framework that helps explain and understand the investigated problem. The process typically begins with a review of existing research and literature in the relevant fields. The goal is to find gaps in the current knowledge and determine the key research questions that need to be addressed. Secondly, a hypothesis is typically formulated based on the research, existing theories, and observations. From here, mathematical models are formulated and a research method is chosen.

2. `Observation` refers to the validation of the research results. From the observations, new theories and models may be proposed. To ensure consistency and accuracy, the observation should be conducted in a systematic and organised manner, using standardised procedures and protocols.

3. `Experimentation` is the stage of the research where the developed system is tested. The experiment should be designed so that the hypothesis and the theories are tested by gathering empirical evidence. Experimentation is a critical component of this scientific method since it provides a rigorous and systematic approach to testing the hypothesis and gathering empirical evidence.

4. `System Development` is the stage where the system is developed. The system is designed and implemented in order to meet a specific set of requirements. These requirements are set so that the research questions may be answered. During the system development prototyping and testing with the goal to refine the design is common.

As previously mentioned, the System Development Research methodology is iterative. This method is popular in scientific fields where the problem and solution are complex, and where the optimal solution cannot be easily determined in a single step. The process allows the engineer to iteratively repeat steps or backtrack. The iterative method provides a flexible and adaptable approach as each iteration may integrate new information and insights obtained from the preceding iterations.

4.2 Process of Systems Development Research Methodology

The Research Process is the application of the scientific research method. In this case that is the Systems Development Research Methodology. The application of system development as a research methodology in IS should accommodate all five steps seen in Figure 10. This process is also derived from Nunamaker and Chen [31].

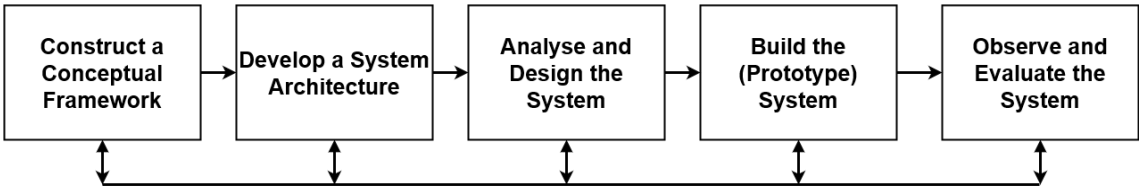

Figure 10: A Research Process of Systems Development Research Methodology [31]

The conceptual framework contributes meaningful research questions, requirements for the system's functionality, and knowledge regarding the relevant literature. A conceptual framework assists researchers in developing a more focused and rigorous approach to their research, as well as ensuring that their findings are based on a well-defined theoretical and conceptual framework. The development of the system architecture is the design and structure of the system, including all components, interfaces, and interactions. It includes setting requirements, both for the users and the stakeholders. When analysing and designing the system the focus should be to acquire a knowledge and database. It may be relevant to develop alternative solutions and then choose one. It is important to gain knowledge and insights about the complexity of the system. Building a prototype system may contribute to the developer's knowledge about the concepts, framework, and design through the development process. By observing and evaluating the system its possible to develop new theories and models.

4.3 Software

For this project, `Python 3.10`[1] is used as the main development programming language. `Matlab 2022a`[2] is used for plotting graphs and for data analysis. In Python, the `OpenCV`[3] computer vision and machine learning software library is used for image manipulation and image processing. `scikit-learn`[4], also known as sklearn is an open-source machine learning library for the Python programming language that is used for its clustering algorithms. `scikit-image`[5], also known as skimage is an open-source Python library for image processing and computer vision tasks, it is used for its segmentation algorithms. For data acquisition, `depthAI`[6] is used to configure the hardware, this is further elaborated in section 6.3. `DexiNed`[7] is an algorithm that is used for edge detection. In order for DexiNed to run in Python, `Kornia`[8], `PyTorch`[9], `Matplotlib`[10], `OpenCV`, and `Python 3.7` or newer are needed.

[1] https://www.python.org/
[2] https://se.mathworks.com/products/matlab.html
[3] https://opencv.org/
[4] https://scikit-learn.org/
[5] https://scikit-image.org/
[6] https://docs.luxonis.com/en/
[7] https://github.com/xavysp/DexiNed
[8] https://kornia.github.io/
[9] https://pytorch.org/
[10] https://matplotlib.org/

4.4 Hardware

This section accounts for all relevant hardware used in this thesis.

4.4.1 OAK-D PoE

For this paper, OAK-D PoE will be used as the optical sensor in order to collect data for the dataset. The OAK-D is supplied with three onboard cameras, one IMX378 camera, and one stereo pair including two OV9282 cameras [32]. The IMX378 camera has a resolution of 12 Mega Pixels (MP) (4032x3040) and a maximum framerate of 60 Frames per second (FPS) whiles the OV9282 cameras have a resolution of 1MP (1280x800) and a maximum framerate of 120FPS. For the power supply and communication, the OAK-D-PoE uses Power-over-Ethernet (PoE). The device offers 802.3af PoE[32] and in order to supply the device with power a PoE injector/switch [33] is needed. OAK-D-PoE is equipped with the OAK-SoM-Pro[34] making it possible to do depth calculations and AI processing on the OAK-D-PoE. The data from the camera is output to the host via the 1000BASE-T ethernet connection[ref]. The enclosure for the device is IP67 rated [35], making it completely protected against dust and protected from water immersion between 15cm and 1m in depth [35] making it possible to mount the camera onto the wheel loader and leave it outside regardless of the weather. The OAK-SoM-Pro is a System Module (SoM) which is integrated into systems to apply real-time AI and depth perception. All the signal I/O between the SoM and the system interfaces through two 10-Gbps-rated 100-pin board-to-board connectors. The OAK-D-PoE will be used while creating the dataset for this project. The stereo depth from the device will be a part of the volume estimation of the material in the bucket.

4.4.2 Computer

The processor in the computer used for testing and validating the system in this thesis is an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 2.00 GHz. The device has a 16GB RAM-memory. The operating system is a 64-bit Windows 10.

5. Ethical Considerations

All data and developed software belongs to Volvo CE. The algorithm originates from the authors. No information regarding confidential data nor technology is shared in this thesis. The stereo camera used in this thesis is not a confidential device, however, it belongs to Volvo CE. No confidential software is used in the development of the system.

6. Implementation

The implementation section describes the practical application of the research conducted in this thesis. This section outlines the steps taken to design and develop an experimental software prototype that is used to evaluate the hypothesis and solve the thesis research problem. The purpose of the implementation section is to provide the reader with a clear and concise understanding of how the research is carried out. It includes a detailed description of the methodology used to implement the research including the design and development of the experimental setup and software.

6.1 System Overview

This section will provide an overview of the implementation conducted in this thesis. Figure 11 shows the step-by-step overview of the proposed algorithm.

The algorithm implemented in this thesis will take a raw RGB image and its corresponding stereo depth as inputs, this is further elaborated in section 6.2, Software for Data Acquisition. For the first step, the algorithm will extract the bucket and remove all the surrounding pixels. This is thoroughly described in section 6.4, Bucket Extraction. From the information obtained from the bucket extraction, the system can now calculate the degree of fullness of the material in the bucket by calculating the volume as described in section 6.7, Volume Estimation. After the bucket extraction, the material is extracted from the bucket as described in section 6.5, Material Extraction. This is done to facilitate the material classification described in section 6.6, Material Classification.

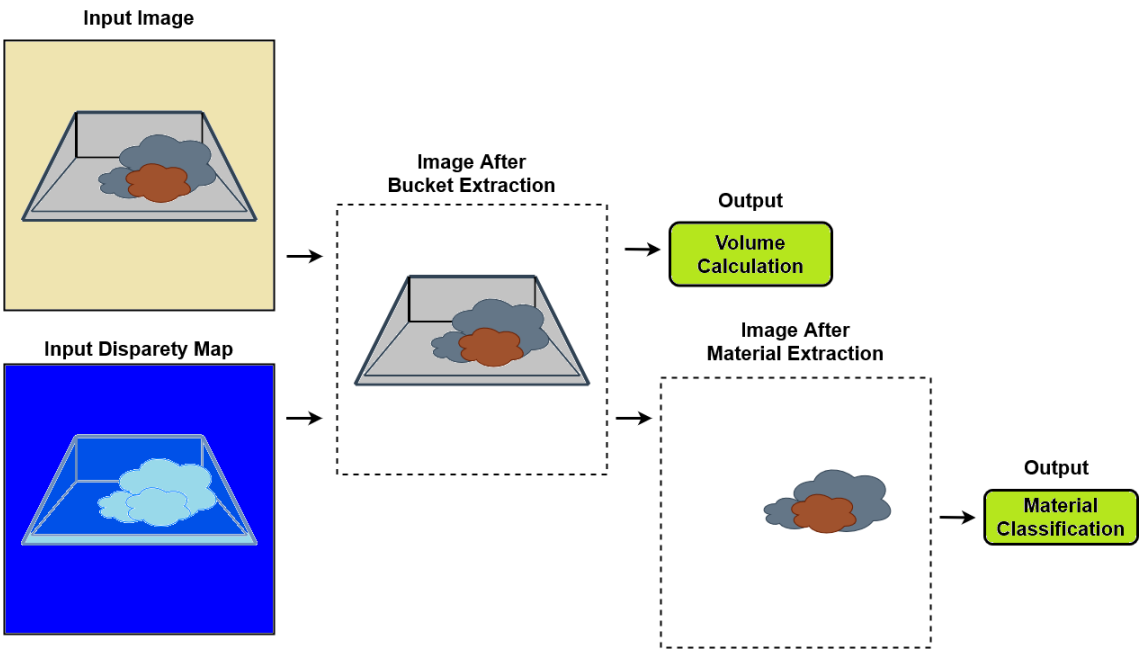

Figure 11: The Input Image is the raw RGB image containing both the relevant and irrelevant information i.e. the bucket with the material and the redundant data which is all the pixels outside of the bucket edge. The Input Disparity Map is an image that gives a visual representation of the depth to objects in an image. The first step of the system is to extract the bucket from the redundant background. When the bucket is extracted, the system can use that information to calculate the volume with stereo-depth data. The system also has all the information necessary to separate the material from the bucket, this process is referred to as material extraction. When the material is extracted, classification can be done.

6.2 Software for Data Acquisition

For data acquisition, this project has used the DepthAI API[11] which is an open-source Python library designed to simplify the development of computer vision and AI applications. The API includes a wide range of features, including support for stereo depth estimation, object detection and tracking, semantic segmentation, and more. It also supports a variety of devices, including the OAK-D PoE, described in 4.4.1, Hardware. The DepthAI API supports running multiple AI models, filters, and other functionalities simultaneously on the same device. This is achieved through pipelines. A pipeline is a collection of nodes, this means that the developer can chain together different nodes in order to access and run a variety of functionalities. Each node provides a specific functionality such as ColorCamera, StereoDepth, and more. The nodes communicate with each other and the host computer by sending messages through XLink. Another key feature of the DepthAI API is that it supports real-time processing and streaming of images. It includes tools for capturing and processing image data from the hardware device, as well as tools for streaming the data over a network. Algorithm 1 gives the pseudo-code for the data acquisition software implemented in this thesis.

Algorithm 1 The Data Acquisition Software

```
Initialise Hardware & Firmware  Initialise Queues
while True do
    image ← readQueue()
    if image is RGB then
        Display(image.RGB);
    else
        Display(image.Depth);
    end
    if Enter is pressed then
        Save(image.RGB) to file  Save(image.depth) to file  Save(depth.data) to file
    end
end
```

The configuration of the hardware and firmware can be seen in Figure 12. The left and right mono cameras are enabled and their output images are sent to the StereoDepth node. The StereoDepth node is configured to calculate the disparity map and depth (mm). The pipeline is also populated by the standard colour camera that outputs an RGB image stream to the host computer.

The colour camera is configured to output images of resolution $1080p$ and the mono cameras outputs images of resolution $720p$. The disparity image is filtered with a 5×5 median filter. No minimum or maximum range for distance calculation is implemented.

6.3 Data Acquisition

The quality of the data is essential for the algorithm constructed in this thesis. It is desirable to have RGB images and depth maps that accurately represent the bucket and the material within the bucket. The data needs to be reliable which means that the images and depth maps are consistent and can be replicated. This is because unreliable data can lead to erroneous conclusions and faulty designs. The data needs to include all relevant information. If the data is incomplete, the system may not be able to function as intended and the results will be impaired. While collecting data, numerous issues were found and solved. This section will account for how the data was collected and how the strategy behind data collection changed over time.

As concluded by Mizuchi et al. the camera position relative to the object is essential for accurate distance measurements when capturing images with a stereo camera.

[11] https://docs.luxonis.com/projects/api/en/latest/

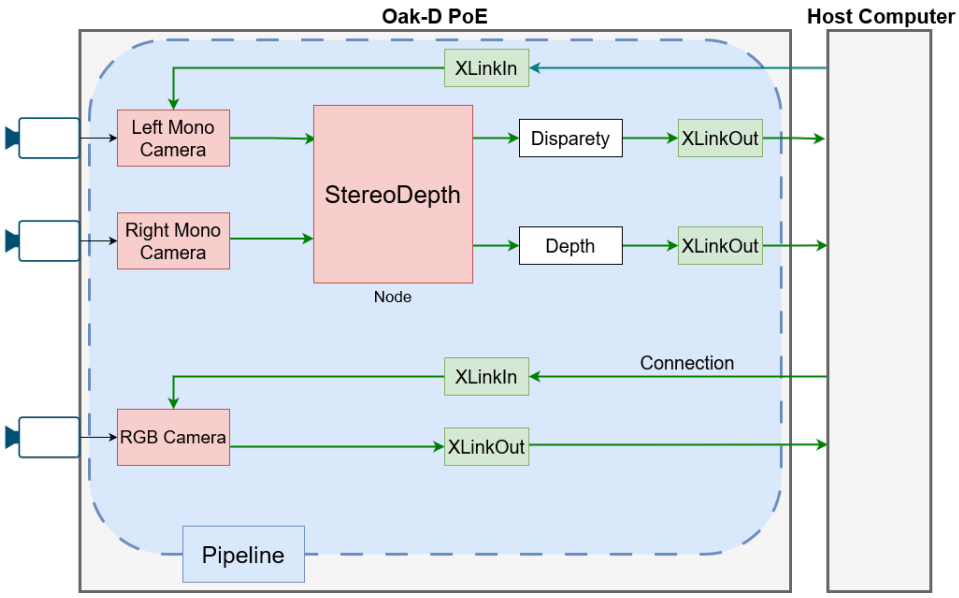

Figure 12: This figure shows how the hardware and firmware are configured. The blue area is the pipeline. The red boxes are nodes. The green boxes are connections between nodes and the host computer.

6.3.1 Camera Setup One

In the first experimental setup, the camera was attached to the upper edge of the wheel-loader cabin. For this setup, the camera could not capture images of the inside of the bucket. The angle of the camera in relation to the bucket was too insufficient to see the inside of the bucket. From Figure 13 it is evident that the camera can not see the bottom of the bucket. Another issue with this setup is that the camera is not focusing on the bucket or the material. The focus is instead on the bellcrank, which results in a lower resolution of the relevant information in the image. Upon inspection of Figure 13, it becomes evident that a substantial portion of the data is redundant, whereas only a minor fraction of the image is pertinent. Another issue with this camera position is that the far edge may be hidden behind the material if the material forms a pile. This will affect the bucket extraction which will be further elaborated in section 6.4, Bucket Extraction.

Figure 13: The left image is the disparity map and the right image is the RGB image from the first camera setup. The colour map shows that dark tones of blue are far away while light tones of blue are closer to the camera.

6.3.2 Camera Setup Two

In the second experimental setup, the camera was attached to an extension arm. The purpose of the arm is to change the angle and position of the camera relative to the bucket. The goal is to be

able to take images directly into the bucket and to take images without the wheel-loader bellcrank, boom, and backside of the bucket. Upon inspection of Figure 14, It is evident that the focus of the camera now is closer to the inside of the bucket which is an improvement from the previous setup. The image is also taken so that the bellcrank, boom, and majority of the backside of the bucket are excluded. However, the initial underlying issue is still present, and that is that the camera can not see the entire inside of the bucket.

Figure 14: The left image is the disparity map and the right image is the RGB image from the second camera setup. The colour map shows that dark tones of blue are far away while light tones of blue are closer to the camera.

Another issue emerged during this setup; due to vibrations while excavating, the camera changed its position. The great forces that affect the wheel loader when excavating from a pile of stones or gravel send a shock through the machine. This shock made the extension arm heavily vibrate which made it change position. This resulted in the data being difficult to replicate.

6.3.3 Camera Setup Three

In the third setup, the images were taken in front of the wheel loader by a person. This technique has a few advantages and disadvantages. The greatest advantage is that the camera can see the full depth of the bucket. This makes it possible to estimate the full range of fullness of the bucket. The second advantage is that the distance between the camera and the bucket is shorter, which results in a better resolution and improved depth perception. Another advantage is that the bucket used is a perfect triangle with only straight edges, this makes edge and contour detection easier. The disadvantage of this setup is the fact that the camera position is unrealistic. A camera can never be mounted in front of an operating wheel loader. Another disadvantage is that the camera position relative to the bucket is not constant.

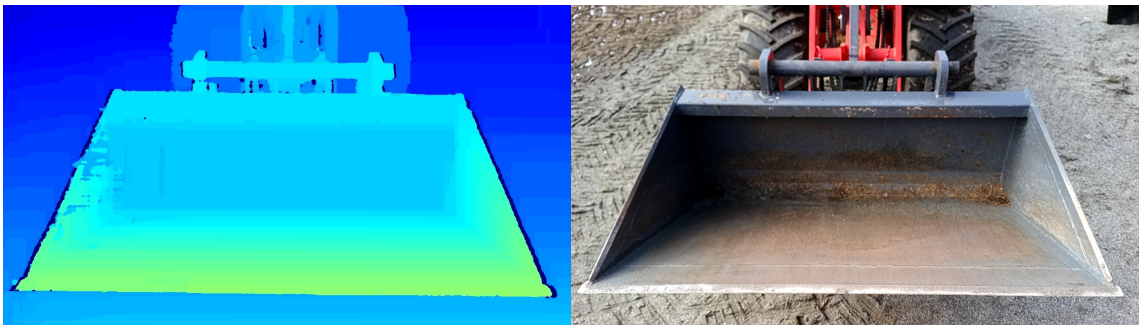

Figure 15: The left image represents the disparity map and the right image is the RGB image. The image is taken by a person standing in front of the wheel loader. The colour map shows that dark tones of blue are far away while light tones of blue are closer to the camera.

6.3.4 Dataset

Each data frame is constructed of one RGB image, its corresponding disparity map, and it's corresponding depth matrix as described in section 6.2, Software for Data Acquisition. The camera is positioned relative to the bucket as shown in section 6.3.3. The bucket has a volume of $0.3m^3$ or 300L. The requirements for the dataset are that all edges of the bucket are included in all images. This is to enable the bucket extraction 6.4. Secondly, all data is obtained indoors in a riding stable, this ensured constant lightning, no interference from shifting weather, and no shadows in the bucket. Third, all data is obtained from a relatively consistent, close distance to ensure that the focus is on the bucket and that the bucket is the largest convex contour in the image. The wheel loader bucket is filled with material by hand measured in intervals of $30L$. Five samples are taken for each interval. A hand-held bucket of volume 15L is used. The volumes sampled are $[0, 30, 60, 120, 150, 180, 210, 270, 300, 330]$ litres, this corresponds to $[0\% - 110\%]$. 30% and 80% were lost during the sampling. This procedure enables validation of the volume estimation performance since the actual percentage of fullness and the estimated fullness can be compared. At each interval, five data samples are collected. The material is reorganised in the bucket between each sample. The purpose of the reorganisation is to add variation to the data.

6.4 Bucket Extraction

In order to reduce the amount of redundant information in the collected dataset the bucket of the wheel loader was extracted from the background. The raw RGB image and the disparity map from the OAK-D-PoE in 4.4.1 were fetched to the standard DexiNed explained in 2.8 to get a thin edge-map which can be seen in Figure 16 and Figure 17. For this thesis, the DexiNed was decided to be implemented to generate a thin edge map that can be used to find the edges of the bucket. The DexiNed was chosen since Xavier Soria et al. concluded that it is superior at finding the thin edges in an image compared to multiple other edge algorithms. The DexiNed was first planned to be used solely for the material classification, similarly to Haijie Li et al. However, by observations the DexiNed showed to be promising for bucket extraction also.

The DexiNed version used was implemented in Pytorch and trained with the BIPED dataset.

Figure 16: (a): Original bucket image. (b): The output fused edge-map from the DexiNed

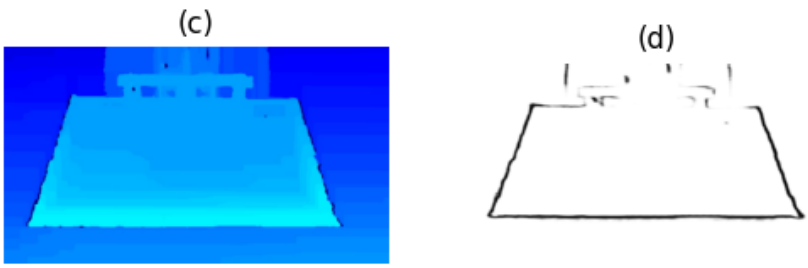

Figure 17: (c): Disparity map which displays a colour map where the colour indicates the distance to an object relative to the camera. (d): The output fused edge-map from the DexiNed from the corresponding disparity map presented in (c).

From the edge-maps in Figure 16 and Figure 17, a binary threshold image was created for each edge-map with the threshold value set to 100 in the OpenCV binary threshold function for the edge-map from the original bucket (b) and the threshold value of 150 for (d). The desired threshold value was determined through iterative testing. In both of the resulting binary images, the edge features of the bucket were not closed and bounded. By applying bitwise AND operation using the OpenCV function `cv2.bitwise_and()` with the two binary threshold images one binary image was obtained, shown in Figure 18. This process allows for the extraction of specific features or areas of interest that are common to both edge maps, resulting in a combined representation of those areas in the final binary image which in this case was the edge features of the bucket. The `cv2.bitwise_and()` takes the two binary images as input where white pixels equal 1 and black pixels equal 0 and performs bitwise AND operation where white pixels in both images outputs a white pixel, i.e. adds the black pixels from both images together.

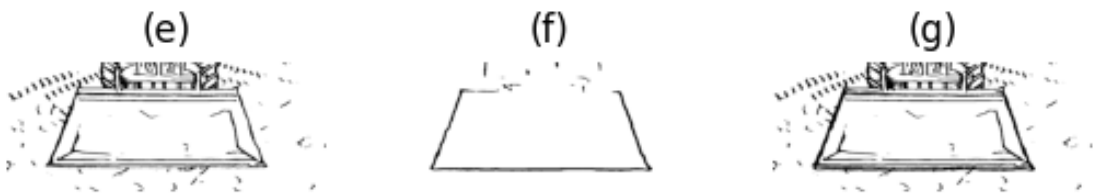

Figure 18: (e): Shows the binary threshold image obtained from edge-map (b) in Figure 16. (f): Represents the binary threshold image obtained from edge-map (d) in Figure 17. (g): Binary image from applying bitwise AND operation with (e) and (f).

On the binary image (g) shown in Figure 18 the OpenCV function `findContours()` was called to find the contours. The function uses a border following algorithm for topological structural analysis [36] on a binary image and returns the contours from the binary image [37]. For each contour, a corresponding hull was created with the `convexHull` function to determent the convexity/concavity of the contours. The OpenCV `convexHull` function finds the convex hull of a two-dimensional point set by using the Sklansky's algorithm [20] that have the complexity $O(NlogN)$ [38]. See Figure 19 for a visual representation of the detected contours and its corresponding convex hull.

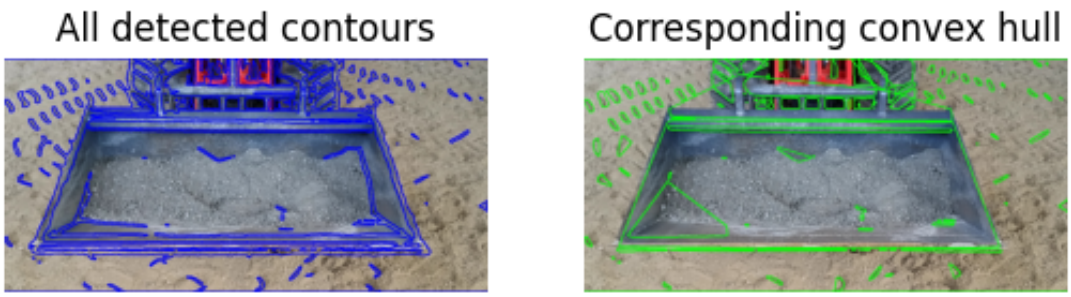

Figure 19: Left image represents the original RGB image with all the detected contours. The right image shows the corresponding convex hull for each contour.

Due to the visual scope of the camera, the assumption was made that the contour of the bucket was the largest convex contour in the image. See Figure 20 for a visual representation of the largest contour and its corresponding convex hull. To determent the convexity of an object the area of each contour and the convex hull was calculated with the `contourArea()` function, which calculates the area of an input contour using the Green's theorem [39], described in 2.15. The ratio area R_c between the contour area and the convex hull area was then calculated as follows:

$$R_c = \frac{Contour\ Hull\ Area}{Convex\ Area} \tag{6}$$

The threshold value of $R_c = 1.5$ was set to determine if the contour belongs to one convex object where the contour with the largest contour area and with a corresponding R_c-value below the threshold value was saved and the convex hull for that specific contour was drawn on an empty black image with the same image shape as the original RGB image. The R_c value was determined by iterative testing for a suitable value. By filling the pixels inside the hull with white pixels a binary mask was created. The last step of the bucket extraction is to remove redundant information from the original RGB image. The `bitwise_and()` function in OpenCV computes the bitwise conjunction of two input arrays, with the possibility to use an 8-bit single channel array as an operation mask. By using the original RGB image as both input images and with the binary mask of the convex hull as the operation mask, the output image was containing the pixels that satisfies the conditions specified by the mask which preserved the pixels from the original RGB image inside the masked area [40]. The result from the bucket extraction can be seen in Figure 21, where both the binary mask and the masked image is the input to the Superpixel segmentation.

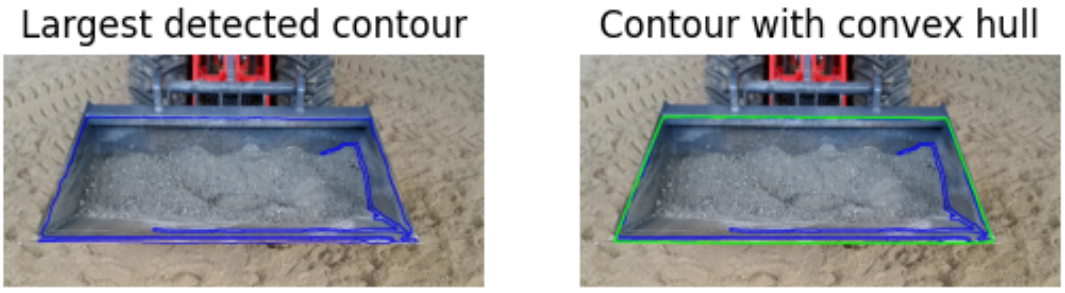

Figure 20: Left image shows the largest detected contour with blue lines drawn around the contour. The right image represents the largest detected contour drawn in blue with the corresponding convex hull drawn in green.

Figure 21: Binary mask represent the binary mask from the convex hull. Right image shows the binary mask merged together with the original RGB image.

When observing the mask produced by the above algorithm it became clear that the pixels in the edges around the bucket contained noise. The noise has two different prominent sources. One source is that the edge is missed by a few pixels and some background information is added to the mask. The other source is that the stereo correspondence fails for some pixels in proximity to the edges. When the stereo correspondence fails it tends to add a few meters to the true distance. A median filter was first applied to the masked depth data to solve this issue. However, the noise was so prominent that the filter had to have a large kernel of approximate size 20×20. This kernel size attenuated the noise, however, it also distorted the depth information. The necessity of a new

approach to this problem became evident. The new, successful approach is eroding the mask by 10 pixels into a smaller foreground (2.12). This removed a large proportion of the noisy pixel and a small number of relevant pixels around the edges. However, a few noise pixels are still left in the dataset after erosion. This is an issue because it will affect the result of the lid interpolation described in section 6.7.2. In order to remove all major noise on the boundary, two numerical approaches were implemented and tested. The distance between the camera and the furthest edge of the bucket is approximately 2.2 meters. For the first numerical approach, a threshold was set to 2.5 meters where all pixels that are further away from the camera are numerically changed. The new distance of the pixel is calculated by the formula 7, where i is the position of the noise pixel.

$$Pixel_i = Pixel_{i-1} + (Pixel_{i-1} - Pixel_{i-2}) \tag{7}$$

This solution is advantageous in theory since it incorporates a method of setting the new values for the noise pixels based on their neighbouring pixels. The underlying idea is sound, as neighbouring pixels often exhibit some level of similarity or correlation. By leveraging this relationship, the approach aims to produce a coherent result. However, during the testing phase of this numerical approach, unexpected and erroneous behaviour was observed. Despite the initial potential, the method failed to consistently deliver the desired outcomes.

Instead, a relatively simpler numerical approach was implemented. The same threshold was set to 2.5 meters. Then if a pixel surpasses that threshold, the pixel is directly set to 2.05 meters. The assigned value of 2.05 meters is an arbitrary set value based on the observation of the true distance between the camera and the bucket. This approach did not have the errors from the first solution, However, it is important to note that the simplicity of this approach comes with certain limitations. Since the assigned value of 2.05 meters is not determined based on the neighbouring pixel values or any contextual information, there is no guarantee of a coherent correlation with the surrounding pixels. Consequently, the resulting image may lack the smooth transition or consistency that could have been achieved through a more complex algorithm considering the pixel's neighbouring pixels.

6.4.1 Validation of the Bucket Extraction

To validate the result of the bucket extraction this paper has visually validated the output for each data frame from the bucket extraction as either failed, successful or ideal extraction of the bucket. Validation was done manually by visually analysing the output shown in Figure 21 for all the data in the dataset. If the algorithm were able to separate the bucket from the background the output was validated as successful, otherwise it is considered as a failure. If any edge feature inside the bucket was missed and therefore led to a loss of information, the data was only validated as a successful bucket extraction but not an ideal bucket extraction. An ideal bucket extraction can be seen in Figure 22 and a successful but not ideal extraction can is shown in Figure 23.

Figure 22: The algorithm has successfully extracted the bucket from the background while preserving all the information contained within the bucket.

Figure 23: The image reveals that the algorithm has omitted certain details in the right corner and along the lower edge.

6.5 Material Extraction

The material extraction aims to separate the material from the steel bucket. The purpose behind the separation of the material and the bucket is to facilitate the classification of the material. Since the bucket may obstruct reliable material classification it is important to create a binary mask that only includes the material, thus the classification will not be done on redundant information. Figure 24 shows the flowchart of the algorithm implemented.

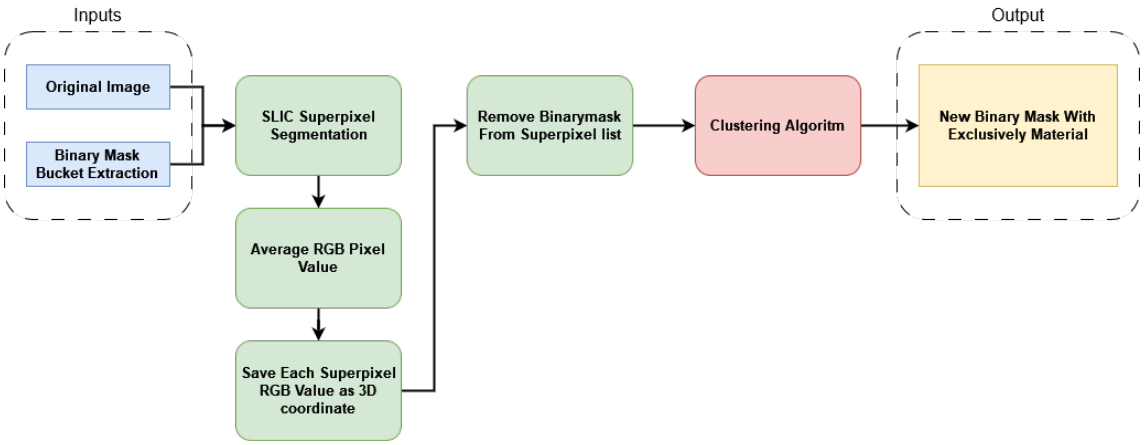

Figure 24: Flowchart of the material extraction algorithm. The algorithm takes two inputs: the original RGB image and the binary mask from 6.4. It performs a SLIC segmentation (see 2.5) and then averages the RGB value for each and every sub-pixel. The sub-pixels are then separated into two clusters. All sub-pixels that are assigned as the material will be assigned as True in the new binary mask while all other sub-pixels are assigned as False. The algorithm then outputs the new binary mask.

The algorithm receives the original image and the binary mask from the bucket extraction 6.4 as two inputs. The first step involves performing SLIC (Simple Linear Iterative Clustering) superpixel segmentation on the original image. The function used is `SLIC()` from the `skimage.segmentation` library in Python. This function takes several inputs: the original image, the binary mask, the desired number of approximate segments for the image segmentation, and a sigma constant. The sigma constant, set to 1 in this case, determines the width of a Gaussian smoothing kernel used for pre-processing in each dimension of the image. By applying this smoothing kernel, the function enhances the segmentation process. The output of the `SLIC()` function is a segmented image, where the regions covered by the input binary mask are not segmented.

SLIC superpixel segmentation was chosen as the segmentation algorithm since it can segment RGB images and not only grayscale as many other segmentation algorithms does, as described by R. Achanta et al. This is an advantage since the conversion of an image to grayscale may attenuate or remove useful information. Another advantage with SLIC is that it finds and conserves edges in images, this means that natural boundary's in an image may be saved. SLIC is also compared to multiple segmentation algorithm by R. Achanta et al. and they concluded that SLIC is the fastest segmentation algorithm with time complexity of $O(nlogn)$ and they stated that it is a memory-efficient algorithm. This is a desirable feature since this material extraction algorithm is planned to be implemented in an online setting in the future. Since G. Xiaojin et al. found success in identifying rock on the moon and Y. Ren et al. found success in separating oolites from carbonated rocks using the SLIC segmentation algorithm, this thesis will test this segmentation algorithm to separate material from the bucket. In summary, SLIC segmentation was chosen as segmentation algorithm based on the related work described in section 3.

Moving on to the next step, the algorithm calculates the mean RGB value for each sub-pixel within the segmented image. These RGB values are then saved and represented as 3D coordinates, denoted as $[R, G, B]$. However, the input binary mask is not mapped as a 3D coordinate. Once the 3D coordinates are obtained, the algorithm proceeds to cluster them into two distinct clusters. Referencing Figure 25, one cluster is labeled as "material", representing the sub-pixels identified as actual material regions. The other cluster is labeled as "redundant information," including the sub-pixels that do not belong to the material region. To generate the final output, the sub-pixels labeled as redundant information are masked as False, indicating that they are not part of the material. On the other hand, the sub-pixels labeled as material are masked as True, signifying their inclusion in the material category. This binary masking process separates the material and non-material regions within the segmented image. This clustering technique in combination with SLIC and the mean RGB value for each sub-pixel originates from this thesis, no related work has tested this particular technique to separate materials in images.

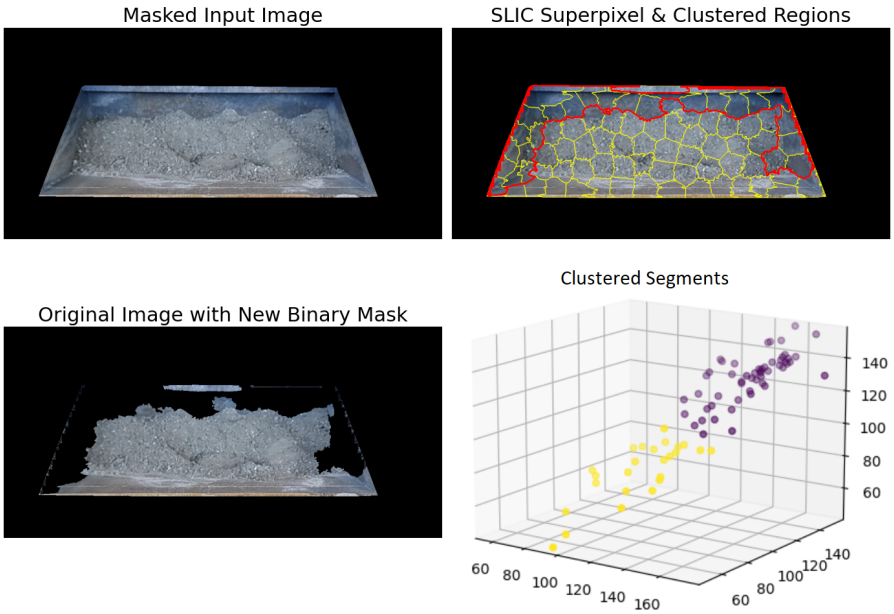

Figure 25: The figure shows the step-by-step material extraction. The first image is the input image from the bucket extraction (6.4). The second image shows the result of the superpixel segmentation and the clustering. The yellow lines are the superpixels and the red line is the region clustered as material. The third image is the output of the system. The fourth image shows the mean RGB value for each superpixel represented as a three-dimensional coordinate. The data points are clustered into two clusters. The purple points are labeled as material and the yellow points are labeled redundant. Each coordinate corresponds to one of the superpixel segments. In the third image, it can be observed that the algorithm successfully masked the upper part of the bucket but failed to mask the bottom part for the input image shown.

In summary, the algorithm takes the original image and binary mask as inputs, performs SLIC superpixel segmentation, calculates the mean RGB values for each sub-pixel, clusters the 3D coordinates into the material and redundant information clusters, and finally masks the sub-pixels as True or False to distinguish between material and non-material regions.

6.5.1 Validation of Material Extraction

The material extraction algorithm is validated by comparing the binary mask generated by the algorithm with a manually annotated binary mask of the same image for each image in the dataset. To determine the accuracy of the algorithm, the intersection area ($A \cap B$) between the extracted material A and the annotated material B is computed and compared to the area of the annotated surface. The intersection area is obtained by applying the bitwise AND operation using the `cv2.bitwise_and()` function. Additionally, the surface areas are determined by utilizing the `cv2.findContours()` function to identify all contours within the binary masks. The area of each contour is then calculated using `cv2.contourArea()`. By summing up the areas of all surfaces, we obtain the total surface area. This approach allows us to evaluate the algorithm's performance by quantifying the proportion of the true material that is successfully extracted. The result is expressed as a percentage, indicating how much of the annotated material is accurately detected by the algorithm.

The second phase of material extraction validation involves calculating the area of the material that is erroneously classified as the actual material. In this thesis, these inaccurately clustered bucket segments are referred to as "extracted redundant material". To determine the area of the redundant material, the binary inverse of the annotated material B is obtained by utilising `cv2.bitwise_not()` as a mask input in the `cv2.bitwise_xor()` function. By applying the bitwise XOR operation, which takes both the annotated material B and the extracted material A as inputs, the areas where the two materials differ are identified. This process isolates the regions where the extracted material includes segments that should not be classified as material. Consequently, the resulting binary mask highlights the areas corresponding to the extracted redundant material, allowing for the calculation of its total area and assessing the algorithm's performance more comprehensively.

6.6 Material Classification

The material classification algorithm classifies the material into two different classifications; **rocks** and **gravel**. The input to the material classification was obtained from the material extraction output. The fused edge-map obtained from the DexiNed described in section 6.4 and the binary mask from the material extraction in section 6.5 are merged together using bitwise AND operation which can be demonstrated in Figure 26.

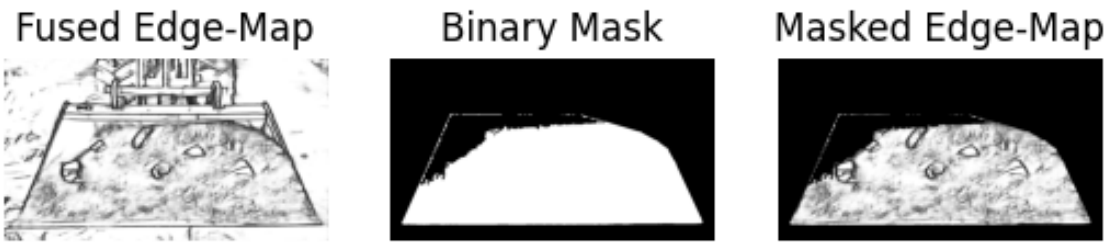

Figure 26: Images represents the fused edge-map from DexiNed, the binary mask obtained from the material extraction and the masked edge-map obtained by merging the fused edge-map and binary mask together.

On the masked edge-map shown in Figure 26 a binary filter was applied using the OpenCV threshold function with a threshold value of 200 which can be seen in Figure 27. After applying the binary filter the morphological operation **opening** was carried out on the binary image with

a kernel window with the size 10×10. The operation first performs erosion followed by dilation to smooth the contours of objects and break narrow interconnections.

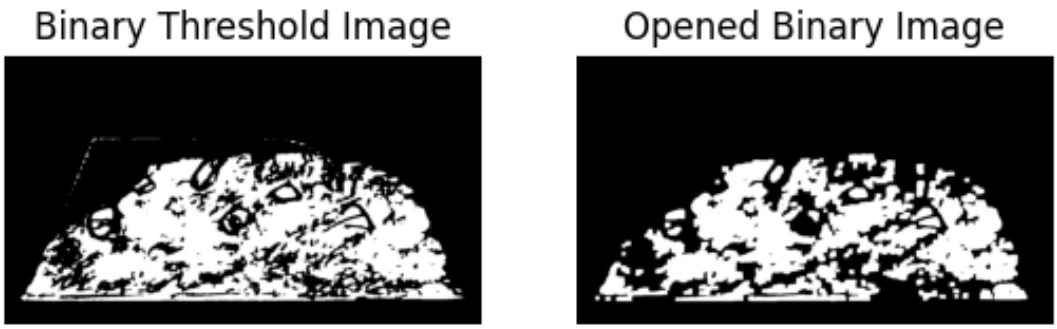

Figure 27: Left image represents the binary image after applying a binary threshold filter and the right image shows the binary image after carrying out the morphological operation **opening** with a 10×10 kernel window on the binary threshold image.

The algorithm detects the contours of the objects in the opened binary image in Figure 27 and the corresponding convex hull for each contour by using the `cv2.findContours()` and `cv2.convexHull()` functions. All the detected contours can be visually observed in Figure 28. In the same way as for the extraction of the bucket in section 6.4 the assumption was made that the object needs to be convex in order to be classified as a rock. To determent the convexity of an object the same approach as for the bucket extraction 6.4 was used in which the area of each contour and the convex hull was calculated with the `contourArea()` function. The ratio area R_c between the contour area and the convex hull area was then calculated using equation 6. To determine if the object was convex a threshold value R_c was set to $R_c = 1.3$, where objects with a R_c value smaller than the threshold value are classified as convex objects. The threshold value of R_c was set to 1.3 since Haijie Li et al. [26] used the same threshold value when they determined whether rocks in their dataset were convex or concave. The algorithm classifies the material into two different classifications; **rocks** and **gravel**. Therefore the assumption was made that **rocks** have a larger contour area than **gravel** and a threshold value was set to cluster the convex contours into either **graver** or **rocks**. The threshold value for the contour area was set to $Area_C = 500$, where convex contours with a contour area larger than the set threshold value were classified as **rocks**. The value for the $Area_C$ was obtained by iterative testing for a suitable value. The clustering of the detected contours and the final classification can be seen in Figure 28.

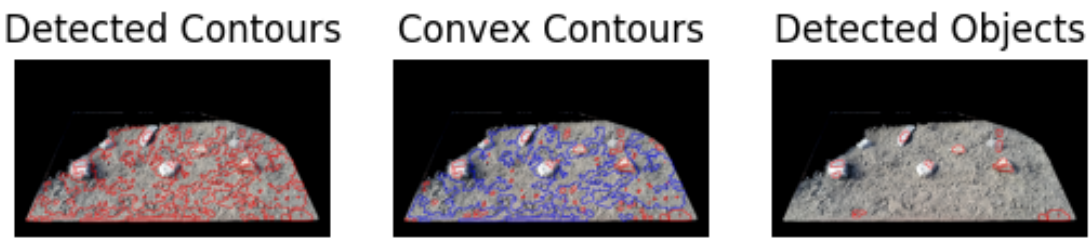

Figure 28: The left image illustrates all the detected contours from the opened binary image in Figure 27, where red lines were drawn around the contours to be able to visually observe the detected contours. The middle image represents the clustering of convex contours highlighted in red and non-convex contours highlighted in blue. The right image highlights the contours classified as belonging to **rocks**.

No result regarding the material classification will be presented. This is because the data set was not versatile enough to thoroughly test the algorithm. This is elaborated further in the discussion section 9.4.

6.7 Volume Estimation

N. Bándi et al. concluded that it is possible to map and reconstruct an object in 3D using multiple images captured from different angles in relation to the object. Since the volume estimation of the object had a robust accuracy of 90% a similar approach will be examined in this thesis. However, the average compilation time for their system was 221 seconds. This compilation time can be directly derived from the amount of data their algorithm was processing, i.e. 30 images per object. Therefore, this thesis will instead of creating a 3D object from multiple stereo-depth images, create a 3D surface from one stereo-depth image. The volume estimation is expressed as the degree of fullness of the bucket, represented in percentage. To obtain this estimation, the algorithm follows a two-step process. Firstly, it calculates the volume units for a reference image, which corresponds to an empty bucket. This step establishes a baseline for comparison. Next, the algorithm compares the volume units between the reference image and the input image, allowing for the estimation of the actual degree of fullness. To determine the depth values, the algorithm utilises the depth information obtained by the OAK-D PoE. By creating a lid for the bucket, the depth for each pixel within the bucket region can be obtained. This lid acts as a reference point, providing the necessary depth information of the bucket. For every pixel inside the bucket region, the algorithm calculates the sum of the depth values between the bucket and the bucket lid. This depth accumulation process enables the determination of the volume unit for the bucket. By comparing the accumulated depth values between the reference image (empty bucket) and the input image (bucket with content), the algorithm calculates the degree of fullness for the bucket. This estimation is expressed as a percentage, indicating the proportion of the bucket that is filled with material.

In summary, the algorithm estimates the volume of the bucket by calculating the degree of fullness in percentage. It achieves this by comparing the volume between a reference image (empty bucket) and the input image. By creating a lid for the bucket and summing the depth values for each pixel inside, the algorithm determines the volume unit. The resulting volume estimation provides an assessment of the bucket's fullness based on the difference in depth values.

6.7.1 Surface Reconstruction

The initial stage of the surface reconstruction involved the removal of depth values for the pixels located outside of the bucket. This process was accomplished by applying the binary mask from the bucket extraction (6.4), which identified the region of interest, and subsequently assigned a depth value of 0 to all the pixels outside of this mask. By eliminating the depth information outside the bucket, the focus was solely placed on the relevant data within the bucket region. In order to visually analyse the remaining depth data, a surface plot was generated. To create a grid system with coordinates, the NumPy function `numpy.meshgrid()` was used. This function utilised the number of pixels in the x-direction and y-direction of the depth data to create a grid of coordinates. The plot is only relevant for the visual analysis of the surface. The surface plot was then constructed using the depth data along with the corresponding coordinates derived from the grid system. By mapping the depth values onto the grid, the plot visualised the three-dimensional shape of the bucket surface, showcasing the variation in depth across different regions. The resulting surface plot allowed for a comprehensive understanding of the depth distribution within the bucket. Figure 29 displays the surface plot generated from the depth data, with the coordinates derived from the grid system. This visualisation aids in comprehending the spatial characteristics of the bucket's surface and provides valuable insights into the overall shape and structure of the captured data.

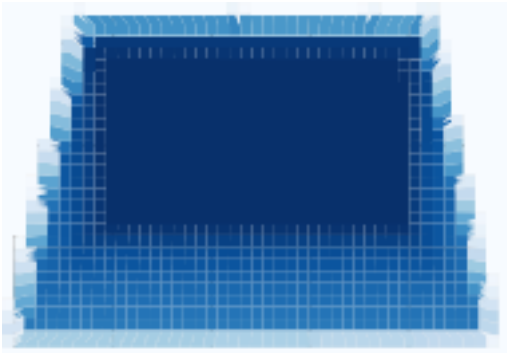

Figure 29: The figure illustrates the surface reconstruction of the bucket using depth values derived from the depth data. The surface plot showcases the three-dimensional representation of the bucket's surface, with the depth information accurately mapped onto the plot. A blue colour map is used to depict the depth values. The colour map represents a range of shades of blue, where darker shades correspond to higher depth values. The colour scheme provides a visual indication of the varying depths across different regions of the bucket's surface.

6.7.2 Lid Construction & Interpolation

To replicate the process described in the surface reconstruction in 6.7.1, a similar approach was adopted for the first step of removing depth values outside the bucket. The subsequent creation of the lid involved iterating through the depth data in the y-direction and performing linear interpolation between the two outer edge pixels in the x-direction. To identify the outer edge pixels in the x-direction, the algorithm iterates through the depth values searching for depth values above 0, which indicated the presence of pixels inside the mask. These valid depth values were stored in an array to be able to determent the maximum and minimum x-values corresponding to the outer edge pixels. By utilising linear interpolation, the algorithm defined depth values for the pixels lying between the edge pixels. This was achieved by establishing a line segment connecting the depth values of the outer edge pixels. The incremental value, referred to as I, between each data pixel in the line segment was calculated as shown in equation 8. $Depth_A$ and $Depth_B$ represent the depth values for the outer edge pixels, while $Points$ denotes the number of points between the edge pixels. The calculated increment value ensured a smooth transition between the depth values of adjacent pixels within the line segment, which was used to create the lid surface that corresponds to the shape of the bucket edges.

$$I = \frac{Depth_A - Depth_B}{Points} \tag{8}$$

The surface plot from the depth data of the bucket lid with coordinates from the grid system described in 6.7.2 can be seen in Figure 30.

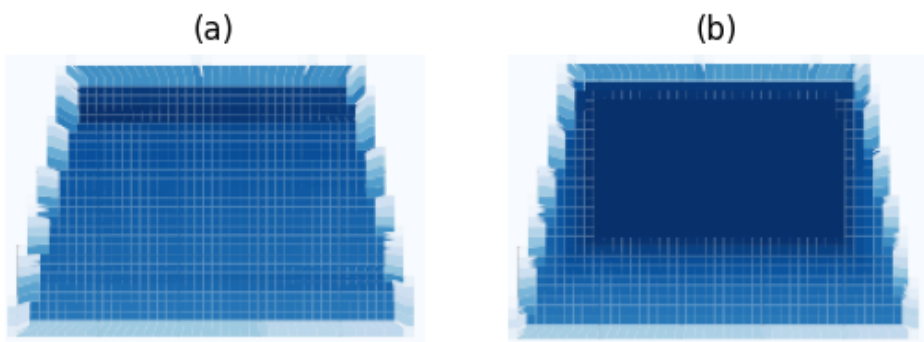

Figure 30: (a) The surface plot for the linear interpolated lid. (b) The surface reconstruction of the bucket.

6.7.3 Volume Calculation

In accordance with section 6.3.4, five data samples were gathered for each volume interval. For the five samples with an empty bucket, the individual bucket volume was calculated for each sample. The median volume value of these samples was designated as the reference volume. To determine the amount of the bucket filled by each new data sample, the volume of the data sample was subtracted from the reference volume. This value was then converted to a percentage representing the degree of fullness (DoF) by dividing the volume unit (Vol) by the reference volume unit (Vol_R) and multiplying by 100, as shown in Equation 9.

$$DoF = \frac{Vol}{Vol_R} \times 100 \tag{9}$$

To calculate the volume of one empty bucket, the bucket depth for each pixel was obtained by subtracting the distance between the camera and the lid from the distance between the camera and the bucket. The total volume is determined by summing the distances of each pixel within the bucket, see Algorithm 2 for pseudo code. The volume unit represents the unfilled portion of the bucket, where a negative value indicates overflow, meaning the material's depth is smaller than the depth of the bucket lid.

Algorithm 2 Volume Calculation for One Bucket

Data: Lid_depth, $Bucket_depth$
$i \leftarrow 0$ $Volume \leftarrow 0$
while $i < size(Lid_depth)$ **do**
  $Volume \leftarrow Volume + (Bucket_depth(i) - Lid_depth(i))$
  $i \leftarrow i + 1$
end

7. Results

This section will present the data that has been collected during this research. The goal is to provide a clear, comprehensive, and concise summary of the results obtained by the research. The results will support the conclusions and recommendations presented in the conclusion and discussion sections.

7.1 Bucket Extraction

The method to validate the bucket extraction is described in section 6.4.1, the section contains clarification of the difference between a successful and ideal bucket extraction which is included in the result in Table 2 and Table 3.

True Volume	# of Samples	Successful Extractions [%]	Ideal Extractions [%]
0L	5	100%	100%
30L	5	100%	100%
60L	5	100%	100%
120L	5	100%	80%
150L	5	100%	100%
180L	5	100%	80%
210L	5	100%	80%
270L	5	100%	80%
300L	5	100%	80%
330L	5	100%	100%

Table 2: The table presents the percentage of successful and ideal bucket extractions for the $TrueVolume$ data samples.

# of Samples	Successful Extractions [%]	Ideal Extractions [%]
50	100%	90%

Table 3: The table presents the compiled result for the bucket extraction from table 2.

7.2 Volume Estimation

The method used to validate the volume estimation is described in section 6.7. The dataset used was obtained as described in section 6.3.4. Table 5, 4, and Figure 31 are based on the same dataset.

True Volume	# of Samples	True [%]	Average [%]	Median [%]	Std dev [%]
30L	5	10%	14.7%	13.9%	2.4%
60L	5	20%	31.3%	31.8%	4.3%
120L	5	40%	47.7%	46.8%	5.4%
150L	5	50%	70.9%	70.2%	6.5%
180L	5	60%	82.5%	82.5%	1.7%
210L	5	70%	93.2%	94.4%	3.4%
270L	5	90%	102.1%	102.2%	2.0%
300L	5	100%	112.0%	112.5%	4.0%
330L	5	110%	132.7%	131.5%	5.1%

Table 4: This table presents the results from the volume estimation. The $True[\%]$ column is the true volume in percentage. The $Average[\%]$ and $Median[\%]$ columns are the average and median output of the algorithm. The $Stddev[\%]$ columns show the standard deviation in percentage points.

Volume	# of Samples	Average	Min	Max	Median	Std dev
0L	5	42901	40773	44674	42550	1688

Table 5: This table presents the calibration of the system where the volume of an empty bucket is calculated. The Volume is presented in an arbitrary unit of volume. The median value of five samples is considered as 100% of the bucket capacity.

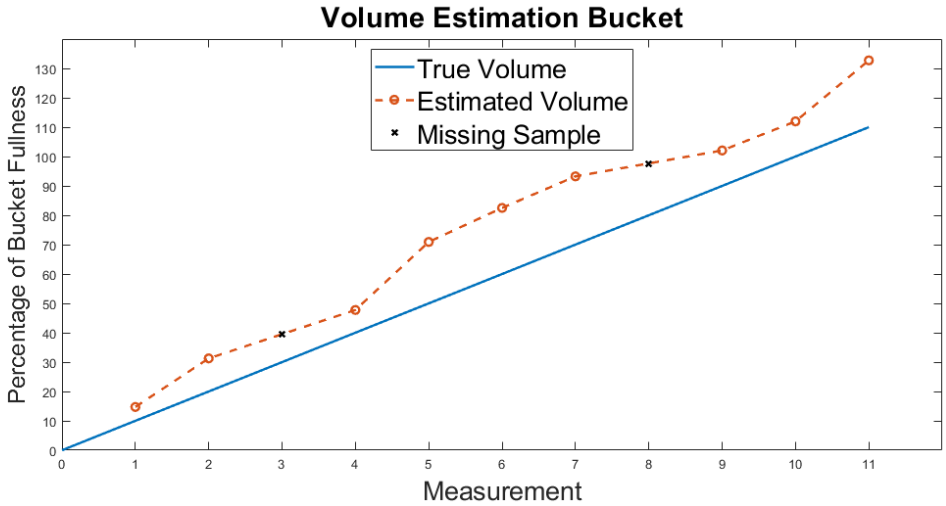

Figure 31: The result of the average estimated volume compared to the true value. The smallest volume tested was 10% and the largest was 110%. The data for 40% and 80% are missing.

7.3 Material Extraction

The material extraction is conducted as described in the Material Extraction section 6.5. The validation of the system is described in the Validation of Material Extraction, section 6.5.1. The algorithm is tested and verified on the dataset described in section 6.3.4. Table 6 shows the compilation time for the system, which is based on the hardware presented in 4.4.2. Figure 32 gives the percentage of material extracted by the algorithm and Figure 33 gives the amount of wrongly extracted bucket. Table 7 gives the fail rate of the material extraction where no material could be extracted by the algorithm.

# of Segments	# of Samples	Average [Sec]	Median [Sec]	Std dev [Sec]
60	50	1.082	1.078	0.101
120	50	1.274	1.281	0.056
180	50	1.459	1.421	0.064

Table 6: The table presents the average, median, and standard deviation in seconds for executing the segmentation and the mean RGB-value calculation for the segments.

# Segments	Fail Rate 210L	Fail Rate 270L	Fail Rate 300L	Fail Rate 330L
60	20%	80%	100%	100%
120	0%	0%	20%	20%
180	0%	0%	0%	20%

Table 7: This table shows the fail rate of the material extraction. A system failure is denoted as a system where the k-means clustering algorithm could not find any separation in the data, thus the entire image is masked. All measurements from $30L - 210L$ has a 0% Fail Rate.

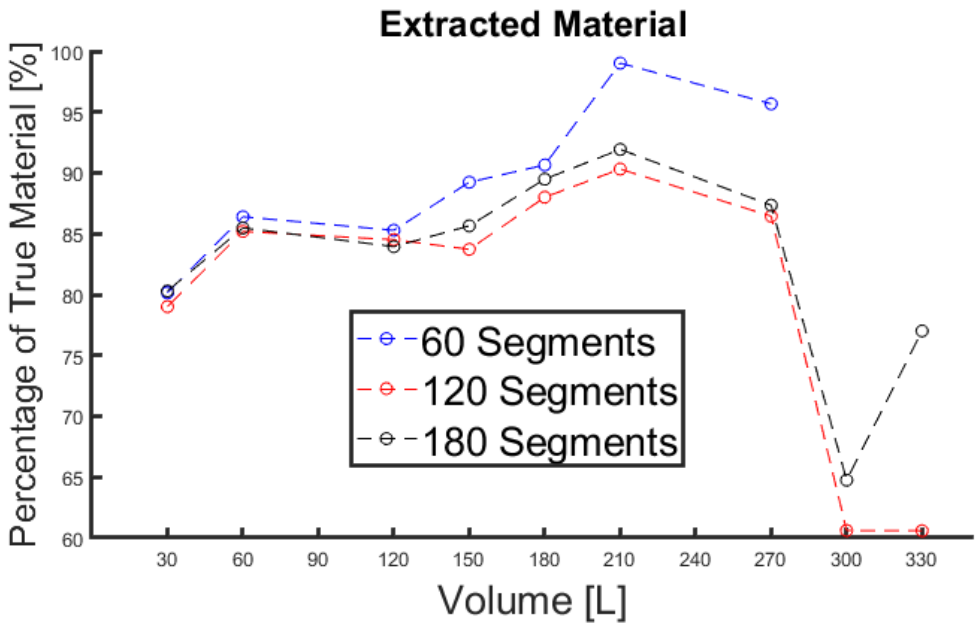

Figure 32: This graph shows the accuracy of the material extraction algorithm. The result shows how much of the true material surface is found by the algorithm.

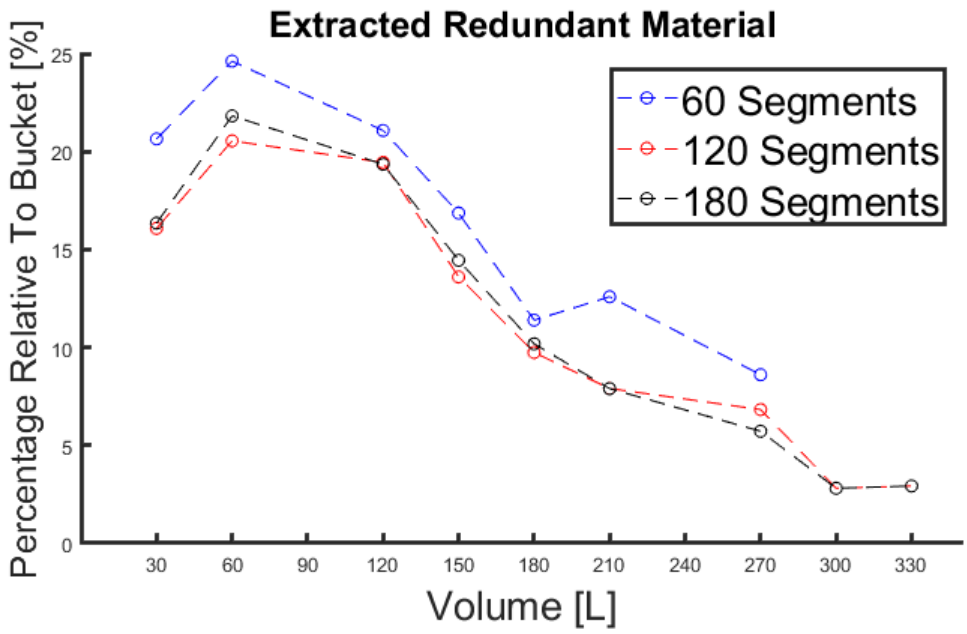

Figure 33: This graph shows how much of the bucket is wrongfully clustered as relevant material. The area of the wrongfully clustered material is compared to the total area of the bucket.

8. Conclusions

This section will summarise the work conducted in this thesis. The significance of the results and contributions of the research will be presented along with the author's reflection.

The work conducted in this thesis involved the collection of data which includes raw RGB images, their corresponding disparity maps, and depth values. The captured images include the wheel loader bucket and its material. Secondly, a system has been developed to extract the bucket from its surrounding environment, calculate the degree of fullness of the bucket, extract the material from the bucket, and classify the material in the bucket. The bucket extraction is a critical step for the system. It is essential that the bucket is correctly extracted in order for all future steps to work properly. The solution for the bucket extraction implemented in this thesis is proposed by the authors since it successfully extracted all buckets in the dataset and 10% of all buckets were extracted successfully but less than ideal. It can be concluded that the stereo matching of the camera produced erroneous results on the edges of the bucket, this resulted in the bucket extraction algorithm having to erode its output binary mask, resulting in a slightly deteriorated result. An issue with the implemented solution is that the data is captured in front of the wheel loader. This camera position is impossible to use in a real, excavation environment. The data was captured in this position since it facilitates edge detection for this algorithm. If data is captured from a position where the edges are concealed and inaccessible, this algorithm will fail.

The volume estimation takes the extracted bucket as an input. This step in the system examines research question 2, "How can computer vision be utilised to identify the amount of material in a bucket on a wheel loader". To summarise this step, the boundary of the bucket's binary mask from the bucket extraction is used to interpolate a lid over the 3D bucket. The distance between the camera and the lid and the 3D bucket is used to calculate the volume of the bucket. The calculated volume is then compared to the reference volume of the bucket to determine the degree of fullness. It can be concluded that the algorithm slightly overestimates the volume and that multiple sources contribute to the error.

The material extraction is the process of separating the material from the bucket with the purpose to facilitate future material classification. The proposed method is not recommended to be applied for material extraction in the future for similar data. This is because the mean RGB value of each superpixel is too similar when comparing bucket and material. This prohibits the clustering algorithm from separating the material and the bucket. This technique may, however, be useful in other environments with distinct colours for the foreground and the background. A simpler and faster solution is proposed where if the volume is less than a set percentage of the bucket capacity, no material extraction is conducted since a retake is necessary. If the material that fills the bucket is more than the set percentage, the bucket mask should be eroded by a statically defined integer to remove the bucket around the edges. This could be a promising solution for the extraction of material since the current system is slow and inefficient.

9. Discussion

This section will analyse and interpret the results obtained from the conducted research. The objective is to discuss and compare the results in the context of existing knowledge in the field of study. The discussion will provide a comprehensive interpretation of the results which involves an explanation of the significance and meaning of the results. It will account for how the results align with or deviates from the initial research questions.

9.1 Bucket Extraction

The bucket extraction algorithm is proven to be successful for all the data tested in this master's thesis. The combined dexined thin edge-map from the raw RGB input image and the disparity map contains all the crucial edges needed to create the largest contour around the bucket. However, the bucket extraction does have a few issues. The first issue is that the algorithm, in certain cases, finds the wrong edge for the furthest edges on the bucket. Visually, a human can easily detect that there are two straight edges parallel to each other. The system creates a convex hull for the binary mask based on the wrong edge 10% of the time, as seen in Table 4 in section 7.1. The consequences of the algorithm finding the wrong edge will be further discussed in the volume estimation section 9.3. The second issue can be derived from the stereo-matching of the camera. Since the camera produces inaccurate distance measurements for some pixels on the very edge of the bucket, the mask needs to be eroded as described in section 6.4. The problem with the erosion is that the correctly found edges will be dilated and information will be lost. The consequences of dilating the binary mask will be further elaborated in 9.3.

9.2 Material Extraction

The algorithm used for material extraction exhibits poor performance with the dataset used in this implementation. Upon visual examination of the SLIC segmentation, it becomes evident that a significant number of edges between the material and the bucket are correctly identified. However, the issue arises due to the similarity in mean RGB values between the material and the bucket. Consequently, the clustering algorithm frequently misclassifies segments, erroneously clustering material as the bucket and vice versa. This Algorithm would benefit if the $[R, G, B]$ values of the bucket and material have more prominent differences since the two clusters would be more separated. Therefore, it becomes evident that the algorithm is flawed since it requires data with certain colour differences. Figure 32 presents an evaluation of the algorithm's performance at different segment sizes. It demonstrates that the algorithm achieves its best results when the image is divided into 60 segments, outperforming the configurations with 120 and 180 segments within the range of $60L$ to $270L$. However, for the 60-segment configuration, the algorithm completely fails to work effectively beyond $300L$. In contrast, both the 120-segment and 180-segment configurations perform similarly within the interval of $30L$ to $270L$, with the 180-segment configuration showing noticeable improvement from $300L$ to $330L$. Furthermore, Figure 33 illustrates the extraction of redundant information from the image. It indicates that the 60-segment configuration tends to exhibit the highest over-extraction, while the 120-segment and 180-segment configurations produce similar results. As the material extraction process progresses from $60L$ and beyond, the over-estimation decreases for all segment sizes. This can be attributed to the reduction in the bucket's surface area relative to the material's surface area as the bucket becomes filled. Table 6 shows the time it takes to compute the SLIC and mean RGB for all sub-pixels. The difference is approximately 200 milliseconds between 60 and 120 sub-pixels and 140 milliseconds between 120 and 180 sub-pixels. This means that the number of sub-pixels affects the time it takes to segment the image.

9.3 Volume Estimation

In Figure 31 the result of the average volume estimation and the true baseline are presented. The corresponding table 4 shows the same information, the median estimate, and the standard deviation. By observing the result it is evident that the average estimated volume correlates with the baseline. However, the system overestimates the volume for each and every measurement. This overestimation is believed to be caused by the erosion of the binary mask from the bucket

extraction as described in the section 9.1. This is because the volume is only calculated for the depth values within the binary mask, thus the calculated capacity of the bucket will be smaller than the true capacity when the edges are eroded.

The standard deviation is believed to be influenced by multiple sources. One of these sources is most likely the lid. The lid is essential for the volume calculation, both for the reference capacity and the estimation. Since it is interpolated based on the depth information from the edge pixels of the bucket extraction binary mask and, as mentioned previously in the section 6.4, these pixels contain more noise than the other pixels in general. It is believed that the construction of the lid deviates from one image to another, resulting in a small deviation in volume calculation over the dataset. In accordance with the details provided in section 6.4, an algorithmic approach was employed to reduce the noise present in the boundary pixels. The objective was to achieve a smoother lid surface. However, the implemented numerical method is flawed since it identifies distances of 2.5 meters or greater from the camera and assigns them a value of 2.05 meters. Consequently, the assigned value is likely to deviate slightly from its neighbouring values and the true depth value for that pixel.

Another source that is believed to impact the result of the volume estimation is the fact that the material was not laying flat in the bucket. As described in the Data Acquisition section 6.3, the material was placed unevenly in the bucket, creating piles and gaps in the material. These piles and gaps will add variation to the volume. This is because certain regions of the material surface are concealed from the camera's view by the piles. As a result, these hidden cavities are mistakenly calculated as being filled with material since the camera cannot capture these regions. Consequently, the algorithm used for volume estimation carries an inherent risk of overestimating the degree of fullness for materials with uneven distributions of material. It is important to acknowledge that this limitation arises from the nature of the data acquisition process described in section 6.3. Since the material was not lying flat and had irregular formations within the bucket, the algorithm's ability to accurately assess the true volume is impacted. However, it is important to recognise that this issue is inherent to the volume estimation process in a real-world setting. When using a wheel loader for excavation, it is inevitable that the material will be unevenly placed in the bucket. The very nature of the excavation process involves scooping and lifting materials, resulting in piles and gaps within the bucket. In practical scenarios, operators or autonomous wheel loaders typically focus on efficiently filling the bucket, often prioritising speed and productivity. This approach naturally leads to variations in the distribution of material, as the primary objective is to complete the task rather than achieving a perfectly uniform load. Therefore, due to the operational constraints and objectives involved in real-world excavations, it is unrealistic to expect the material to be consistently and uniformly laid flat within the bucket. As a consequence, the volume estimation algorithms must contend with the inherent unevenness caused by the excavation process. Recognising this inherent limitation, it becomes crucial to understand the potential for an overestimation in volume calculations. The algorithm's risk of overestimating the degree of fullness is influenced by the irregularities and concealed regions within the material caused by uneven placement in the bucket. Understanding this is essential when interpreting and utilising volume estimations obtained in real-world scenarios involving wheel loaders or similar equipment.

The current volume estimation estimates the degree of fullness and could not be applied with Camera Setup 1 described in section 6.3.1 and Camera Setup 2 from section 6.3.2 due to the view inside of the bucket from the camera being insufficient. However, with some adjustments to the algorithm, it should be possible to use the same proposed approach for the volume estimation but instead of calculating the degree of fullness instead, use the system to detect if enough material is in the bucket to not have to take a retake. For example, if the desired degree of fullness of the bucket for each dig was set to at least 60%, then the wheel loader can angle the bucket in a position where only 40% of the inside of the bucket is visible from the camera. With the same current solution for estimating the volume, the algorithm should follow the same two-step process. First, calculate the volume units for a reference image, which corresponds to an empty bucket in the position where only 40% of the inside of the bucket is visible. Next, the algorithm should compare the volume units between the reference image and the input image in order to detect material in the input image. If material is detected it can be assumed that the non-visible 60% of the inside bucket contains material. This solution should use some margin of error. The information obtained could be useful in order to make autonomous wheel loaders more effective.

Human error may also have played a role in affecting the accuracy of volume estimation through the annotation of the dataset. The validation process assumes that the total volume of the bucket is precisely $300L$, representing 100% of its capacity. However, since the bucket was manually filled using a handheld bucket with a capacity of $15L$, it was challenging to ensure that the exact amount of $15L$ was added each time. This introduces a potential source of minor discrepancies. The cumulative effect of these small errors may contribute to a larger margin of error in the volume estimation.

9.4 Material Classification

As mentioned in the implementation of the material classification, section 6.6, no final result has been completed and disclosed regarding the material classification for this thesis. This means that research question 3 can not be answered. The collected dataset that is described in the dataset section 6.3.4 has too little variation regarding rocks and gravel. This means that it is not feasible to verify the performance of the system. In order to properly verify the proposed material classification algorithm a dataset with a wide variation of rocks and gravel needs to be obtained. Secondly, a method to quantify the precision of the proposed algorithm needs to be developed. By observation during implementation, it became clear that the algorithm proposed by Haijie Li et al has the potential for successful classification of the material in the bucket. Nonetheless, no concise results have been compiled, thus no conclusions of the system can be made.

References

[1] S. Dadhich, 'Automation of wheel-loaders,' Luleå University of Technology, 2018, ISBN: 978-91-7790-259-1.

[2] N. Koyachi and S. Sarata, 'Unmanned loading operation by autonomous wheel loader,' in *2009 ICCAS-SICE*, 2009, pp. 2221–2225.

[3] V. CE. 'History of volvo ce.' (), [Online]. Available: https://www.volvoce.com/global/en/this-is-volvo-ce/our-history/. (accessed: 04.04.2023).

[4] K. R. Raghunathan, 'History of microcontrollers: First 50 years,' *IEEE Micro*, vol. 41, no. 6, pp. 97–104, 2021. DOI: 10.1109/MM.2021.3114754.

[5] R. Szeliski, *Computer vision: algorithms and applications, 2nd Edition*. Springer, 2021, ISBN: 978-3-030-34372-9.

[6] X. Zhang and S. Xu, 'Research on image processing technology of computer vision algorithm,' in *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, 2020, pp. 122–124. DOI: 10.1109/CVIDL51233.2020.00030.

[7] R. Choudhary and H. K. Gianey, 'Comprehensive review on supervised machine learning algorithms,' in *2017 International Conference on Machine Learning and Data Science (MLDS)*, 2017, pp. 37–43. DOI: 10.1109/MLDS.2017.11.

[8] S. V. Mahadevkar, B. Khemani, S. Patil *et al.*, 'A review on machine learning styles in computer vision—techniques and future directions,' *IEEE Access*, vol. 10, pp. 107 293–107 329, 2022. DOI: 10.1109/ACCESS.2022.3209825.

[9] Y. Ji, Y. Li, X. Sun, S. Yan and N. Guo, 'Stereo matching algorithm based on binocular vision,' in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, 2020, pp. 843–847. DOI: 10.1109/IFEEA51475.2020.00177.

[10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua and S. Süsstrunk, 'Slic superpixels compared to state-of-the-art superpixel methods,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012. DOI: 10.1109/TPAMI.2012.120.

[11] F. Yi and I. Moon, 'Image segmentation: A survey of graph-cut methods,' in *2012 International Conference on Systems and Informatics (ICSAI2012)*, 2012, pp. 1936–1941. DOI: 10.1109/ICSAI.2012.6223428.

[12] A. Dubey and A. D. A. Choubey, 'A systematic review on k-means clustering techniques,' vol. 6, no. 6, DOI: https://doi.org/10.1016/j.ins.2022.11.139.

[13] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, 'A survey of convolutional neural networks: Analysis, applications, and prospects,' *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022. DOI: 10.1109/TNNLS.2021.3084827.

[14] X. Soria, E. Riba and A. Sappa, 'Dense extreme inception network: Towards a robust cnn model for edge detection,' in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020, pp. 1912–1921. DOI: 10.1109/WACV45572.2020.9093290.

[15] S. M. Pizer, E. P. Amburn, J. D. Austin *et al.*, 'Adaptive histogram equalization and its variations,' *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987, ISSN: 0734-189X. DOI: https://doi.org/10.1016/S0734-189X(87)80186-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0734189X8780186X.

[16] 'Opencv, contour features.' (), [Online]. Available: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html (visited on 03/04/2023).

[17] 'Morphological operations.' (), [Online]. Available: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html (visited on 06/03/2023).

[18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Pearson Education International, 2008, p. 965. DOI: 10.1117/1.3115362.

[19] G. George, R. M. Oommen, S. Shelly, S. S. Philipose and A. M. Varghese, 'A survey on various median filtering techniques for removal of impulse noise from digital image,' in *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, 2018, pp. 235–238. DOI: 10.1109/ICEDSS.2018.8544273.

[20] J. Sklansky, 'Finding the convex hull of a simple polygon,' *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79–83, 1982, ISSN: 0167-8655. DOI: https://doi.org/10.1016/0167-8655(82)90016-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167865582900162.

[21] R. A. Adams and C. Essex, *Calculus*, 8th. Boston: Pearson, 2013, ISBN: 978-0-321-78107-9.

[22] X. Gong and J. Liu, 'Rock detection via superpixel graph cuts,' in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 2149–2152. DOI: 10.1109/ICIP.2012.6467318.

[23] Y. Ren, J. Liang and S. Luo, 'Two extraction methods for carbonate rock oolites based on image segmentation algorithm,' in *2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, 2021, pp. 137–141. DOI: 10.1109/ICISCAE52414.2021.9590702.

[24] D. Martin, C. Fowlkes, D. Tal and J. Malik, 'A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,' in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, 416–423 vol.2. DOI: 10.1109/ICCV.2001.937655.

[25] P. Arbeláez, M. Maire, C. Fowlkes and J. Malik, 'Contour detection and hierarchical image segmentation,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011. DOI: 10.1109/TPAMI.2010.161.

[26] H. Li, G. Asbjörnsson and M. Lindqvist, 'Image process of rock size distribution using DexiNed-based neural network,' *Minerals*, vol. 11, no. 7, p. 736, 7th Jul. 2021, ISSN: 2075-163X. DOI: 10.3390/min11070736. [Online]. Available: https://www.mdpi.com/2075-163X/11/7/736 (visited on 23/01/2023).

[27] Y. Mizuchi, T. Ogura, Y. Hagiwara, A. Suzuki, Y. Kim and Y. Choi, 'Distance measurement system using a stereo camera and radial pattern target for automatic berthing control,' *Journal of the Korea Society For Power System Engineering*, vol. 17, no. 5, pp. 121–127, 31st Oct. 2013, ISSN: 1226-7813. DOI: 10.9726/kspse.2013.17.5.121. [Online]. Available: http://koreascience.or.kr/journal/view.jsp?kj=DRGGBE&py=2013&vnc=v17n5&sp=121 (visited on 02/02/2023).

[28] G. Ohira, S. Kashioka, Y. Takao, T. Iyota and Y. Tsuda, 'Autonomous image-based navigation using vector code correlation algorithm for distant small body exploration,' *Acta Astronautica*, vol. 196, pp. 400–413, 2022, ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2020.10.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576520306068.

[29] N. Bándi, R.-B. Tunyogi, Z. Szabó, E. Farkas and C. Sulyok, 'Image-based volume estimation using stereo vision,' in *2020 IEEE 18th International Symposium on Intelligent Systems and Informatics (SISY)*, 2020, pp. 000 055–000 060. DOI: 10.1109/SISY50555.2020.9217089.

[30] N. Snavely, S. M. Seitz and R. Szeliski, 'Photo Tourism: Exploring Photo Collections in 3D,' en, DOI: https://doi.org/10.1145/1141911.1141964.

[31] J. Nunamaker and M. Chen, 'Systems development in information systems research,' in *Twenty-Third Annual Hawaii International Conference on System Sciences*, vol. 3, 1990, 631–640 vol.3. DOI: 10.1109/HICSS.1990.205401.

[32] 'OAK-d-PoE — DepthAI hardware documentation 1.0.0 documentation.' (), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/SJ2088POE.html (visited on 30/01/2023).

[33] 'Powering PoE devices — DepthAI hardware documentation 1.0.0 documentation.' (), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/guides/powering_poe_devices.html#powering-poe-devices (visited on 30/01/2023).

[34] 'OAK-SoM-pro — DepthAI hardware documentation 1.0.0 documentation.' (), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/BW2099.html#bw2099 (visited on 30/01/2023).

[35] 'Waterproof enclosures — DepthAI hardware documentation 1.0.0 documentation.' (), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/articles/waterproof.html#ip67-rated-poe-enclosures (visited on 30/01/2023).

[36] S. Suzuki and K. be, 'Topological structural analysis of digitized binary images by border following,' *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985, ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X85900167.

[37] 'Opencv, findcontours.' (), [Online]. Available: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a (visited on 27/03/2023).

[38] 'Opencv, convexhull.' (), [Online]. Available: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga014b28e56cb8854c0de4a211cb2be656 (visited on 27/03/2023).

[39] 'Opencv, contourarea.' (), [Online]. Available: https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga2c759ed9f497d4a618048a2f56dc97f1 (visited on 27/03/2023).

[40] 'Opencv, bitwise_and.' (), [Online]. Available: https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga60b4d04b251ba5eb1392c34425497e14 (visited on 27/03/2023).