RGB Image-Based Detection of Liquid Fill Levels in Known Transparent Containers

DIPLOMARBEIT

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. M. Vincze
Dr.techn. Jean-Baptiste Nicolas Weibel, MSc

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
Elisabeth Fabini
e11813822@student.tuwien.ac.at

Vienna, April 2025

Vision for Robotics Group
A-1040 Wien, Gusshausstr. 27, Internet: https://www.acin.tuwien.ac.at

Acknowledgements

I want to thank Ao.Univ.Prof. Dipl.-Ing. Dr.techn. M. Vincze for supervising this thesis and providing valuable input for improvement.

Further, I would like to express my gratitude towards Dr.techn. Jean-Baptiste Nicolas Weibel, MSc, who supported me throughout the course of this thesis. His ideas and suggestions on how to approach challenging parts were essential to the completion of this work.

Vienna, April 2025

Preamble

In the light of academic integrity we kindly acknowledge the support of Grammarly [1] for spell-checking and improving the readability of this thesis, using the AI tool: "Improve it".

Furthermore, we acknowledge the assistance of OpenAI's ChatGPT (GPT-3.5 and GPT-4o Mini) [2] in modifying and refining code.

Vienna, April 2025

Abstract

Automatic handling of transparent liquid containers is required in many current and future applications in laboratories, industries, and in homes, necessitating the estimation of the objects' fill levels. Examples include robotic systems used in beverage production and assistive care robots in domestic settings. While several methods have been developed for liquid detection, only a few solely rely on RGB images. The advantage of using RGB images lies in their widespread availability, low-effort and low-delay acquisition, making these methods more accessible. This thesis presents a novel approach for detecting two-dimensional and three-dimensional liquid fill levels in known transparent containers. Unlike earlier methods, it does not require any restrictions on the environment or the containers themselves. This approach integrates the Neural Networks Region of Interest Neural Network (ROI-NN) and Segment Anything Model (SAM) for liquid segmentation, along with additional gradient descent optimizations to refine the output and achieve three-dimensional results. An evaluation of the proposed method concludes that it yields more accurate results than current state-of-the-art techniques for liquid level detection. On the tested dataset with cluttered and realistic scenarios, it achieves over 98% accuracy. This demonstrates its potential for real-world applications that require high-quality results, such as robotic manipulation in the beverage industry or automated liquid handling in laboratory settings.

Kurzzusammenfassung

Die automatische Handhabung von durchsichtigen Flüssigkeitsbehältern ist in vielen aktuellen und zukünftigen Anwendungen in Labors, der Industrie und in Haushalten erforderlich und erfordert die Schätzung des Füllstands der Objekte. Beispiele hierfür sind Robotersysteme in der Getränkeherstellung und unterstützende Pflegeroboter im häuslichen Bereich. Es wurden zwar mehrere Methoden zur Erkennung von Flüssigkeiten entwickelt, aber nur wenige stützen sich ausschließlich auf RGB-Bilder. Der Vorteil der Verwendung von RGB-Bildern liegt darin, dass sie weithin verfügbar sind und mit geringem Aufwand und geringer Verzögerung aufgenommen werden können, was diese Methoden leichter zugänglich macht. In dieser Arbeit wird ein neuartiger Ansatz zur Erkennung von zwei- und dreidimensionalen Flüssigkeitsfüllständen in bekannten transparenten Behältern vorgestellt. Im Gegensatz zu früheren Methoden erfordert er keine Einschränkungen hinsichtlich der Umgebung oder der Behälter selbst. Dieser Ansatz integriert die Neuronalen Netze Region of Interest Neural Network (ROI-NN) und Segment Anything Model (SAM) für die Segmentierung von Flüssigkeiten, zusammen mit zusätzlichen Optimierungen durch Gradientenverfahren, um die Ausgabe zu verfeinern und dreidimensionale Ergebnisse zu erzielen. Eine Evaluierung der vorgeschlagenen Methode kommt zu dem Schluss, dass sie genauere Ergebnisse liefert als die derzeit modernsten Techniken zur Erkennung von Flüssigkeitsständen. In dem getesteten Datensatz mit unübersichtlichen und realistischen Szenarien erreicht sie eine Genauigkeit von über 98%. Dies zeigt ihr Potenzial für reale Anwendungen, die qualitativ hochwertige Ergebnisse erfordern, wie z. B. die Robotermanipulation in der Getränkeindustrie oder die automatische Handhabung von Flüssigkeiten in Laborumgebungen.

Contents

List of Figures

List of Tables

1 Introduction

Handling transparent objects remains a significant challenge in automation, particularly when dealing with liquid-filled transparent containers, an area of growing interest. Whether in manufacturing, laboratory work, or household applications, detecting and managing these containers is crucial for tasks such as automated filling, pouring, and monitoring liquid levels. While this issue is particularly relevant in laboratory environments, its applications are expanding to hospitals and homes as well [3].

Automated handling of containers containing liquids requires the knowledge of the liquid fill level of containers, used for automatically filling in and pouring out of a container or controlling fill levels. For example, checking the fullness of bottles in production [4] or controlling hospital drips [5] [6].

There are many methods to acquire this information for different use cases and the fill level detection changes drastically depending on the available and required information. Three aspects that alter the detection method are technological availability, accuracy requirements, and setup considerations.

1. Technological Availability: Advanced technologies, like contact-based sensors, lasers, and depth cameras, require expensive equipment. While contact measurement methods yield reliable results, placing a sensor directly in the liquid is not feasible in many situations. Non-contact approaches necessitate specific setups and often struggle to identify the liquid fill level accurately. [7] In contrast, using only RGB images is generally more available and affordable but introduces significant challenges, including reflections and refractions on transparent liquid containers and background noise. Finding the fill level in transparent containers could be further helped by prior knowledge of the containers and their poses, reducing the possibility of error in contrast to having to find the containers in the image first.

2. Accuracy Requirements: Depending on the application, different degrees of accuracy are necessary. For some tasks, it is only required to know whether the liquid is higher or lower than a specified threshold [8]. For others, categories like “low”, “middle” or “high” are enough to determine the fill level [9]. Other tasks require the determination of the level as accurately as possible [10].

3. Setup Considerations: Another varying factor is the setup. If the container is in a fixed position, with a plain background in a controlled environment, the detection process reduces to a repeated, predictable problem that is solved with simple approaches. If the objects with liquid are standing anywhere on a table, for example in a cluttered kitchen, with no specified camera angle and noisy background, simple methods that are used for the first case are not applicable anymore.

Due to the number of different applications, requirements, and restrictions, finding a universal method of detecting the liquid fill level is not realistic. The method of detection changes for available technologies, required accuracy, and setup, which is what makes covering each of them in different methods so important.

1.1 Challenges

A setup for detection using advanced technologies such as lasers or depth cameras is not always available, which is why this thesis relies on only RGB images for detection. The following section covers specific challenges arising when using RGB images for detection, which has been gaining more attention lately. Figure 1.1 shows the growing interest by showing publications by year on Google Scholar [11] with the keywords: "RGB image transparent container liquid fill level detection".

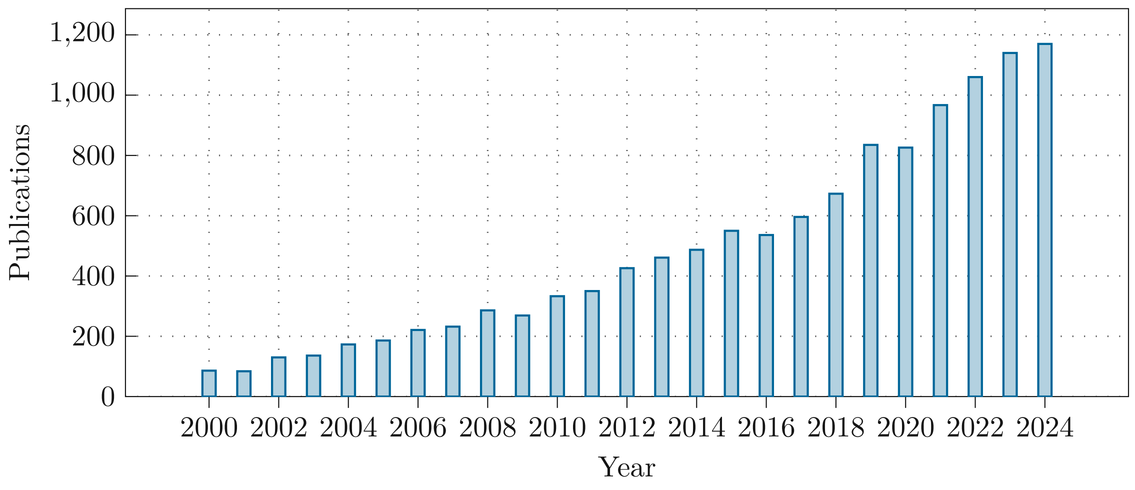


Figure 1.1: Publications on Google Scholar with the keywords "RGB image transparent container liquid fill level detection"

1.1.1 Challenges detecting transparent objects

In images, transparent object detection is particularly difficult, because the objects don't follow characteristics of nontransparent objects, like having a specific color or pattern on the surface. Instead, they show part of the background, usually distorted by light refractions, along with reflections in between. Inspecting the same object from different angles in different lighting leads to seemingly completely different-looking objects. While humans easily recognize transparent objects based on context and subtle visual cues, computer vision struggles due to the absence of clear edges and the high variability in appearance.

1.1.2 Challenges detecting liquid in transparent objects

Detecting liquids inside transparent containers makes the process even more complicated, especially if the liquid is clear and blends with the surrounding colors. The detection difficulty increases further with varying liquid color, transparency, and different container materials, which affect light transmission and refraction. Often, only a few faint edges indicate the liquid fill level, that are frequently less visible than background edges, which proves to be a challenge for automated detection. Additionally, reflections, distortions, and optical effects caused by container shape and lighting conditions further complicate the process.

1.1.3 Previous solutions and their limitations

Many previous works simplify this challenge by setting restrictions regarding the background and the object position [8][5][6][12][13][10]. This cannot be applied to realistic environments. Even though a simple solution in restricted cases is beneficial, a more thorough approach is needed for generalized setups. A few approaches [14][15][16] use no restrictions by training Neural Networks (NNs) for detecting both objects and liquids freely, however, partly coverage of the containers or noisy background leads to slightly inaccurate results. Additionally, most existing techniques for detecting liquid fill levels only provide predictions in 2D. For precise handling of liquid-filled containers and accurate volume estimation, it is essential to obtain 3D results. Unlike these previous approaches that rely on strict environmental constraints, a method must be established that works in varied, realistic environments with high accuracy and generates 3D liquid fill level estimations.

1.2 Contributions

This thesis introduces a method satisfying the qualities required in the previous section. It proposes liquid fill level detection in RGB images in known containers with known container poses. Without restrictions, the liquid containers with known poses are freely placed in the scene. Bottles, for example, don't necessarily have to stand vertically but are also tilted, the containers do not need to have a specific shape, and objects are even partly covered with still functioning fill level detection. The background is noisy, and the setup is random and realistic, as shown in an example image in Figure 1.2. The difficulty of liquid detection is raised to a high level by setting little restrictions.

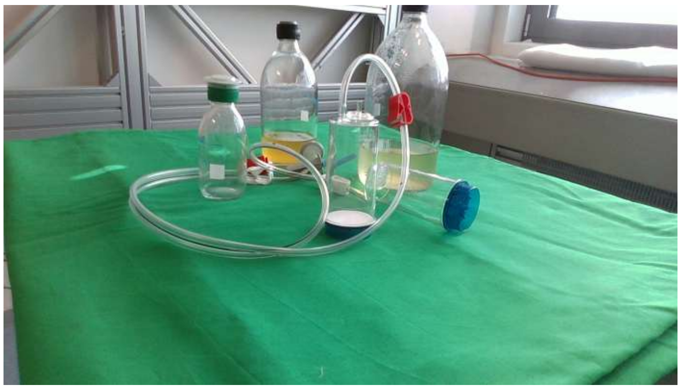


Figure 1.2: RGB Image 8

Another challenge this thesis tackles is achieving maximum accuracy by optimizing the detection of the fill level and not sorting the results into categories or checking against a threshold. This is achieved by working with known containers and their corresponding 3D models. This thesis not only detects the liquid fill level in RGB images but at the same time returns the fill level in the 3D models of the objects, enabling further applications, such as precise manipulation of liquid-filled objects. This is needed in robotics, automation, and scenarios closer to real-world deployment.

Figure 1.3 shows the process of finding the liquid fill level in RGB images. It is separated into two functions, the first being liquid edge prediction and the second the optimization loop.

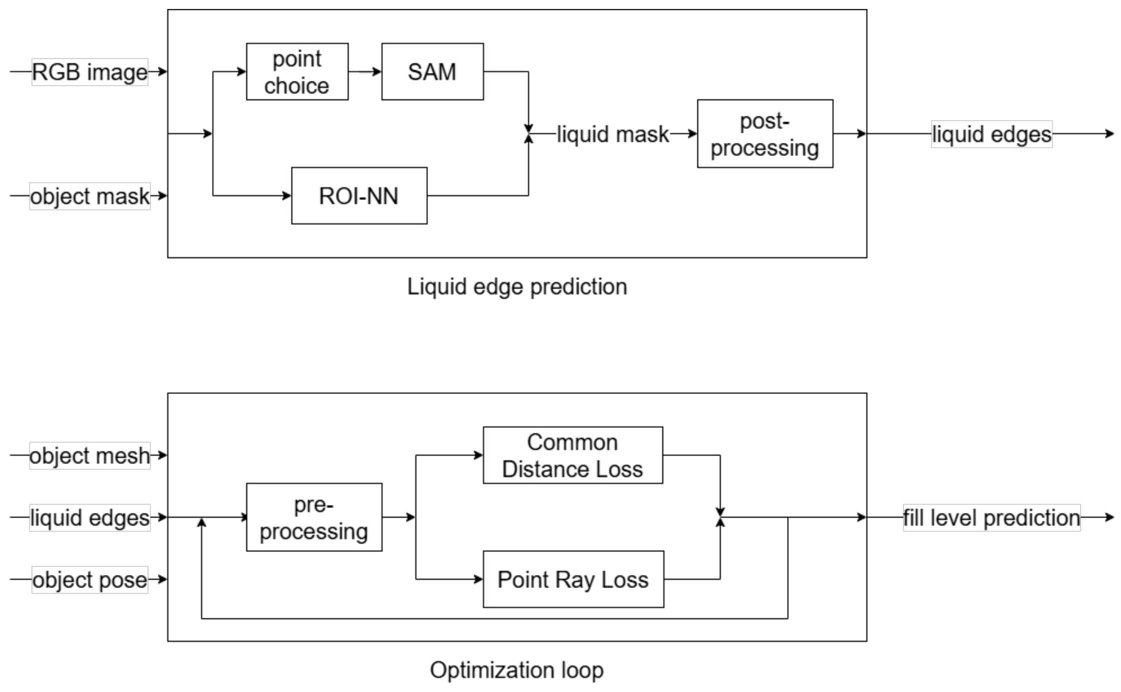


Figure 1.3: Flowchart

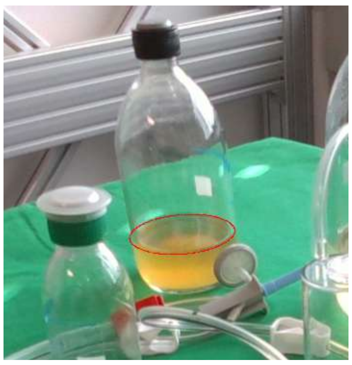
In liquid edge prediction, two NNs first predict liquid masks by receiving an RGB image and a binary object mask as input. The first NN, reffered to as Region of Interest Neural Network (ROI-NN) [16], predicts the liquid mask without any modifications. The second NN, Segment Anything Model (SAM) [17], requires the selection of two points: one inside and one outside the liquid area, to perform segmentation, which is done automatically. The resulting liquid masks are then combined for improved accuracy and undergo post-processing, including the use of a Canny filter [18], to obtain the liquid edges.

These edges are essential for the second part, the optimization loop, which also involves the usage of the object mesh and pose. The variable to be optimized is the liquid fill level. In the pre-processing step, an intersection between the object mesh and a plane is calculated. This plane is positioned at the height of the current liquid fill level estimation and is parallel to the ground, aligned with the gravity vector, representing the liquid surface. After projecting the resulting intersection to 2D, the loss functions are called. Their goal is to align the projection of the intersection with the previously predicted liquid edges.

The Common Distance Loss function tries to minimize distances between similar shapes along one direction and assures that similar curves are matched. This is achieved by examining the distances between all points of the edges in the object's z-direction. Points in similar shapes have similar distances to each other. Only the most common distance is used for calculating the loss value.

The Point Ray Loss function creates rays through each point of the intersection projection and the predicted edges and attempts to minimize their distances. This additional approach leads to higher accuracy in the results when combined with the Common Distance Loss.

The execution of the pre-processing and the loss functions is repeated until changes in the loss are minimal, at which point the final fill level prediction is returned. Figure 1.4a illustrates the resulting intersection projection overlaid on the original RGB image. Finally, using the predicted value, the liquid plane is represented in the 3D meshes of the objects, as shown in Figure 1.4b.


(a) MediumBottle 46, optimization result

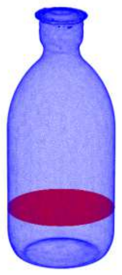

(b) MediumBottle 46, mesh with liquid plane

Figure 1.4: Preview of the results

This method demonstrates higher accuracy compared to other state-of-the-art approaches for determining liquid fill levels.

1.3 Thesis outline

Chapter 3 details the methods SAM and ROI-NN used to predict liquid edge images, followed by the description of the optimization loop, which finds the liquid fill level in the transparent containers with high accuracy using these edge images. The results are shown in Chapter 4 followed by a discussion of the difficulties in Chapter 5.

2 Related work

Determining the liquid fill level in transparent objects is accomplished through many methods, which depend on the specific use case, the available equipment, and knowledge of objects and poses. Before describing state-of-the-art methods for liquid fill level detection, the fundamental algorithms used for this purpose are explained. These basic methods include image contour extraction and image segmentation.

2.1 Image contour extraction

Image filters are the most common approach for extracting contours from images. The following sections introduce three widely used standard filters: The Sobel, Canny, and Laplacian of Gaussian (LoG) filter. [19] compares these approaches and highlights advantages and disadvantages of each.

2.1.1 Sobel

A Sobel filter [20] is a 3x3 convolution filter applied to an image to calculate the gradients of each pixel, representing the first derivative of the image. The gradients must be obtained separately in the x and y directions, for which individual filters are used.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.1}$$

The value of each current pixel is calculated by subtracting the intensities of its neighboring pixels. The resulting gradient increases with higher intensity changes between the neighboring pixels. If there is no change in intensity, the gradient of the current pixel is zero. By combining the results from the x and y filters, a final value is computed.

$$G_{Sobel} = \sqrt{G_x^2 + G_y^2} \tag{2.2}$$

2.1.2 Canny

A Canny filter [18] combines multiple filters and processing steps. The first filter applied is a Gaussian filter, which blurs the image to reduce noise. The formula for calulating the kernel is provided in Equation 2.3.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{2.3}$$

An example kernel with the dimensions 5x5 is provided in Equation 2.4.

$$G_{Gauss} = \frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix} \tag{2.4}$$

The larger the kernel size, the stronger the blurring effect, which results in less noise but also less detail in the original image. After blurring the image an edge detection filter, such as the Sobel filter, is applied to calculate the gradient image. This process identifies areas of high-intensity changes.

Next, maximum thresholding is employed to simplify the edges in the gradient image into thin lines. It compares neighboring pixels and only retains the ones with the highest gradient. Lastly, the intensity of the edges in the final image is determined by two variable thresholds: a low threshold and a high threshold. Pixels with gradient intensities bigger than the high threshold are retained, while those below the low threshold are discarded. Pixels that fall between the two thresholds are kept only if they are part of an edge that includes a pixel exceeding the high threshold, otherwise, they are removed.

By increasing the low threshold, the resulting edges are less detailed but more stable against noise. Raising the high threshold leads to fewer edges in the final image. This combination of different filters and processing steps is commonly used for edge detection because it produces effective results that are adjusted through the thresholds based on the requirements.

2.1.3 Laplacian of Gaussian

The LoG filter [21], like the Canny filter, begins by blurring the image using a Gaussian kernel. However, instead of focusing on gradients, which are the first derivative of the image, it uses the second derivative for edge detection, called the Laplacian. The first derivative shows gradient peaks for high-intensity changes, which leads to zero crossings in the second derivative. These zero crossings are then marked as edges, indicating the highest intensity variations, while automatically filtering out weaker ones.

2.1.4 Further contour extraction methods

In addition to the methods introduced, various approaches exist for different use cases with individual parameters. The authors of [22] and [23] present and discuss a collection of edge-detection methods. Besides the previously mentioned differentiation-based methods using no a priori knowledge, statistical methods, spatial-frequency-based methods, anisotropic diffusion-based methods, and more are introduced, along with others relying on known image information for better, but more application-specific edge detection.

2.2 Image segmentation

Many tasks in image processing require the segmentation of specific areas in an image. Depending on the requirements and resources, different approaches have been developed. [24] reviews state-of-the-art color-based segmentation algorithms, highlighting their reliance on color spaces and monochrome segmentation methods. Following this approach, the next section outlines segmentation techniques using grayscale images, followed by integrating color information for more accurate detection.

2.2.1 Segmentation using monochrome images

The major segmentation approaches for monochrome images include histogram thresholding, characteristic feature clustering, region-based methods, and fuzzy techniques, as discussed in [24]. Although edge detection is also used for segmentation, it is not covered in this section since it was explained earlier. Each of the other methods is outlined shortly in the following sections. A more comprehensive overview is presented in [24] and [25]. [26] and [27] further discuss advantages and challenges of the approaches.

Histogram thresholding

The basic concept of histogram thresholding is the selection of an intensity threshold in grayscale images to differentiate between the background and objects. This threshold is calculated automatically by creating a histogram based on the distribution of the gray levels of the pixels. In this histogram, the effectiveness of potential thresholds is evaluated using various criteria, such as within-class variance and between-class variance. Further information is presented in [28]. Other adapted approaches based on histogram thresholding are collected in [29].

Characteristic feature clustering

Before starting the segmentation process, a feature space needs to be defined. In this space, each pixel in the image is represented by a feature vector that includes information such as brightness, spatial coordinates, and texture. The goal of this approach is to identify regions within the image that best partition the feature space into distinct areas, called clusters. Various methods, such as hierarchical clustering, DBSCAN, and spectral clustering [30], are used to find these clusters, with k-means [31] being one of the most popular. K-means aims to minimize the distances between each pixel and the centroids of the clusters based on pixel intensities [32]. A commonly used metric for measuring distance is the Euclidean distance. By assigning pixels to the clusters with the smallest distance, pixels with similar characteristics are grouped [33].

Region-based method

The Region-based method divides an image into multiple homogenous regions. It utilizes three concepts: region growing, splitting, and merging. [24]

Region growing begins with randomly selected seed pixels. It further expands regions around these seeds by adding neighboring pixels with similar features [34]. Region splitting divides areas until they become homogenous, while region merging combines adjacent regions with similar features. These concepts are often combined to achieve the best results, such as using region growing and merging [35] or region splitting and merging [36].

Another popular region-based approach is graph-cut segmentation [37]. This method treats an image as a graph, where the pixels are the nodes. Each pixel is connected to its neighbors, and the connections, or edges, have assigned weights. There are two kinds of edges: n-links and t-links. N-links connect neighboring pixels, assigning a higher weight the more different the pixel's characteristics are. T-links base the weight on the likelihood of a pixel belonging to the for- or background.

The goal of graph-cut segmentation is to find an edge or cut in the graph that minimizes a cost or energy function, resulting in a path with minimal weight. This process results in similar pixels being combined to a cut between different regions. The optimal path is determined using min-cut/max-flow algorithms [38]. Multiple of these algorithms are explained in [39].

Fuzzy techniques

Fuzzy techniques [40] are utilized in the methods introduced in the previous sections. They are especially common in clustering, which is called Fuzzy C-Means (FCM) [41]. Instead of classifying each pixel into a specific cluster, FCM assigns a membership value that reflects the likelihood of a pixel belonging to that cluster. This approach is not only applied to clustering but also extends to other methods, by creating soft borders and memberships instead of "all or nothing" classification. As a result, a pixel may have a certain probability of belonging to an object and a complementary likelihood of belonging to the background.

2.2.2 Integration with color information

In colored images, using color information for segmentation is a straightforward strategy. To incorporate this into existing monochrome segmentation methods, it's essential to choose an appropriate color space. A color space defines how colors are represented and manipulated. [24] gives an overview of often utilized color spaces. The most known color space is RGB, primarily used for image representation. Another popular one is Hue, Saturation, and Intensity (HSI), which separates color and brightness making it invariant to conditions like shading, highlights, and shadows. Each color space focuses on different image aspects, making some more suitable for specific applications than others. [25]

Only minor adjustments are needed to integrate color into monochrome approaches. For example, instead of using histogram thresholding on monochrome images, it is applied to each component of the color space [42]. This technique is often combined with other methods, such as color histogram thresholding with region merging [43] or FCM [44].

In the case of clustering, the feature vectors are updated to also contain color information and region-based algorithms are processed similarly.

Despite the large number of color spaces, RGB remains the most common choice across these methods, as it is widely used for image representation and is easily understood by humans.

2.2.3 Segmentation using Neural Networks

A NN consists of nodes or neurons that are interconnected. In image processing, the first layer corresponds to the image pixels, each pixel represented by a node. These nodes are then combined to form subsequent layers. The deeper a network, meaning the more layers it has, the more complex data it stores and processes. Each layer abstracts features, starting with edge information in the early layers and progressing to shape recognition in deeper layers.

Each connection between nodes is assigned a specific weight learned by training the network on thousands of labeled images. The training process modifies all weights to minimize a loss function, which represents how well the network output matches the Groundtruth (see Section 2.4 for details). After training, the network identifies the features or areas it was trained on, even in unknown images.

All previously introduced segmentation methods produce semantic segmentation results, where each pixel is labeled with a specific class. A different approach, instance segmentation, distinguishes between objects within classes. NNs produce either result, depending on the data they are trained on. [45]

A common network structure used for semantic segmentation is the Fully Convolutional Net (FCN) [46]. Traditional NNs often rely on the simplest form of layers, which are fully connected layers. They connect each node in each layer to all nodes of the previous and next layer. A more complex implementation involves convolutional layers, which apply local filters with a small kernel. This approach enables a better spatial and hierarchical understanding of the image. Due to varying filter sizes, a convolutional layer often yields outputs of different sizes than the inputs. [47]

FCNs are composed of convolutional layers, instead of fully connected ones, leading to more complex image comprehension. The model structure allows for arbitrary input sizes while producing corresponding output sizes. FCN is the first network trained end-to-end on a pixel level, meaning that it contains all processing steps and directly maps an input image to an output segmentation. [46]

Mask Region-based Convolutional Neural Network (Mask R-CNN) [48] is an example of a network that generates instance predictions. The network is based on a Region-based Convolutional Neural Network (R-CNN), which extracts region proposals as bounding boxes in images. For each region, the Convolutional Neural Network (CNN) then calculates the features and uses them to classify each region. [49] As a result, R-CNN returns bounding boxes for each object along with corresponding region labels. Mask R-CNN additionally returns binary segmented masks for each Region of Interest (ROI) individually, allowing for accurate predictions of the shapes of the regions within these bounding boxes. [48]

Further NNs used for segmentation are collected in [50], [45], and [51].

SAM [17] is a NN for segmentation that offers extra input prompts, like bounding boxes, points, or masks, for user-specified segmentation. This makes it more flexible for selecting areas to segment, without requiring training on task-specific images, such as detecting only certain objects or classes.

2.2.4 Segmentation using depth estimation

Multiple approaches using deep learning and NNs extract depth information from single RGB images. Many papers cover and overview such methods, such as [52], [53], and [54]. Popular methods for depth extraction include MiDaS [55], Depth Anything [56], and Monodepth [57].

Monocular depth estimation is a challenging task because extracting depth typically requires a stereo view. However, recent NNs and other deep learning approaches keep getting more complex and deeper, due to more data they are trained on. These advancements lead to increasingly accurate depth estimations, helping to differentiate multiple objects from the background.

In the depth images produced by these methods, objects in the foreground and background appear in different colors or pixel intensities. This characteristic is used to segment objects from the background by grouping and filtering similar features. In many approaches, the estimated depth is combined with other methods to achieve better results, like in [58], [59], and [60].

2.3 Detection of liquid in transparent containers

Liquid detection in transparent containers is either accomplished using solely RGB images or by employing more advanced techniques.

2.3.1 RGB-based approaches

There are two main methods for determining the liquid fill level in transparent containers: extracting edges related to the liquid and its surface or segmenting the entire liquid.

Using edges

In an image, various edges are present, related to the object, the liquid, the background, and any other objects within the scene. The most commonly used edge detection algorithms in many methods are Canny [18] (such as in [8], [12], [13], and [10]), and Sobel (in [5] and [6]). However, LoG [21] and Infinite Symmetrical Exponential Filter (ISEF) [61] are also utilized in [8]. Extracting only the edges that correspond to the liquid fill level, which often appear very faint, is a challenging task.

Many approaches address this issue by setting strict restrictions. [8], [5], [6], [12], and [13] apply edge detection, however, they all record the transparent containers only from one specific angle, restricting the edge detection to a single horizontal line. Often, surrounding lighting is adjusted to improve the clarity of the edge indicating the liquid fill level. [10] allows for a larger variety of angles, however only in the vertical direction, with containers centered in front of the camera. This transforms the search for the liquid fill line into a search for liquid ellipses. Furthermore, the approaches typically have plain, clean backgrounds that do not change between each detection.

[5] and [6] detect the fill level in hospital infusion bottles by detecting moving lines by comparing consecutive images. Due to the angle, lighting, and background remaining constant, only one line visibly changes between the pictures, which is the liquid surface edge.

The accuracy of detection varies between the different methods. Approach [8] only determines whether a container is overfilled or underfilled in the beverage industry. Due to the never changing position of the checked container, a ROI is defined as a small area around the desired fill level. This ensures accurate results but is limited to that specific region. Without a strict ROI, the results are often very rough, like in [13], where the outcomes are categorized simply as full, 75% full, half-full, and empty.

All approaches that utilize edge detection for liquid fill level detection yield 2D results in the image. If the viewing angle and the container measurements are known, 2D result is converted into a 3D result, such as in [12]. However, this conversion depends on having complete knowledge of the container's details and camera positioning.

Using region-based approaches for segmentation

[62] and [63] use the graph-cut method to determine a border between liquid, solid, and air. In liquid detection, the edge created must separate liquid and air. However, for unclear edges or disturbances due to reflections on the container or intersecting objects, the method often returns incorrect results. This occurs because the path with minimum energy does not necessarily align with the actual edges of the liquid.

Using NNs for segmentation

Many NNs trained for detecting liquid levels only return the fill level in intervals, such as [64], [65], and [9]. However, a few approaches have been developed to train NNs to segment liquid fill levels and produce binary masks. [15] presents three approaches trained on laboratory images: the first uses semantic segmentation, the second instance segmentation, and the third a hierarchical combination of both. They utilize the previously introduced FCN for semantic segmentation and mask R-CNN for instance segmentation. Besides differentiating between objects and their contents, the NNs accurately identify different types of contents, such as liquid, foam, or suspension.

[16] extends the previous work by introducing a ROI as an additional input, which is the 2D mask of the objects in which the liquid level is detected. By specifying the area of the transparent container, the detection accuracy is further improved. The results of these networks, however, are all limited to 2D segmentation masks.

Another approach [14] not only segments liquid and object masks through semantic segmentation but includes a 3D shape estimation of the liquid and the object's shape. It is one of the first NNs to attempt such a prediction.

2.3.2 Non-RGB approaches

Besides liquid fill level detection using RGB images, other technologies are also employed for this task. A few of them are combined with RGB images, such as depth images in [66]. Other approaches use contactless methods, like lasers or measurements related to changes in light and reflections ([67], [68], and [69]). Pressure ([70]), radar ([71]), and acoustic Time-of-Flight (ToF) signals ([72]) are also used in applications. While contactless methods are often preferred because they do not disturb the liquid, some approaches use sensors positioned either inside or outside the liquid ([73] and [74]).

2.4 Optimization algorithms

Optimization is a very complex topic that many books cover, such as [75] and [76]. A small subset of algorithms that are particularly important for this thesis is introduced in the following sections.

2.4.1 Gradient descent approach

Gradient descent is a method used to minimize a function, often applied to the loss function when training NNs.

The approach begins by setting up a model with one or multiple undetermined parameters. The initial values of these parameters are chosen based on the application. Once the model is executed, a loss value is calculated to measure the difference between the model's output, and the Groundtruth. The key aspect of this optimization method is not the loss itself but its gradient, which indicates how the loss changes with respect to each parameter. The gradient points in the direction of the steepest loss increase, which means that for minimization purposes the negative gradients are used to update the model's parameters. [77] [78]

This update process is called backpropagation, where the weights and parameters of the model are tuned to ensure that the output resembles the Groundtruth more closely. The learning rate specifies the extent to which these parameters are updated. This process, of calculating the model's result and then backpropagating the difference, is referred to as a step. With each step, the parameters are tuned again, progressively aligning the model's output with the Groundtruth. This iterative process continues until the loss falls below a specified threshold, shows minimal change or a predefined number of steps is reached. [77] [78]

A disadvantage of the gradient descent approach is that it most algorithms find local minima rather than global minima [78]. Once a minimum is reached, the algorithms do not change to a more optimal minimum. Additionally, gradient descent requires continuous gradients, which are not supported by all functions in every library. The biggest advantage, however, is that by following the negative gradient, the parameters are efficiently adjusted to converge faster to a minimum.

An example of a gradient descent optimization algorithm is Adam [79] It is widely used because it adapts the learning rate for each parameter based on the frequency of the parameters' updates, making it an efficient method. Other algorithms are introduced in [77] and [78].

2.4.2 Non gradient approaches

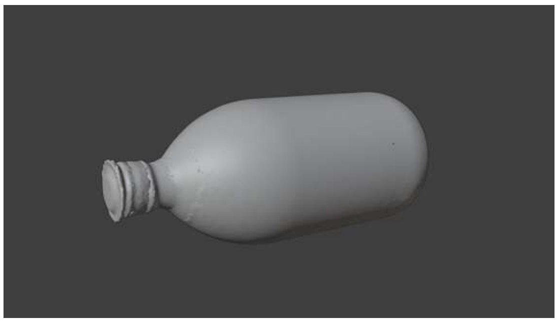
Not requiring gradient tracking is advantageous because the model is not restricted to specific functions and libraries. However, this freedom makes approaches often more complex than gradient descent methods [80]. There is a variety of approaches available, such as Particle swarm optimization [81] and Monte Carlo Sampling [82], which optimize to global minima, and the Nelder-Mead method [83], resulting in local minima. A detailed explanation of various non-gradient descent methods would extend the scope of this thesis. More approaches are introduced and outlined in [80].

3 Fill-Level estimation in known containers

This thesis requires prior knowledge of three components to optimize the detection of liquid fill levels in transparent objects within RGB images: the 3D objects in the scenes, their poses in each RGB image, and the gravity vector.

The data utilized in this thesis is sourced from the project titled "Depth Estimation of Transparent Objects" [84], which fulfills these requirements. This dataset consists of various scenes with known 3D meshes of the objects (Figure 3.1), their 3D poses, as well as RGB images (Figure 3.2a) accompanied by 2D masks for each object in every scene (Figure 3.2b). The intrinsic parameters of the camera are also provided, enabling the conversion from 3D to 2D.

The focus of this thesis is on scene "j_005" from the dataset (an example image of the scene is shown in Figure 3.2a), which includes liquid-filled BigBottle (Figure 3.1a) and MediumBottle (Figure 3.1b). The dataset includes both large and small formats of the 3D meshes, but the smaller versions are selected for both objects for faster calculations.


(a) Small 3D mesh of BigBottle

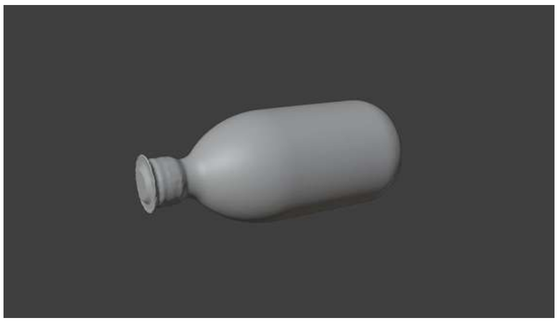
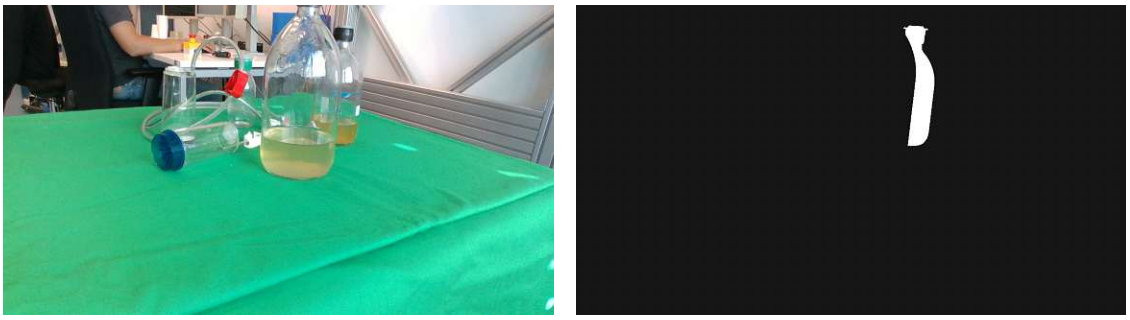

(b) Small 3D mesh of MediumBottle

Figure 3.1: Small 3D object meshes

Before beginning the optimization process, it is essential to predict the liquid edge images. These images represent the edges in the containers related to the liquid and serve as important indicators for the liquid fill level. Once the liquid edge images are predicted, the optimization loop, along with its loss functions, is defined. The output of this loop is the optimized fill level, which is then used to accurately add a liquid plane at the optimized positions within the 3D meshes of the objects. All these steps will be explained in detail in the following sections.

3.1 Predicting liquid edge images

To obtain liquid edge images, several approaches are possible, as discussed in (link to related). The methods that have proven to be the most effective in this realistic setup are the use of SAM [17] and ROI-NN [16], a NN utilizing the object masks for semantic and



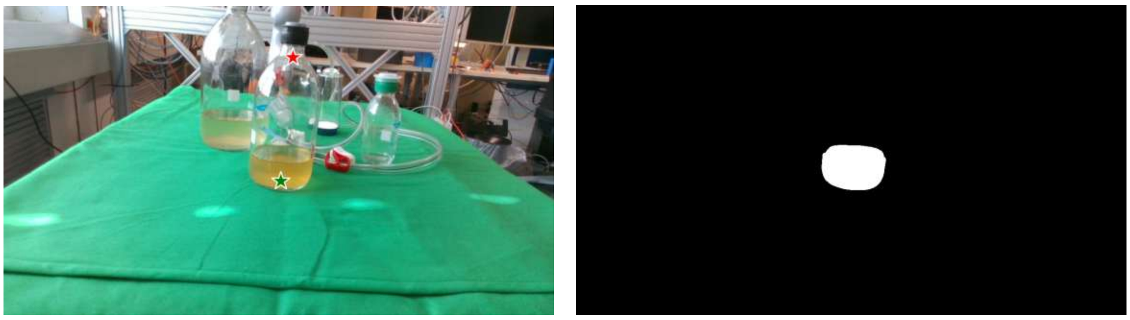
(a) RGB image 1 (b) MediumBottle object mask 1

Figure 3.2: RGB image 1 of the scene j_005 with the respective object mask of Medium-Bottle

instance segmented liquid mask prediction. Both approaches are explained shortly in the following sections.

3.1.1 Predicting liquid edge images using Segment Anything Model

SAM functions by predicting a liquid mask based on the definition of two points: one point is located inside the liquid, while the other is positioned on the part of the object that is not in contact with the liquid. Additionally, labels indicate which point is inside the liquid (marked in green) and which is outside (marked in red). When these points are defined correctly, SAM generates a mask that segments the liquid area. Figure 3.3a illustrates the placement of these two points, while Figure 3.3b displays the resulting predicted liquid mask.



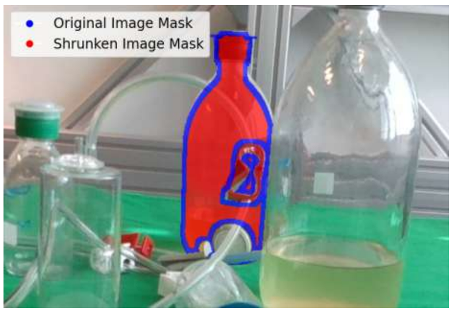
(a) MediumBottle 19 with points in liquid (green) and outside of liquid (red)

(b) MediumBottle 19 liquid mask prediction by SAM

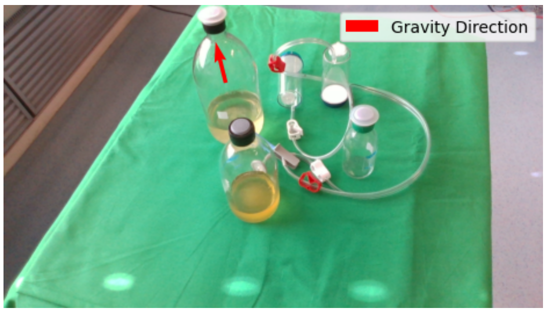
Figure 3.3: RGB image 19 of the scene j_005 with points for SAM segmentation and the predicted liquid mask of MediumBottle by SAM

Points selection

The SAM algorithm is applied to the original RGB image, utilizing surrounding information to distinguish between the background and the contents of the container. Points are selected on the object mask, which is shrunk by 10 pixels to prevent points from being too close to the mask's border and allow for slightly inaccurate object masks (Figure 3.4a).



(a) MediumBottle object mask 5, shrunk by 10 pixels



(b) BigBottle 91 with gravity vector shown in red

Figure 3.4: Processing steps for the point choice for SAM

To locate the liquid within the container, the 3D gravity vector is projected onto the 2D image (Figure 3.4b). The object mask is then rotated so that the gravity vector aligns vertically with the image's y-axis. Figure 3.5 illustrates both the original and rotated gravity vectors, along with the corresponding object masks and the selected segmentation points on the rotated mask.

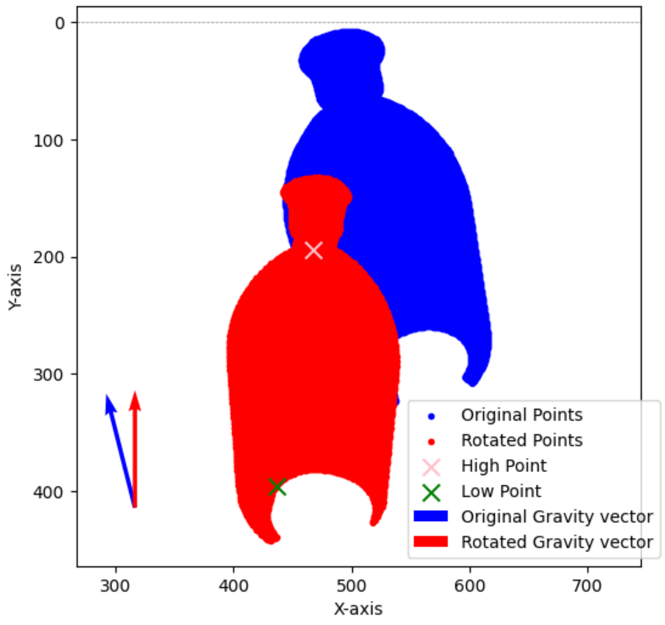


Figure 3.5: BigBottle 91 original and rotated image mask, original and rotated gravity vector and chosen segmentation points

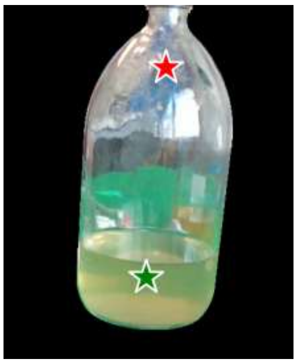
This ensures that the liquid is positioned at the bottom of the object mask in the vertical direction, regardless of the object's orientation. Consequently, the point representing the liquid is located at the lower part of the mask, while the other is at the upper part. A height of 0 corresponds to the top of the mask, meaning higher points have a lower height value. The positions of these points on the rotated masks are defined as follows:

For MediumBottle, the y-coordinate of the upper point is set at 20% of the object mask height, while the lower point is at 95%. Similarly, for BigBottle, the y-coordinate of the upper point is positioned at 20% of the object mask height, and the lower point is at 85%.

Due to the previous rotation, the coordinates of the object masks are floating values, which means a specific percentage of the mask height might not correspond to a valid coordinate point. Additionally, if the object is covered in that area, there may be no existing points at the specified height. Therefore, after calculating the exact value at the mentioned mask height, the closest available y-coordinate from the existing mask is used for further calculations.

The x-coordinate for the points is determined at 50% of the mask width at the specific y-coordinate calculated earlier. The rows are defined with a slight margin to accommodate floating values. Again, the closest valid x-coordinate is selected to ensure the point lies on the object mask.

The defined points are then rotated back to fit the original image (Figure 3.6a). If the object is partly covered, the points are adjusted automatically to ensure they lie on the object mask (Figure 3.6b). The y-coordinate of the points needs to be modified for different containers since bottle caps and other object parts might distort the results. The optimal value for the best prediction results also depends on the shape and size of the objects. For instance, positioning points too low in BigBottle occasionally result in only segmenting the ground of the object rather than the entire liquid due to a color difference. This is why the point in the liquid is positioned higher than for MediumBottle (Figure 3.6).


(a) BigBottle 1, segmentation point placement

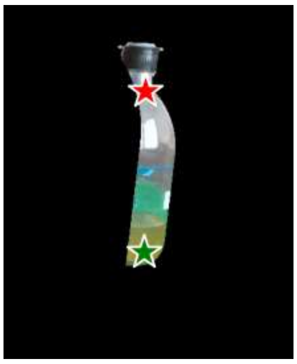

(b) MediumBottle 1, segmentation point placement

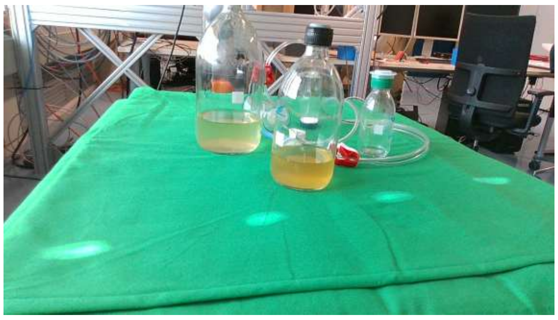
Figure 3.6: Segmentation point placement on containers

3.1.2 Predicting liquid edge images using Region of Interest Neural Network

ROI-NN takes an RGB image and an object mask as input to predict the liquid mask. It generates semantic and instance liquid masks, which are discussed in (link to related work).

Paper [85] explains, that semantic segmentation provides a coarser detection by only identifying different classes (for example, vessel and liquid), while instance segmentation recognizes each object and component as a unique instance. In this thesis, only the "liquid" result of both outputs is used. Although semantic and instance predictions show similar results, instance segmentation achieves slightly better accuracy due to its finer segmentation. Therefore, instance segmentation will be referenced in the following explanations and sections whenever ROI-NN is mentioned unless stated otherwise.

An advantage of this method over SAM is that it does not depend on any points, but uses solely the pre-trained information for prediction. However, the resulting liquid masks tend to have more noise outside the liquid than when using SAM, as pictured in Figure 3.9c. A successful prediction result, along with the original image, is shown in 3.7b.


(a) RGB image 20

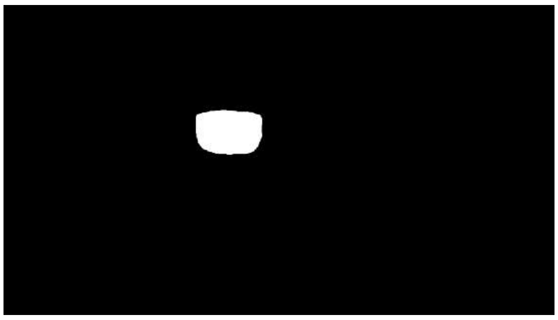

(b) BigBottle 20 liquid prediction by ROI-NN

Figure 3.7: RGB image 20 of the scene j_005 and the predicted liquid mask of BigBottle by ROI-NN

3.1.3 Post-processing the liquid mask

Both prediction methods do not consistently produce clean segmentation masks, which complicates the optimization process. Such issues arise when objects or parts of objects occlude the liquid in the container. For instance, in Figure 3.9a, a cable interferes with the liquid mask, resulting in gaps in the predicted liquid mask. While it is possible to remove interfering objects identified as such, the cable is not included as an object in the object mask and remains undetectable. If the point used for segmentation by SAM lies on the cable, detection fails, as shown in Figure 3.8.

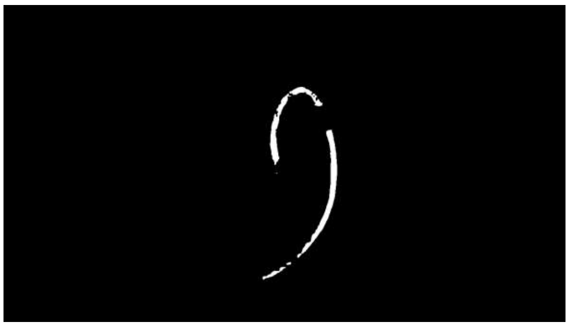


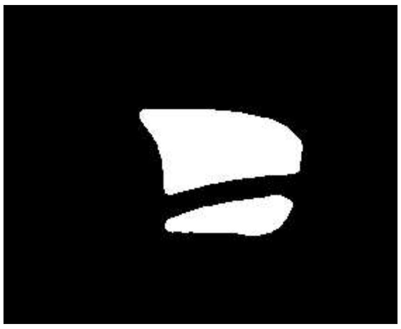
Figure 3.8: BigBottle liquid mask 57 segmented cable

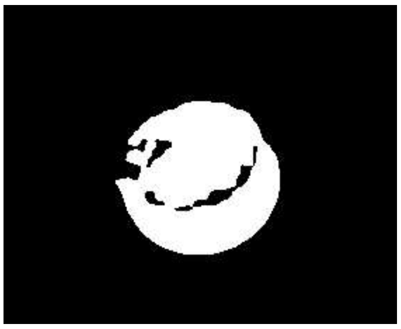
Similarly, ROI-NN is partially disturbed by objects, as shown in the prediction in Figure 3.9d, where one side of the mask is noticeably higher than the other.

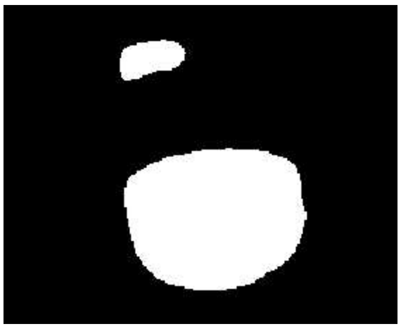
In other cases, such as in Figure 3.9b, the liquid surface is not segmented clearly, resulting in jagged edges unsuitable for optimization. To eliminate such liquid-unrelated or disturbing information, post-processing is necessary.

To evaluate the stability and accuracy of the liquid edge detection, three approaches are used: Using the predicted liquid masks from SAM, from ROI-NN, and both methods combined.

To visualize all post-processing steps, the following liquid masks in Figure 3.9 are presented for visualization.


(a) BigBottle SAM liquid mask 12, intersected by an unmasked cable


(b) MediumBottle SAM liquid mask 96, unclean surface segmentation


(c) MediumBottle ROI-NN liquid mask 42, noise outside the liquid

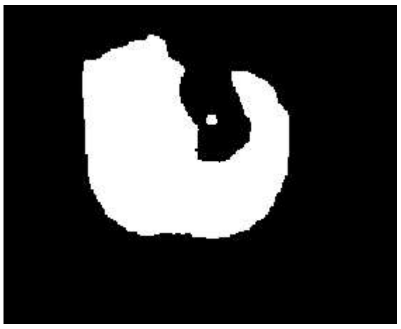

(d) BigBottle combined liquid mask 58, disturbance by overlapping unmasked objects

Figure 3.9: Unclean liquid masks in different containers

Removing noise in the liquid mask

The predictions, especially those from ROI-NN, produce noise outside the liquid, disrupting the optimization process. The first post-processing step is to remove this noise by assessing the sizes of each separated area in the segmented images. Only areas bigger than 0.25 times the size of the largest area are retained for the following process. The reason for not only keeping the largest area is that liquid masks intersected by objects might result in multiple parts holding relevant information about the liquid fill level.

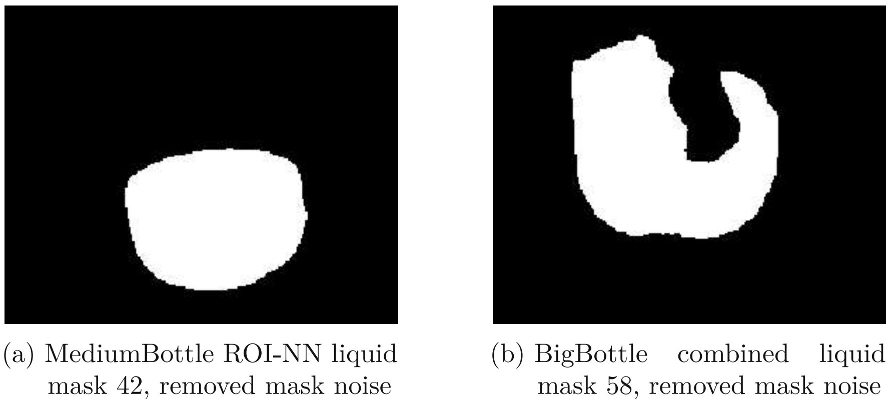

(a) MediumBottle ROI-NN liquid mask 42, removed mask noise
(b) BigBottle combined liquid mask 58, removed mask noise

Figure 3.10: Small parts of the masks predicted by ROI-NN and the combination of both methods removed

Morphologically closing the liquid mask prediction

A crucial post-processing step eliminates unnecessary liquid-related information by filling gaps in the liquid masks using morphological closing [86]. In Figure 3.11, the closed areas are highlighted in grey on top of the liquid masks.

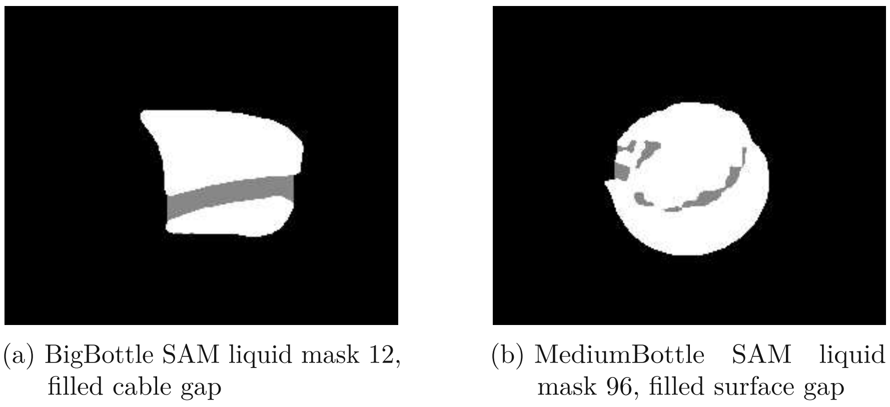

(a) BigBottle SAM liquid mask 12, filled cable gap
(b) MediumBottle SAM liquid mask 96, filled surface gap

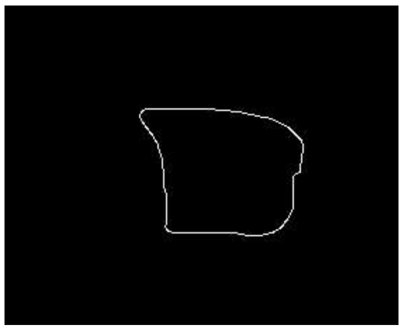
Figure 3.11: Morphologically closing the liquid masks

Canny filtering

A Canny filter [18] transforms the liquid mask into usable edges for optimization. Since the liquid mask is binary, the choice of threshold values does not significantly affect the expected outcomes. Figure 3.12 illustrates the results.

Removing border edges

Edges close to the object's border or outside of it are not significant for determining the liquid fill level, so they are removed. The same applies to edges that might appear due to known objects intersecting the mask. This is accomplished by shrinking the object mask by 10 pixels and eliminating all edges outside, as seen in Figure 3.13.


(a) BigBottle SAM liquid mask 12, canny edge image

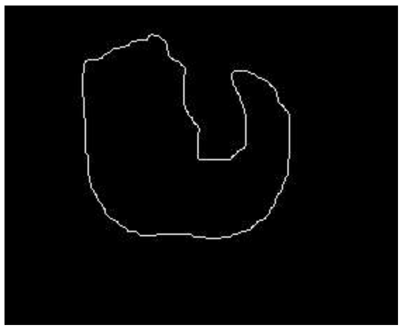
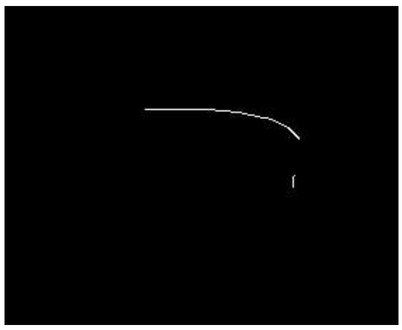

(b) BigBottle combination liquid mask 58, canny edge image

Figure 3.12: Edge images after applying a Canny filter to the liquid masks


(a) BigBottle SAM liquid mask 12, fill level related edges

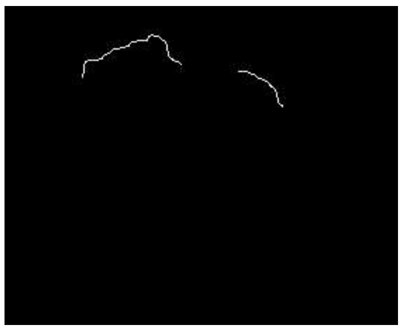
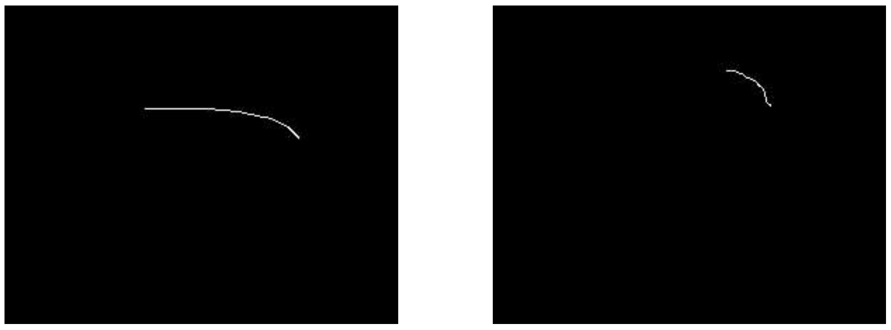

(b) BigBottle combination liquid mask 74, fill level related edges

Figure 3.13: Fill level related edges, accomplished by removing edges outside the object mask

Filter noisy edges

In the remaining edges, there are still outliers, noise, and other very short edges that interfere with the optimization process. Due to connected and incorrectly segmented areas, such as those shown in Figure 3.13b, noisy edges outside the liquid persist even after the initial post-processing step, which aims to remove noisy mask parts. These edges are removed by retaining only those at least 0.25 times the length of the longest edge. Performing the optimization on all of the remaining edges leads to instability and potentially wrong alignment, as they often differ in height. Therefore, a final processing step is necessary.

For ROI-NN and the combined method, noisy edges within the liquid are less likely than outside. Consequently, the final edge used for optimization is the lowest remaining edge, again aligned with the gravity vector to account for the tilting of objects or the camera. The result is visualized in Figure 3.14.


(a) BigBottle SAM liquid mask 12, filtered edges

(b) BigBottle combined liquid mask 58, filtered edges

Figure 3.14: Final edges after removing noise and other disturbing edges

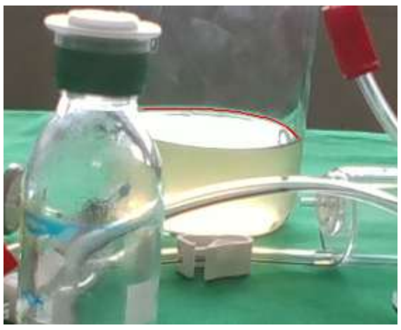
Overlaid results

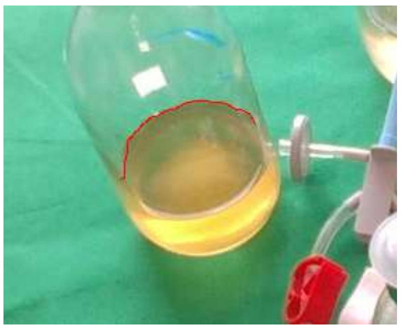
Figure 3.15 shows the resulting edges overlaid with the original RGB images.

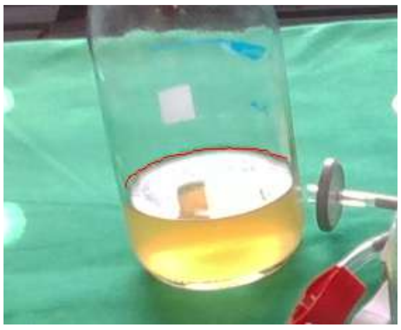
If the mask does not contain valid edges, the liquid edge image appears entirely black. Without valuable information, it is excluded from the optimization process for determining the height of the liquid fill level.

3.2 Height Optimization in Containers

The variable being optimized is the height of the liquid in the transparent container, which corresponds to a value along the z-axis of the object. The coordinate systems of the objects are sometimes misaligned with the objects themselves, meaning the z-axis may not point in the intended direction (see Figure 3.16a). For instance, in the case of BigBottle, a transformation matrix is applied to adjust the coordinate systems (see Figure 3.16b). This transformation matrix must be individually created for each object in advance and was generated using MeshLab [87] for this project.


(a) BigBottle SAM liquid mask 12, final liquid related edges RGB overlay


(b) MediumBottle SAM liquid mask 96, final liquid related edges RGB overlay


(c) MediumBottle ROI-NN liquid mask 42, final liquid related edges RGB overlay

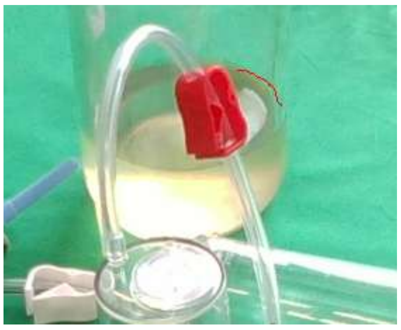
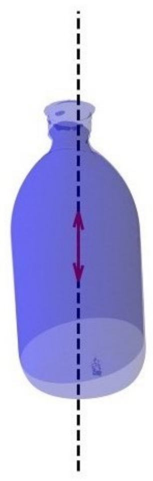

(d) BigBottle combined liquid mask 58, final liquid related edges RGB overlay

Figure 3.15: Final liquid fill level related edges overlaid with the original RGB images


(a) BigBottle unaligned z axis

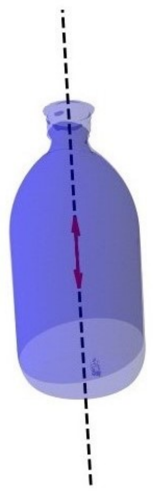

(b) BigBottle aligned z axis

Figure 3.16: Comparison of the z axis of the object coordinate system before and after alignment

3.2.1 Optimization Loop

After refining the optimization direction, the optimization loop is executed on the z-coordinate within the container. For this optimization, an Adam optimizer [79] is utilized, which tracks gradients to achieve more accurate and faster optimization results.

For each current z-coordinate, starting from zero, a new coordinate system with the following properties is created: the z-axis is aligned with the gravity vector, which, in this dataset, points in the same direction as the initial system. The x-axis points away from the camera system's origin, projected to have a 90-degree angle to the z-axis, while the y-axis is determined using the other two axes. The origin of the new coordinate system is located at the current z-coordinate. For visualization, the newly defined coordinate system is shown alongside the initial coordinate system and the camera coordinate system in Figure 3.17 and within the container in Figure 3.18.

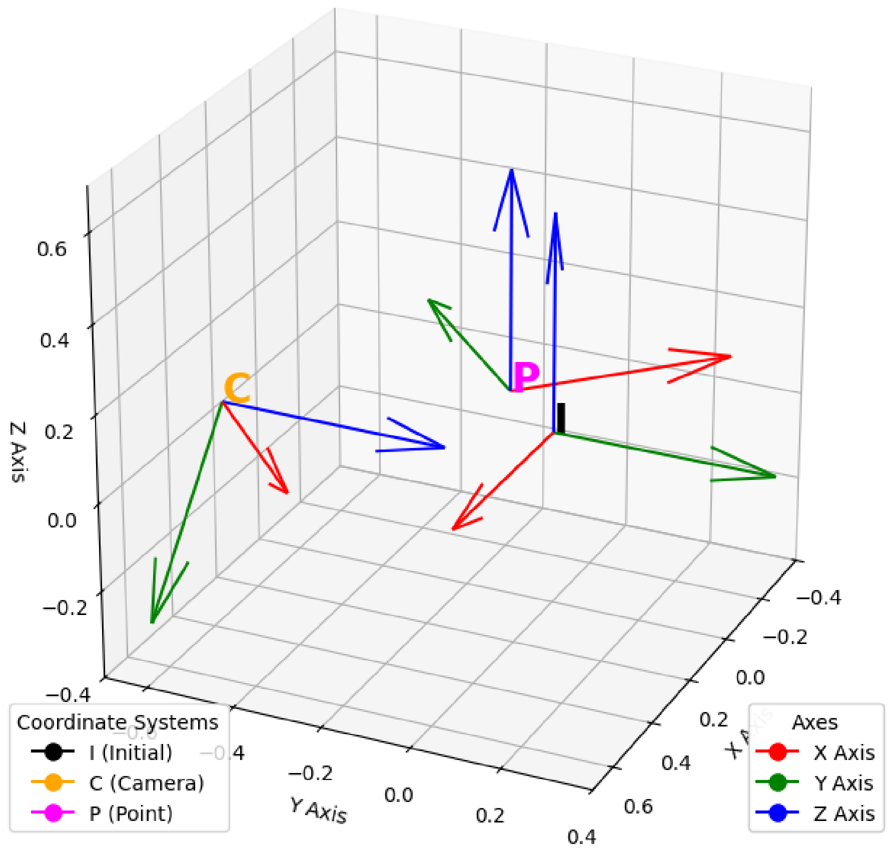


Figure 3.17: BigBottle RGB image 16: Initial, Camera and Point coordinated system

By aligning the z-axis with the gravity vector, the liquid must lie in the xy-plane of the new coordinate system. To determine the edges that would be formed by the liquid at that height, the intersection of the plane with the 3D mesh of the object is calculated.

Existing mesh-plane intersection functions [1] are not differentiable and cannot be used in this implementation. Due to the absence of a differentiable version of this function,

[1] for example from Trimesh [88]

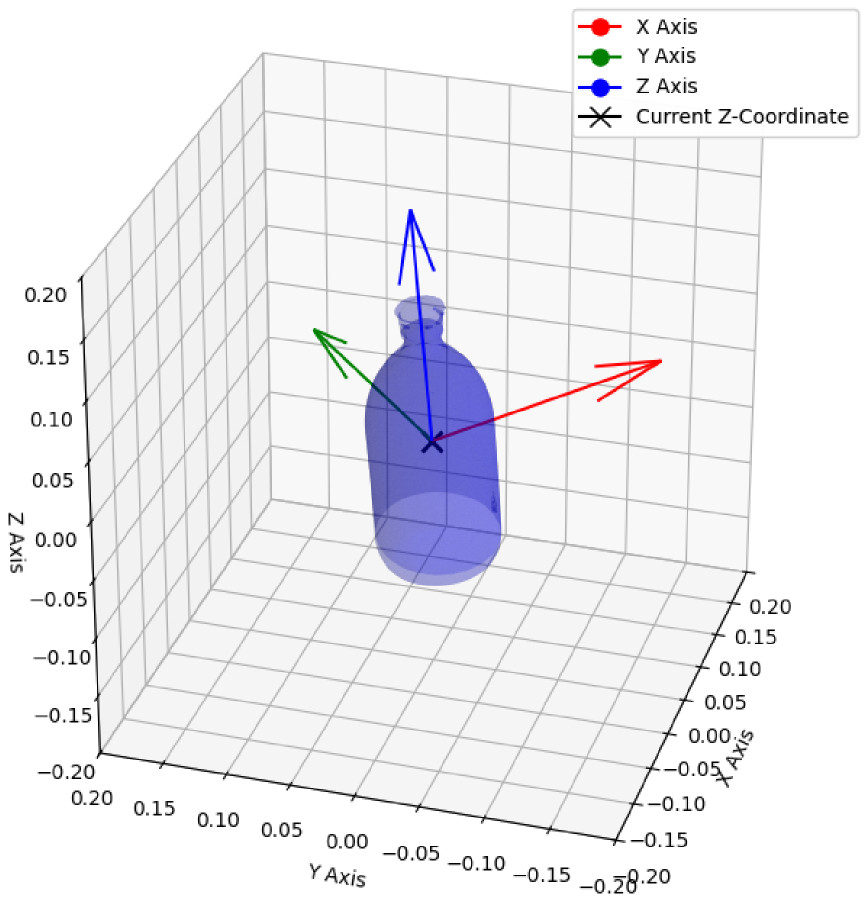


Figure 3.18: BigBottle RGB image 16: Point coordinate system in Container

existing functions have been adapted into a differentiable form without changing their functionality. Given the plane origin $\mathbf{o}_{\text{plane}}$ (which is the origin of the new coordinate system in this case) and a normal vector $\mathbf{n}_{\text{plane}}$ (which represents the z-axis of the new coordinate system), the function determines for all vertices $\mathbf{v}$ whether they lie above or below the plane with the following equation:

$$\mathbf{d} = (\mathbf{v} - \mathbf{o}_{\text{plane}}) \cdot \mathbf{n}_{\text{plane}} \tag{3.1}$$

Here, $\mathbf{d}$ is an array where values are positive if the corresponding vertex is above the plane and negative if it is below. For each side of each triangle, the function checks the signs of the vertices. If there are different signs for any line, it indicates that there is an intersection. Using interpolation it finds the exact intersection points, as shown in Figure 3.19.

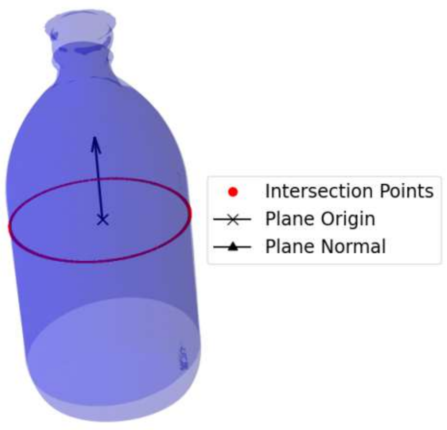


Figure 3.19: BigBottle Mesh-Plane intersection

This intersection is projected onto a 2D image using the intrinsic values. In the case of a bottle, it appears either round or elliptical. Figure 3.20 illustrates the intersection in red over the object mask.

If no intersection occurs or none is visible on the object mask, the current z-value is projected instead. This projection serves as a reference point for optimization and aids in guiding the z-value back into the container.

3.2.2 Loss Functions

The projected intersection or the z-value is further compared to the predicted liquid edges calculated earlier, using two loss functions: Common Distance Loss and Point Ray Loss.

Common Distance Loss

Due to the shape of the predicted liquid edges typically being only a segment of the projected intersection points, using a standard minimal-distance loss leads to misalignments.

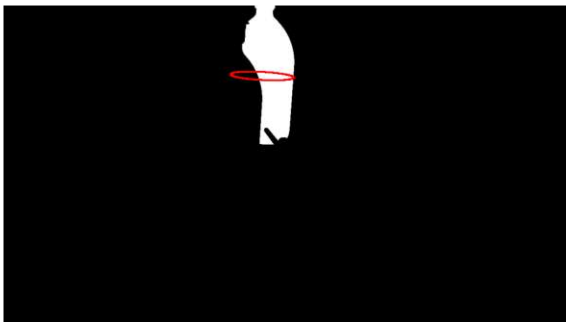


Figure 3.20: BigBottle mask 16, Projection of the mesh-plane intersection to 2d

When aligning an ellipse with a half ellipse, the half ellipse often ends up centered within the ellipse because the points from the top and bottom both try to minimize the distance, resulting in a compromise. Therefore, a better and more stable loss function is needed in this case.

The Common Distance Loss method considers the shape of the edges during alignment. It first projects the z-axis of the container to the 2D RGB image. For each point of the predicted edges, the method searches for intersection points in that direction, within a small margin. Each predicted point corresponds to only a few matching intersection points. Figure 3.21 illustrates the z direction and all matching intersection points within that direction.

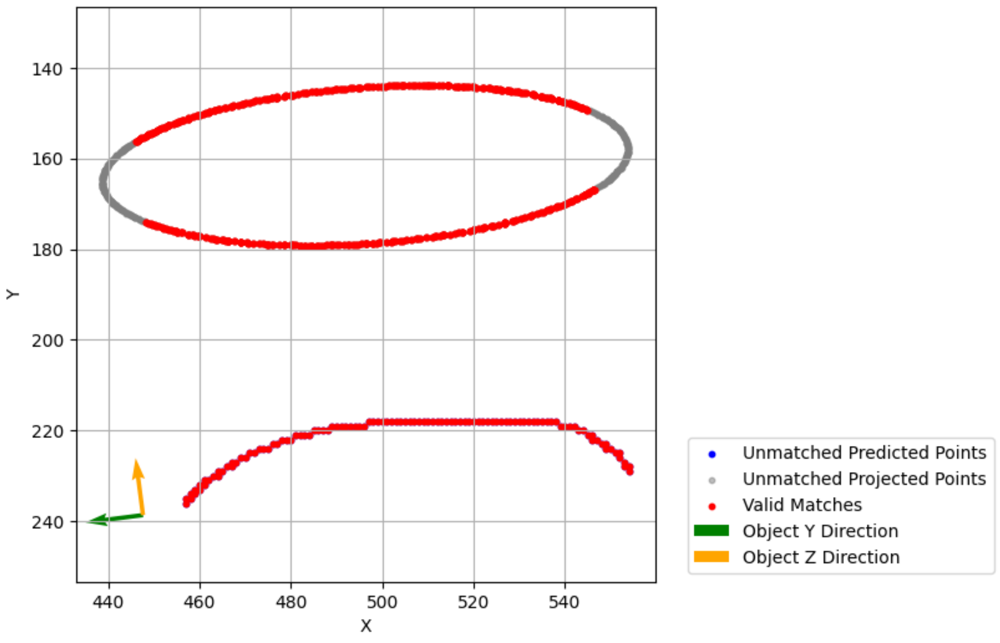


Figure 3.21: MediumBottle 46, SAM, points in the Object Z Direction

Next, the direction toward these matching points is calculated. The key aspect is that distances between different points, calculated in this way, are similar only if the shapes of the edges are similar. Distances from the projected edge to the upper intersection edge are more consistent than distances to the lower intersection edge.

To achieve optimal alignment, the goal is to minimize only those similar distances, which aligns the predicted edge with the upper projected edge in this case. This is done by calculating the most common distance within a defined threshold and returning it as the loss to be minimized. Points that correspond to this most common distance are shown in Figure 3.22.

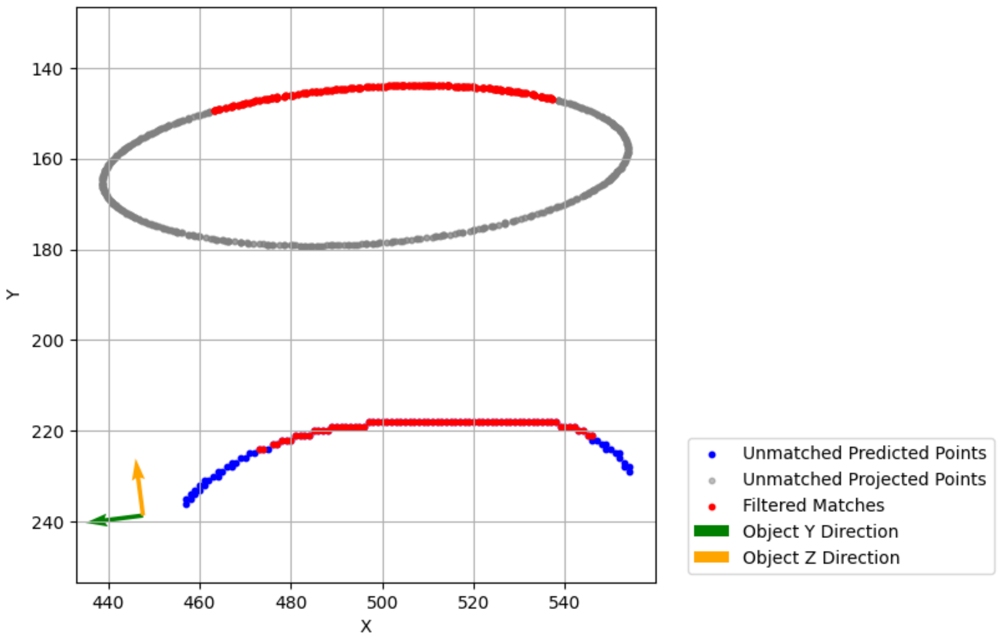


Figure 3.22: MediumBottle 46, SAM, points with the most common distance

Point Ray Loss

In the second loss function by [89], rays are generated from the predicted 2D liquid edge points and the unprojected 3D intersection points. The minimum distance between each prediction ray and its corresponding minimal intersection ray is used as the loss function, which is then minimized. This loss is employed for more precise optimization.

Loop termination

The optimization loop runs for a maximum of 300 iterations. However, it terminates early if one of the following conditions is met: if the loss does not change by more than a specified threshold over the last 10 iterations, or if the loss falls below a certain threshold. Figure 3.23 displays an example of the optimization results on the corresponding RGB image.

3.3 3D Model with Liquid Plane Creation

The creation of the model closely resembles the optimization loop, with the key difference that the z-coordinate is not tracked or optimized. Instead, the previously determined

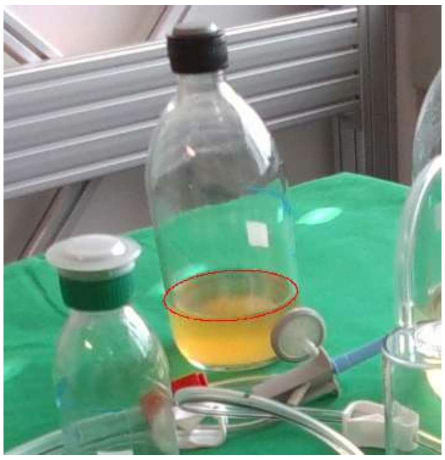


Figure 3.23: MediumBottle 46, result of the optimization using SAM

optimal z-value is used for calculations. After performing the 3D mesh-plane intersection, the intersection points are connected, resulting in a flat surface within the mesh. This is achieved by calculating the average value of the intersection points and then sorting these points by angle. Delaunay triangulation [90] generates a mesh from all these points, resulting in the desired output. The plane is illustrated in red within the container mesh in Figure 3.24.

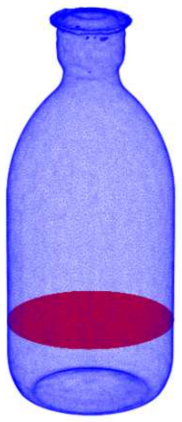


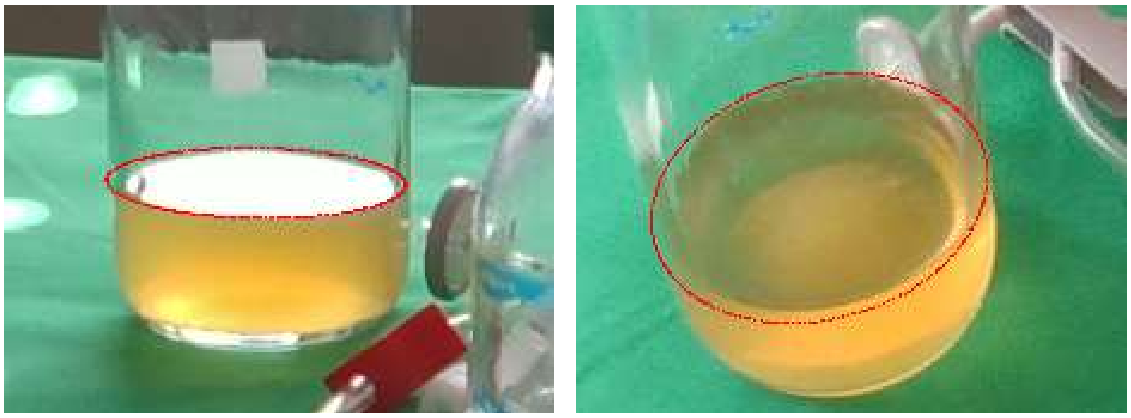
Figure 3.24: MediumBottle RGB image 46: Mesh including the liquid plane at the optimized position

4 Results

This section presents the overall results of this thesis, further difficulties are discussed in Chapter 5.

4.1 Groundtruth

To analyze the results, a Groundtruth liquid fill level for each object needs to be established. Due to the original liquid in the bottles not being available for measurement, a height within the object mesh was manually selected, to match the liquid level visible in the images. However, due to slight errors in the poses of the objects and the effects of refraction, achieving perfect alignment for the same height in all images is not feasible. Figure 4.1 shows an example of a well-aligned result and a distorted result for MediumBottle.



(a) MediumBottle 12 Groundtruth good alignment (b) MediumBottle 66 Groundtruth slight offset in alignment

Figure 4.1: Chosen Groundtruth in MediumBottle

The height of the objects used in this thesis, measured from the bottom to the bottle cap, was identified as 20cm for BigBottle (ranging from -10.5cm to 9.5cm from the origin of the coordinate system), and 16cm for MediumBottle (from -9.1cm to 6.9cm). The Groundtruths are determined as -5.7cm and -5.5cm respectively.

4.2 Accuracy predicted height

The optimized liquid fill level for each image of every object is recorded for analysis and comparison with the Groundtruth. The average accuracies of the liquid height predictions using the methods previously explained: SAM, ROI-NN, and the combination of both, are summarized in Table 4.1.

	Combination	**SAM**	**ROI-NN**
BigBottle	97,883%	97,815%	96,993%
MediumBottle	98,598%	97,689%	97,727%

Table 4.1: Average results of the combined method, SAM and ROI-NN

Figures 4.2 to 4.7 show the predicted heights for BigBottle and MediumBottle in the individual images. The Groundtruth is indicated by an orange line as a reference.

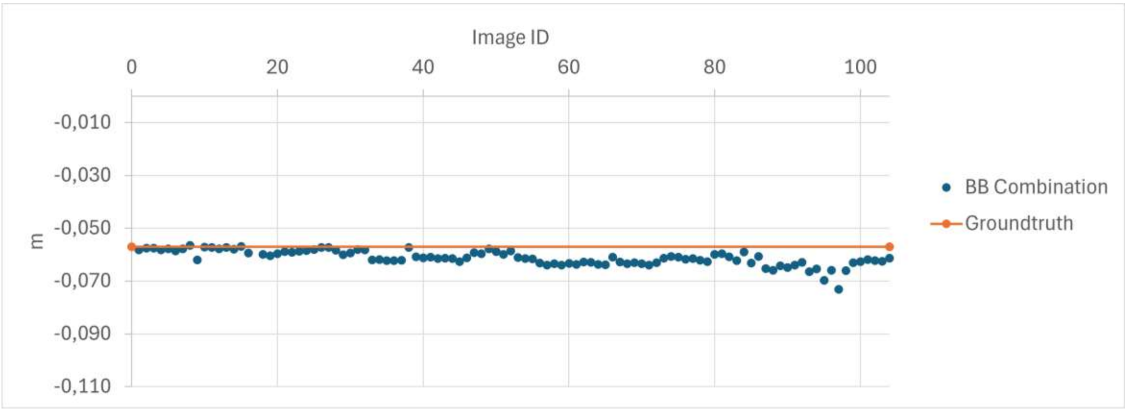


Figure 4.2: Predicted liquid fill level in BigBottle by the combined method

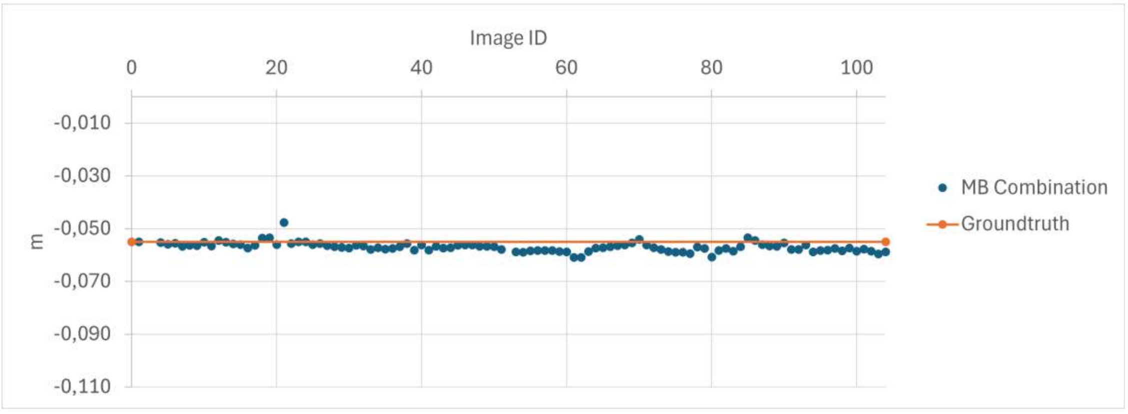


Figure 4.3: Predicted liquid fill level in MediumBottle by the combined method

Using SAM leads to offsets higher and lower than the Groundtruth, while ROI-NN and the combined method generally yield lower results. As previously mentioned, ROI-NN tends to segment more liquid area than SAM, which leads to more noise outside the liquid mask, but more fully segmented parts inside the liquid. SAM segments the area that best separates the area between the "inside the liquid" and "outside the liquid" points. Figure 4.8 shows three possible ways for a successful segmentation for SAM. This leads to alignment at the lower edge of the liquid for some images and at the higher edge for others. The other methods produce the result shown in Figure 4.8a and only align with the higher border, resulting in an overall lower height prediction.

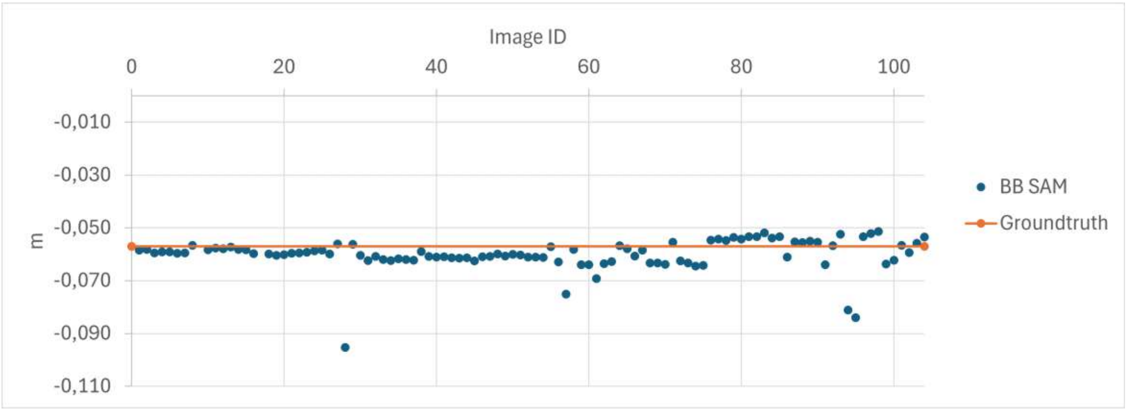


Figure 4.4: Predicted liquid fill level in BigBottle by SAM

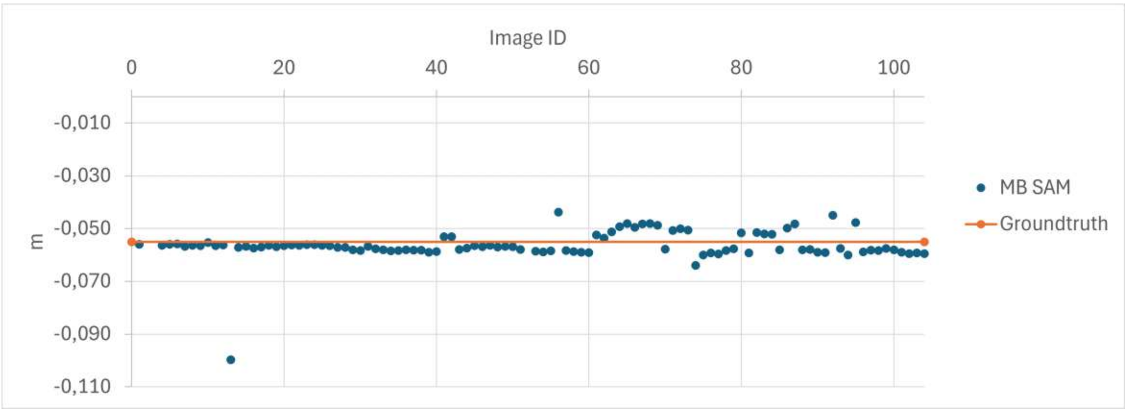


Figure 4.5: Predicted liquid fill level in MediumBottle by SAM

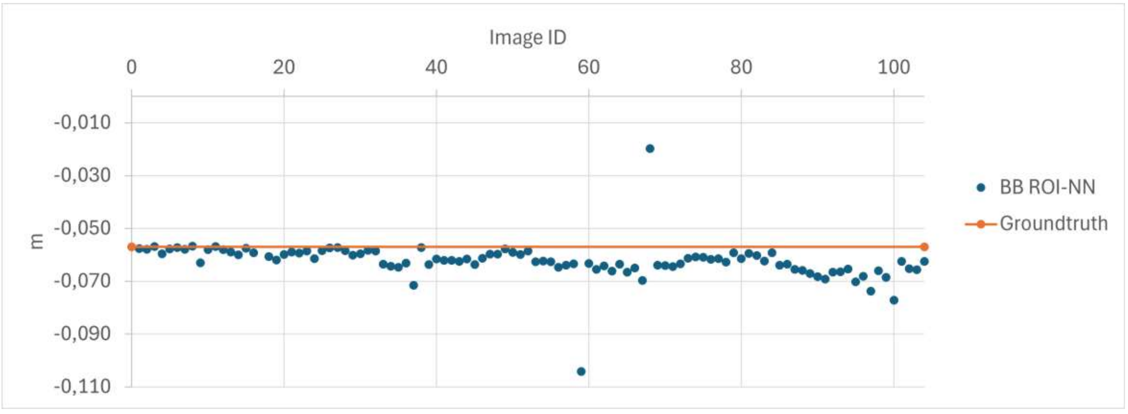


Figure 4.6: Predicted liquid fill level in BigBottle by ROI-NN

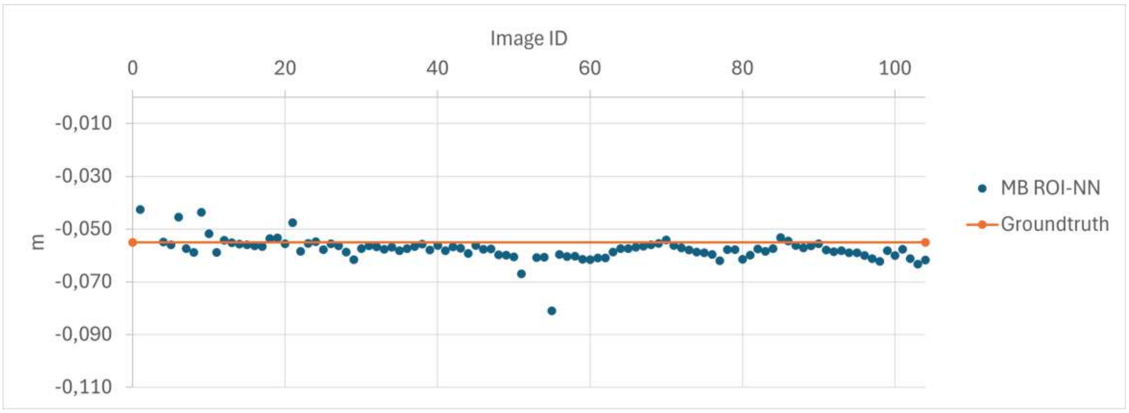
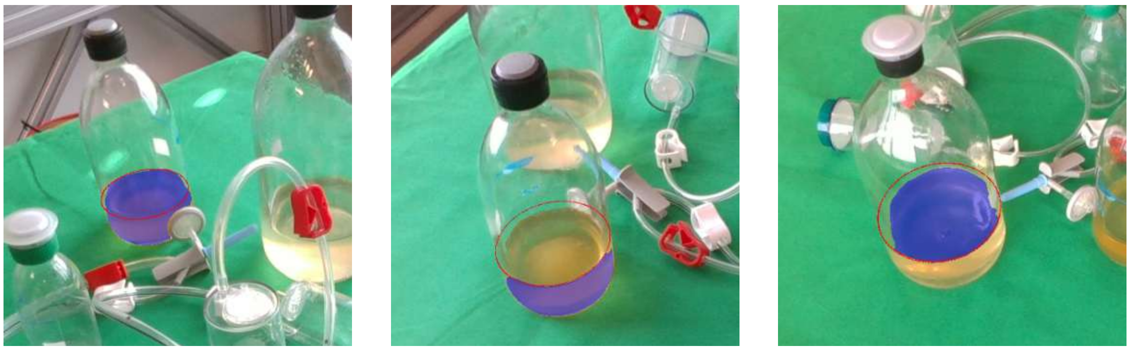


Figure 4.7: Predicted liquid fill level in MediumBottle by ROI-NN



(a) MediumBottle 57 SAM, full liquid detection (b) MediumBottle 66 SAM, outer liquid detection (c) BigBottle 85 SAM, surface liquid detection

Figure 4.8: Different ways SAM detects liquid in objects

Additionally, the pose error increases with higher viewing angles. The images are taken from a total of four different heights: Low viewing angle (image IDs 1-26), middle-low viewing angle (image IDs 27-52), middle-high viewing angle (image IDs 53-78) and high viewing angle (image IDs 79-104). For instance, Figure 4.1a with ID 12 and a low viewing angle shows the result from a side view, while Figure 4.1b with ID 66 and a middle-high viewing angle is taken from a higher position. The Groundtruth in those images also shows a higher offset for increasing viewing angle height. This contributes to a greater error in the predicted liquid heights since they are compared to the Groundtruth which does not perfectly align with the images.

In specific cases, MediumBottle and BigBottle are covered by other objects, making either the bottles or the liquid inside them invisible. Such images are excluded from comparison. These are MediumBottle Image IDs: 2, 3, and 52, and BigBottle Image ID: 18. SAM additionally fails to return a height for BigBottle Image ID: 9, which it is unable to predict, unlike the other methods. This image is shown in Figure 4.9.

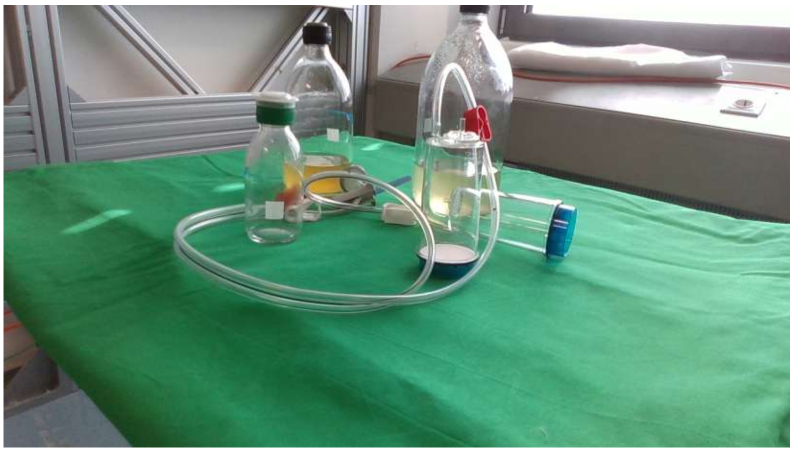
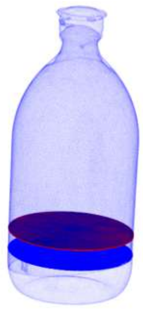


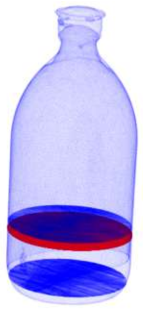
Figure 4.9: RGB image 9

For better visualization of the results, the minimum and maximum values of each method per object are shown in the mesh of the objects, along with the Groundtruth marked in red (see Figures 4.10 and 4.11). It should be noted that the large offsets are often the result of outliers, and the average predictions are typically much closer to the Groundtruth than shown in the meshes.

4.3 Accuracy heights comparison

To visualize the differences between the methods, the average liquid height error within the bottles, as well as the average standard deviation are compared. They are pictured in Figures 4.12 to 4.15. In addition to the previously shown "Combination instance", "ROI-NN instance" and "SAM", the semantic prediction by ROI-NN and the combination of this method with SAM are also compared. In rare cases, the error of the semantic prediction by ROI-NN shows a slightly smaller error than the instance prediction. However, when combined, the instance prediction consistently shows the smallest error. This is likely due to the finer segmentation process used for instance segmentation compared to that of semantic segmentation.


(a) BigBottle combined


(b) BigBottle SAM

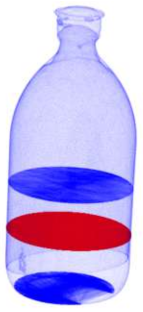
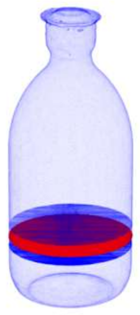
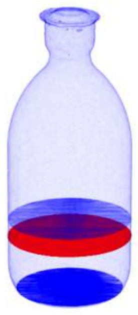

(c) BigBottle ROI-NN

Figure 4.10: Minimum and Maximum prediction results in BigBottle, Groundtruth shown in red


(a) MediumBottle combined


(b) MediumBottle SAM

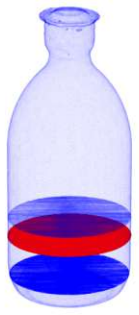

(c) MediumBottle ROI-NN

Figure 4.11: Minimum and Maximum prediction results in MediumBottle, Groundtruth shown in red

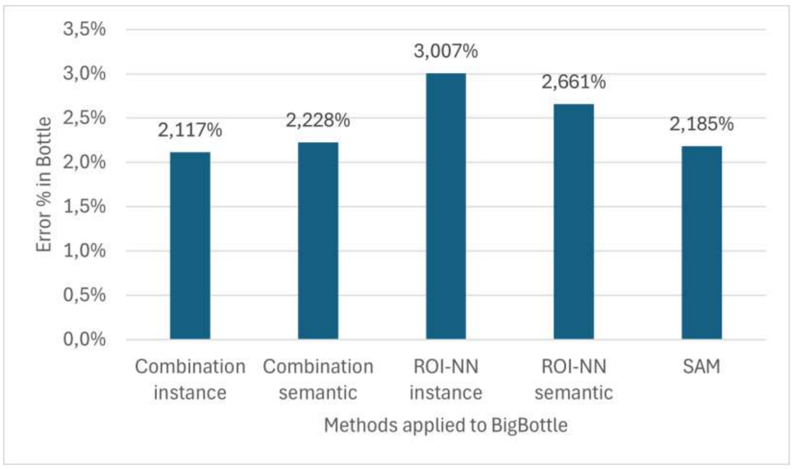


Figure 4.12: BigBottle average liquid height prediction error in object

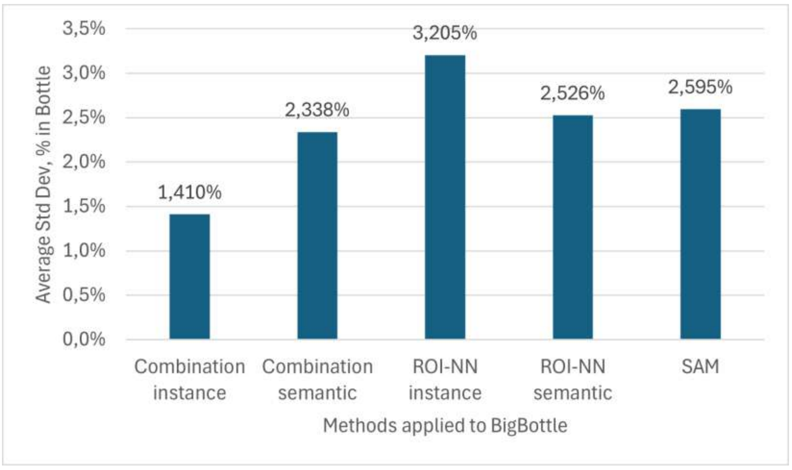


Figure 4.13: BigBottle average standard deviation in object

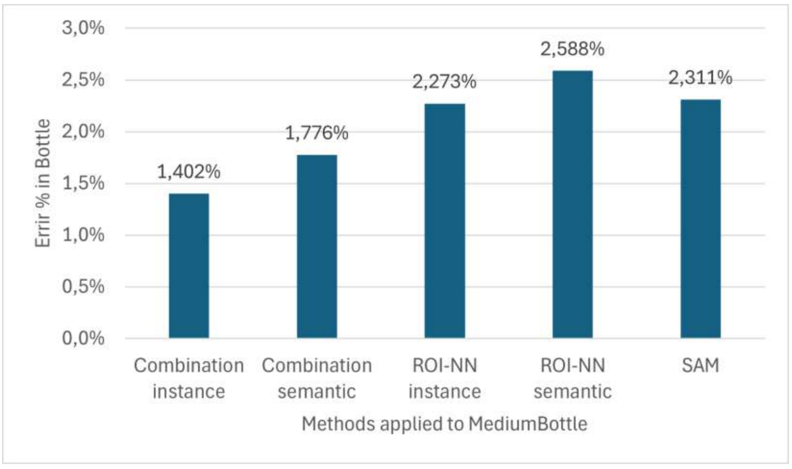


Figure 4.14: MediumBottle average liquid height prediction error in object

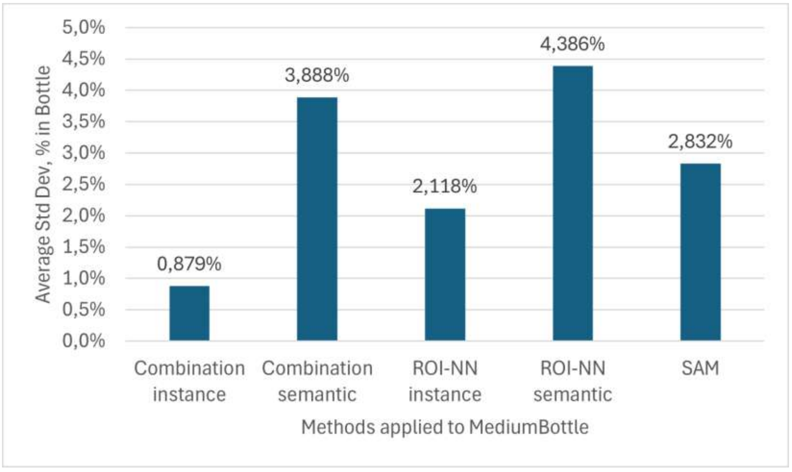


Figure 4.15: MediumBottle average standard deviation in object

4.4 Accuracy masks comparison

Another way to compare the results is by comparing the liquid masks. To create those masks, the object parts below the calculated liquid fill level are projected to 2D and are masked as liquid for all images. The same is applied to the Groundtruth which is used for comparison. Areas inside the bottle within the Groundtruth liquid mask are classified as "Positive", while areas outside the mask but inside the bottle are "Negative". Areas outside the bottle are not taken into account. Consequently, pixels in each image for each object are classified as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). For each image, the total TP + TN + FP + FN is defined as 100 percent. Figure 4.16 and 4.17 show the resulting average percentage of each category.

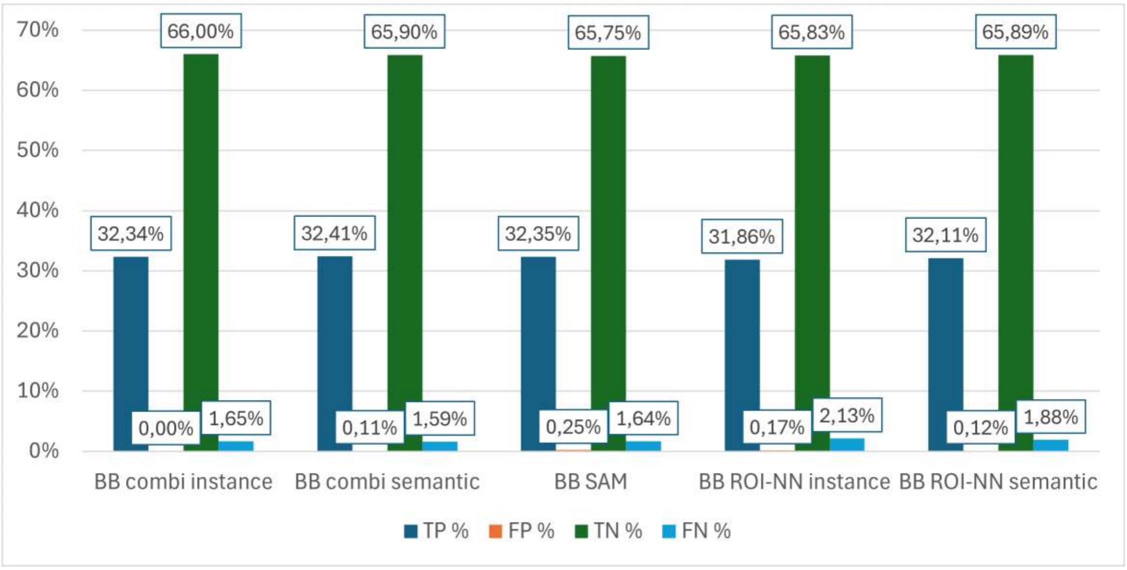


Figure 4.16: Average percentage of True Positive, True Negative, False Positive and False Negative pixels when predicting the liquid fill level in BigBottle

TP + TN are combined as "True" and FP + FN as "False" to simplify the results for better visualization and comparison, as seen in Figures 4.18 and 4.19. The results discussed in Section 4.3 closely resemble the ones visible here.

Using liquid masks also allows for the comparison between the optimized results and the original predicted masks. A slight improvement is shown in Figures 4.20 and 4.21. This is likely due to noise in the masks inside and outside the liquid, which was filtered in post-processing (Section 3.1.3), but is still present in the original predicted liquid masks used in this comparison.

Another insightful comparison involves other NN results. The additional networks presented are Generator Evaluator Selector Net (GES-Net) [15] and TransProteus [91]. Both of these NNs work without object masks.

GES-Net uses hierarchical segmentation by predicting first the vessels and then the content materials in separate steps using FCNs. While a single-step segmentation is introduced in the same paper [15], the results do not show significant differences since both methods are trained on the same data. The paper introducing both implementations further demonstrates that GES-Net produces more accurate results, which is why it is

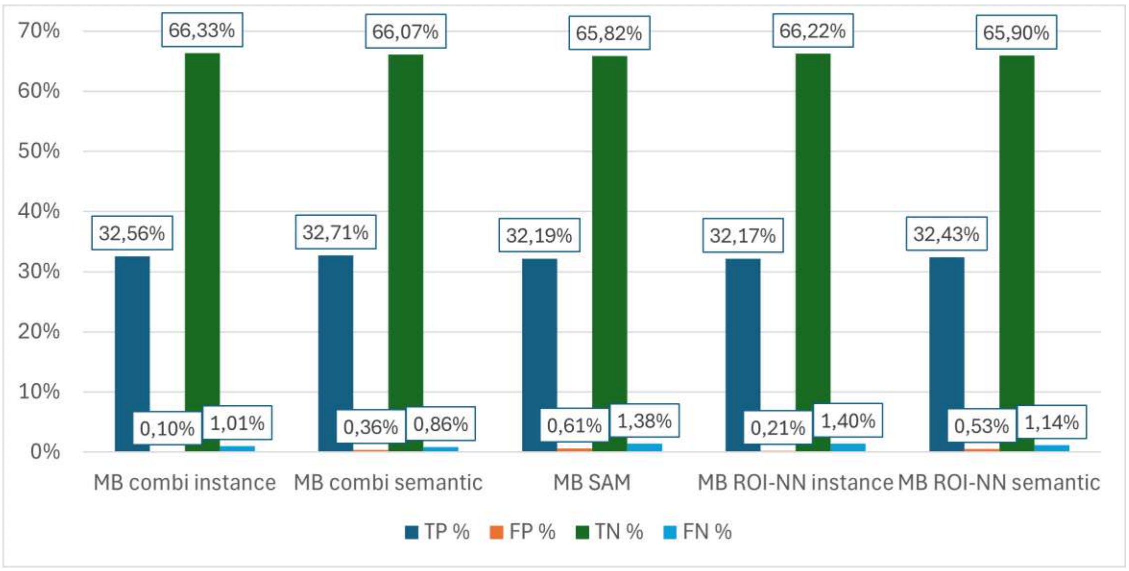


Figure 4.17: Average percentage of True Positive, True Negative, False Positive and False Negative pixels when predicting the liquid fill level in MediumBottle

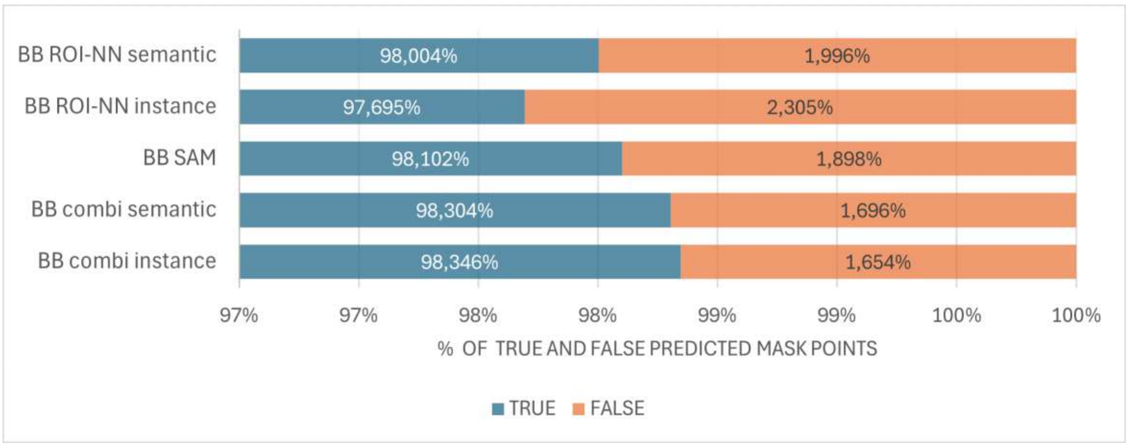


Figure 4.18: Average percentage of True and False pixels when predicting the liquid fill level in BigBottle by different methods

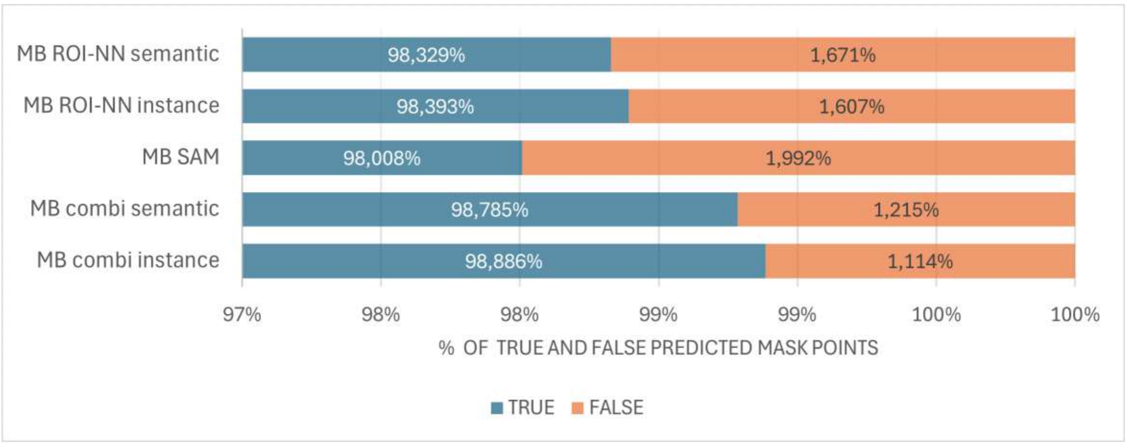


Figure 4.19: Average percentage of True and False pixels when predicting the liquid fill level in MediumBottle by different methods

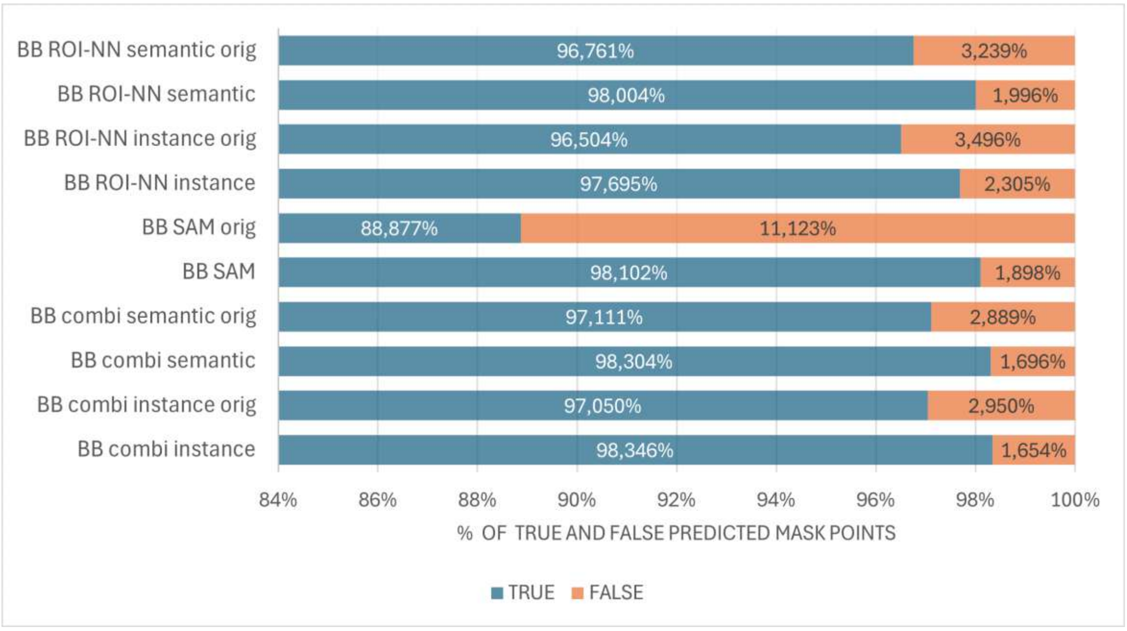


Figure 4.20: Average percentage of True and False pixels comparing original prediction mask and optimized fill level masks of BigBottle

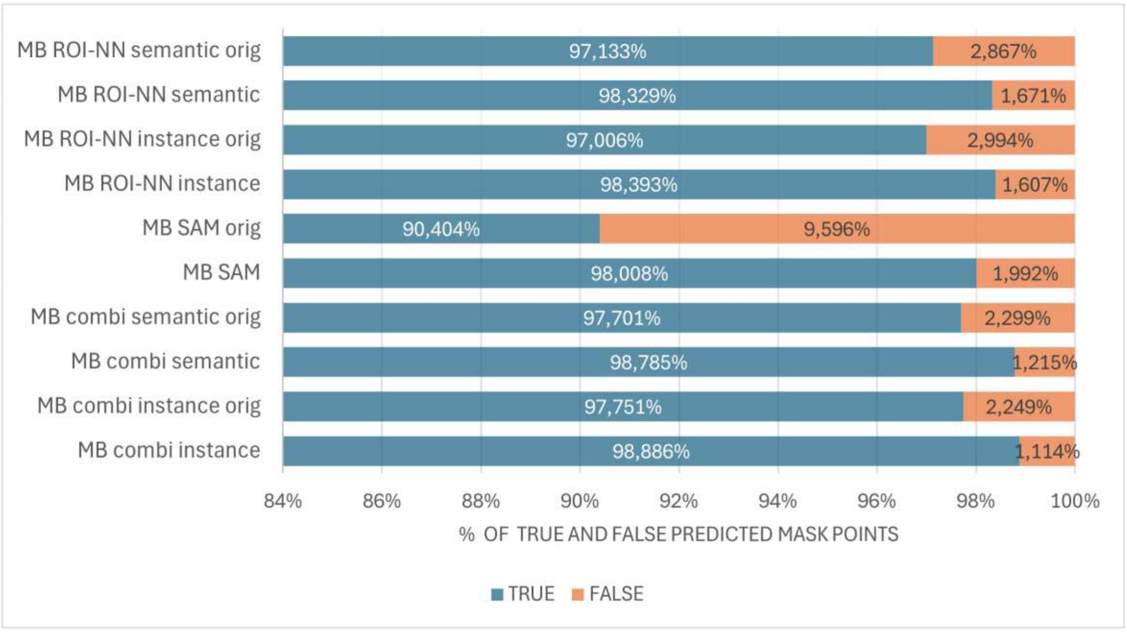


Figure 4.21: Average percentage of True and False pixels, comparing original prediction mask and optimized fill level masks in MediumBottle

used for this comparison.

GES-Net produces both semantic and instance-segmented results, whereas TransProteus uses a semantic approach and additionally predicts the 3D structure of the vessels and the contents [91]. However, the 3D output is not detailed enough for fill-level comparison, so only the 2D predicted masks are compared.

As Figures 4.22 and 4.23 show, the error rate is higher for the simpler NNs that do not consider the object mask compared to the combined method presented in this thesis.

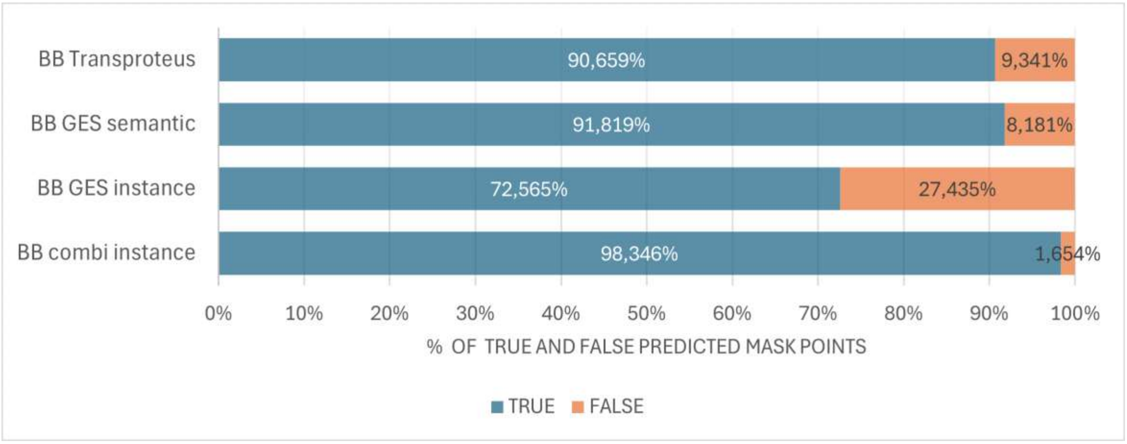


Figure 4.22: Average percentage of True and False pixels, comparing NNs and the combined method of this thesis in BigBottle

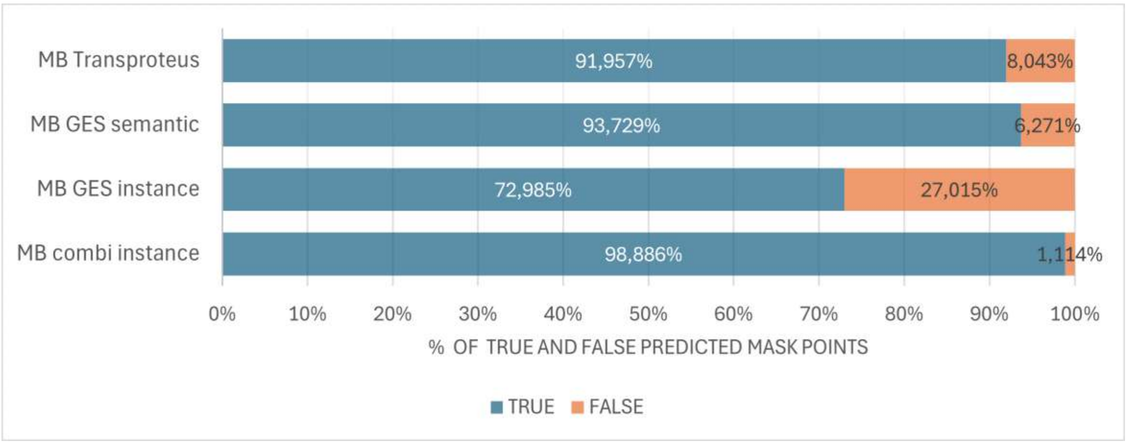


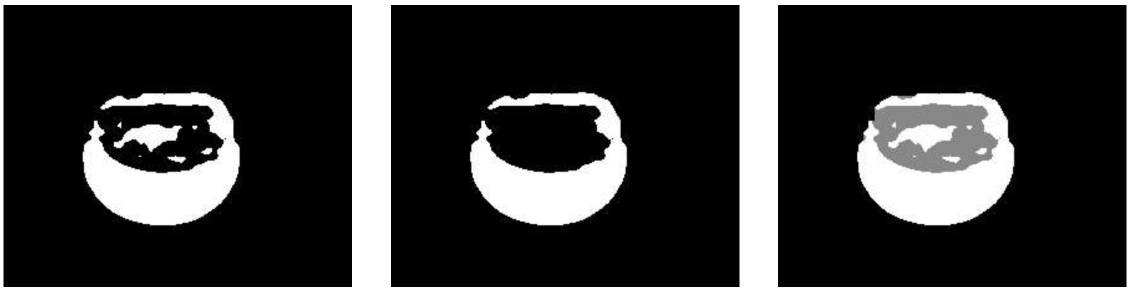
Figure 4.23: Average percentage of True and False pixels, comparing NNs and the combined method of this thesis in MediumBottle

4.5 SAM modified post-processing results

Liquid masks segmented by SAM tend to have more noise in the liquid itself, such as disturbed surfaces. In contrast, segmentation by ROI-NN has more noise outside the liquid in the form of extra annotated areas. These differences suggest that post-processing as discussed in Section 3.1.3 is not optimal for SAM. Better results are achieved when changing these two steps:

4.5.1 Skipping mask noise removal

The first original post-processing step involves removing noise from the liquid mask before applying morphological closing in Section 3.1.3. However, removing small areas from the liquid mask first results in the loss of important information, as illustrated in Figure 4.24. By skipping this step and proceeding directly to the morphological closing of the mask, the liquid mask is retained better.



(a) Original prediction of MediumBottle 74 by SAM without post-processing (b) MediumBottle SAM mask 74, morphological closing after removing noise (c) MediumBottle SAM mask 74, morphological closing without removing noise

Figure 4.24: Modified post-processing for SAM in MediumBottle 74

4.5.2 Different filtering of the liquid fill level related edges

Instead of using the lowest remaining edge for optimization, as described in Section 3.1.3, a few long edges are retained for this purpose. To mitigate noise, any edge shorter than 0.5 times the length of the longest edge is removed to ensure more stable optimization. However, no additional filtering steps are necessary. The reason for not relying solely on one longest or lowest edge is that the remaining edges are likely all related to the liquid fill level but may not be connected. Relying on just one edge could result in losing valuable information.

Figure 4.25 illustrates edge predictions, showing two components after filtering out noise. The original post-processing approach detailed in Section 3.1.3 would only select the lower edge for optimization. By keeping the upper edge as well a bigger area of interest is covered.

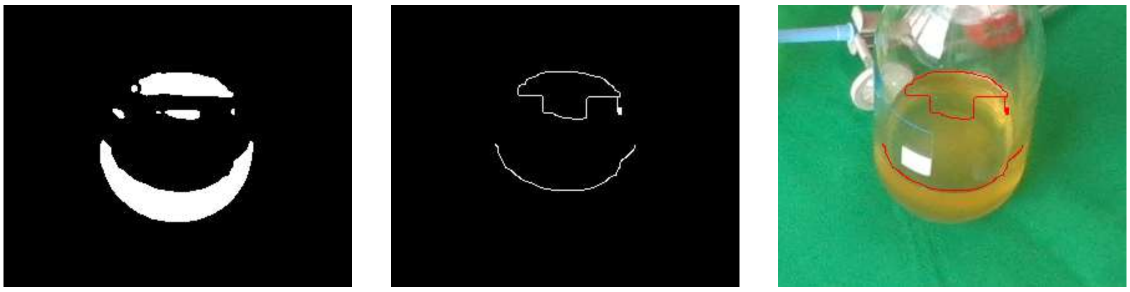

(a) Original prediction of MediumBottle 87 by SAM without post-processing (b) MediumBottle SAM mask 87, filtered edges using modified post-processing (c) MediumBottle SAM mask 87, modified post-processing edges overlay

Figure 4.25: Modified post-processing for SAM in MediumBottle 87

4.5.3 Improved results for SAM

Figure 4.26 and 4.27 compare SAM with original and modified post-processing. The results indicate a slight improvement in average correct detections.

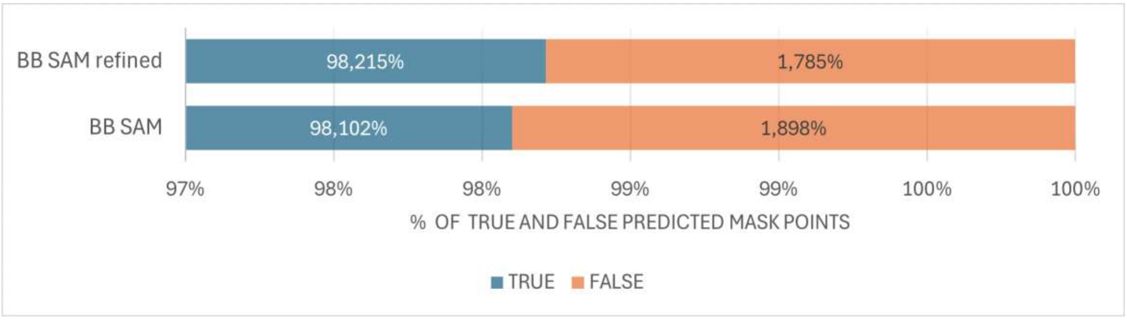


Figure 4.26: BigBottle comparison original and refined SAM

Finally, SAM with refined post-processing is compared to other methods introduced in this thesis in Figures 4.28 and 4.29.

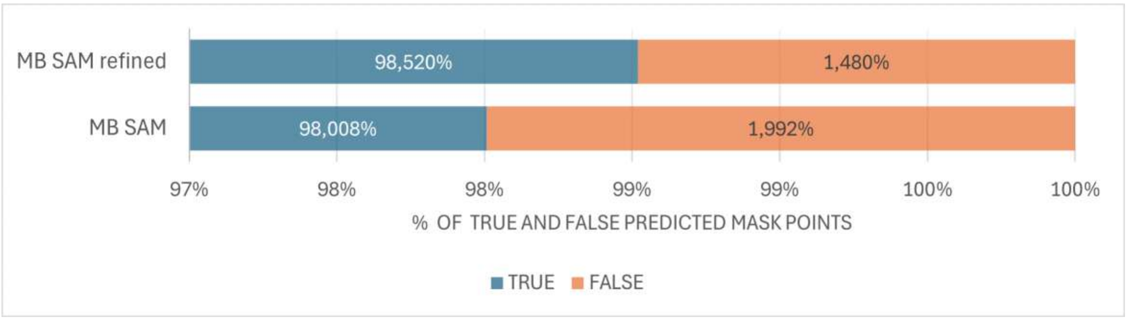


Figure 4.27: MediumBottle comparison original and refined SAM

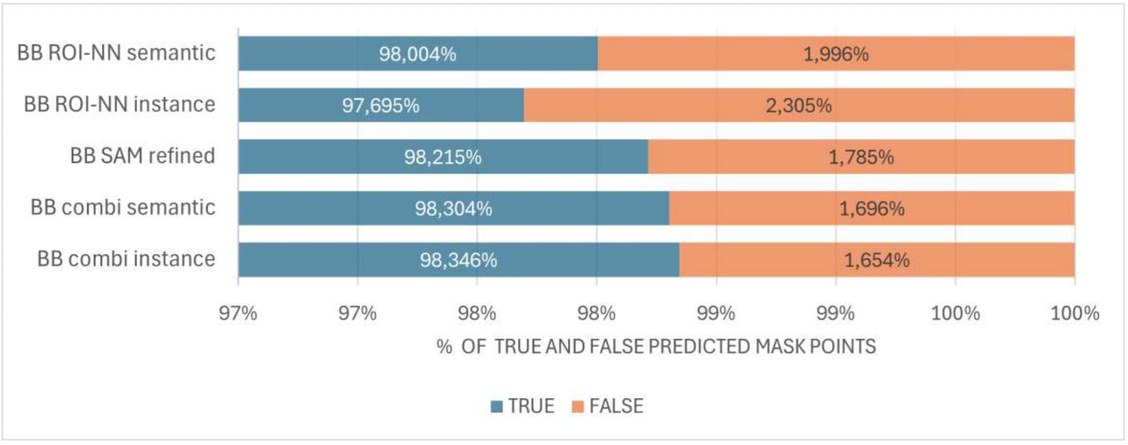


Figure 4.28: BigBottle comparison methods with refined SAM

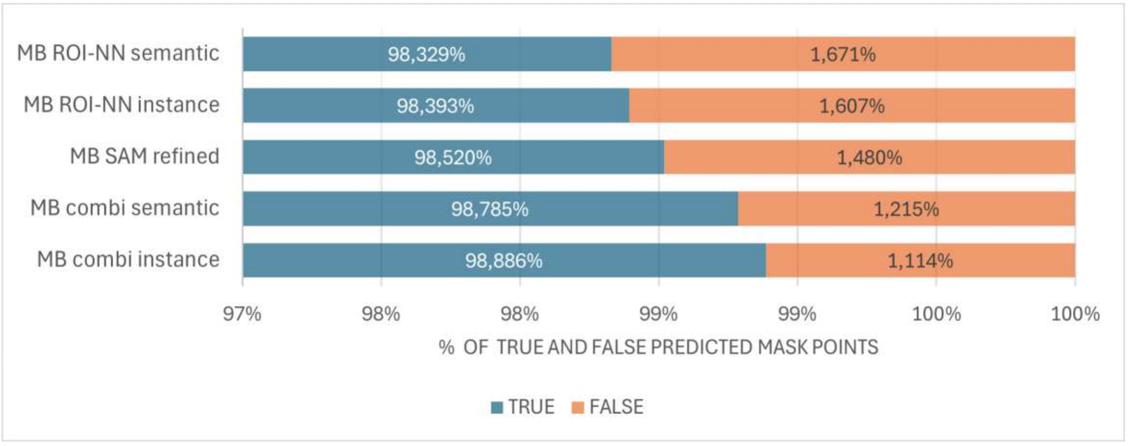


Figure 4.29: MediumBottle comparison methods with refined SAM

4.6 Results of different scenes

The previous liquid fill level results, using the combined method, are further compared to the results of different scenes. These scenes present challenges, such as a more cluttered environment in scene j_008 (see Figure 4.30), or the presence of a more transparent liquid in scene p_020. Furthermore, the images in this scene are intentionally captured upside down to increase the level of difficulty (see Figure 4.31).

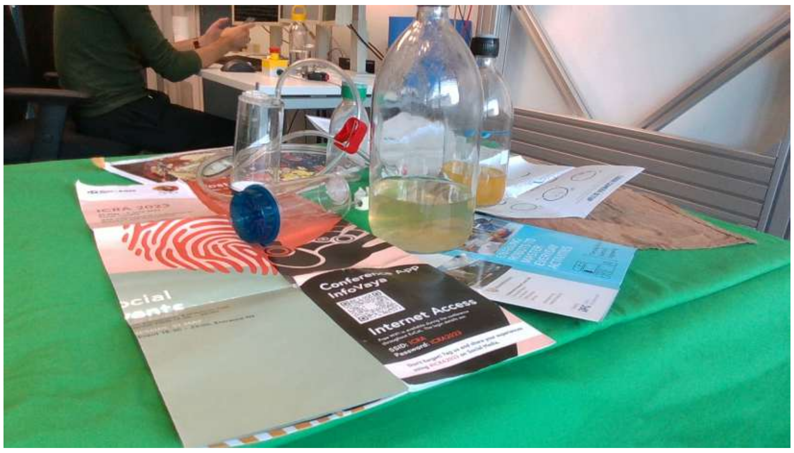


Figure 4.30: Scene j_008, image ID 1

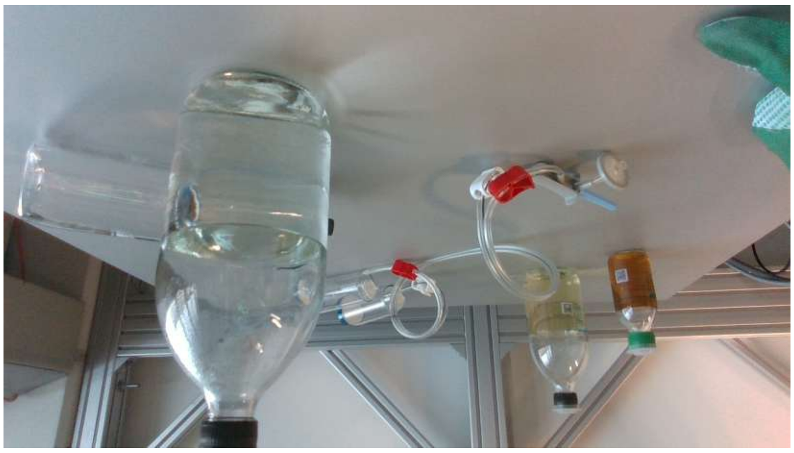


Figure 4.31: Scene p_020, image ID 10, images captured upside down

Figures 4.32 to 4.35 show the average error and standard deviation for each scene for BigBottle and MediumBottle. It is evident that liquid fill level prediction works well for the original scene j_005 and scene j_008, but shows higher error for scene p_020.

Scene j_008 uses the same objects and liquid fill level as the original scene j_005. The similarities between these scenes lead to expected accurate optimization results. The overall average values of scenes j_005 and j_008 are similar. In Figure 4.34, MediumBottle in the more clustered scene leads to even better liquid-level predictions. This improvement is due to more accurate poses in each image, evident in Figures 4.36 and 4.37, which show less offset than scene j_005 in Figures 4.2 and 4.3. However, the standard deviation of

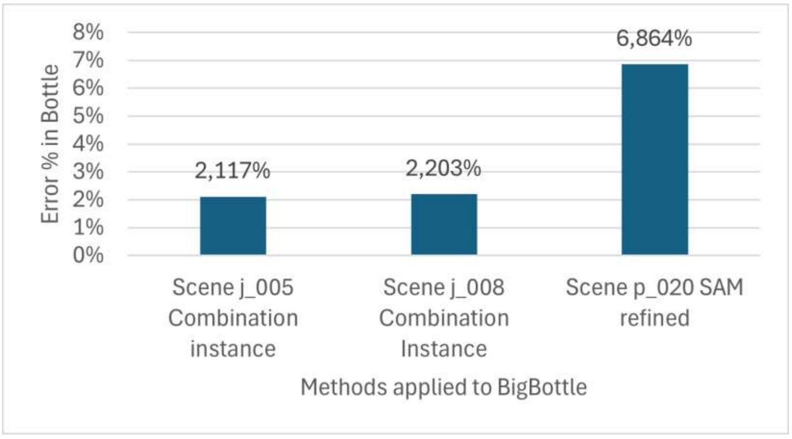


Figure 4.32: BigBottle, comparison of the average error of liquid fill level results in different scenes

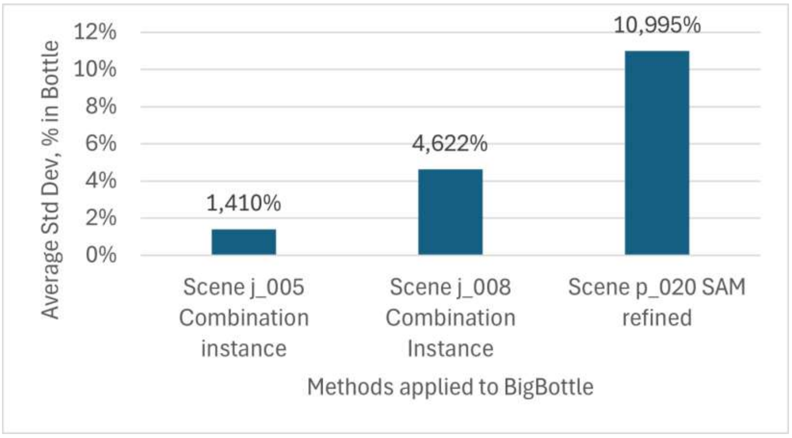


Figure 4.33: BigBottle, comparison of the average standard deviation of liquid fill level results in different scenes

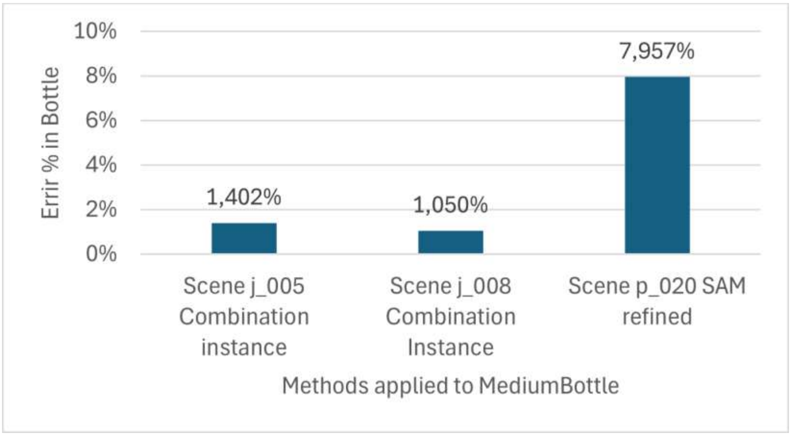


Figure 4.34: MediumBottle, comparison of the average error of liquid fill level results in different scenes

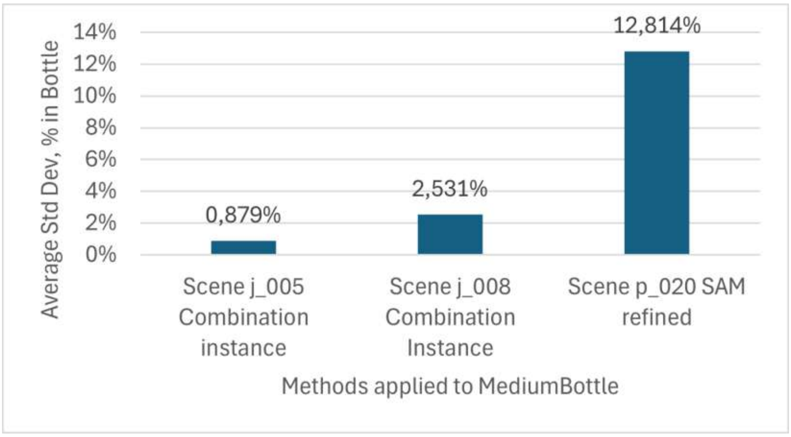


Figure 4.35: MediumBottle, comparison of the average standard deviation of liquid fill level results in different scenes

scene j_008 is higher due to more outliers in the clustered environment. Most outliers occur because cables or other objects intersect with the containers, leaving less visible liquid and making the liquid mask prediction in the cluttered environment error-prone.

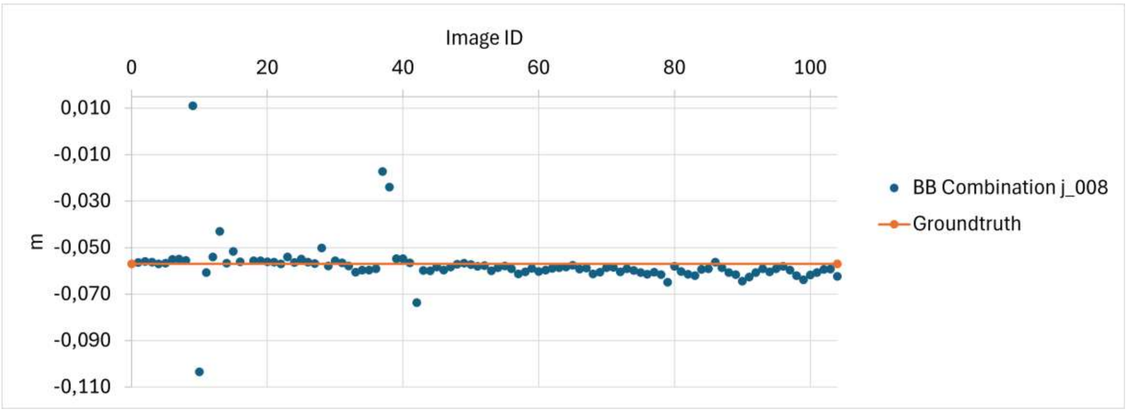


Figure 4.36: Predicted liquid fill level in BigBottle by the combined method, scene j_008

In scene p_020, the gravity vector must be inverted since the initial coordinate system in the images points upwards, while the gravity vector must point in the opposite direction. The objects in this scene are not detectable by ROI-NN, and consequently not by the combined method, as it fails to segment the liquid masks correctly. It appears to have been trained primarily on liquid-filled containers without camera rotation.

Due to placing the points for SAM based on the gravity vector, SAM does not struggle with camera rotation, but faces a different challenge: transparent liquid. The clear liquid makes other edges appear sharper than liquid fill level-related ones, and the NN segments wrong areas. By using refined SAM and placing the points closer to the center, slightly better results are achieved, but compared to the other scenes it does not perform well. Figures 4.38 and 4.39 show many large outliers that are not detected correctly, where the liquid fill level detection fails.

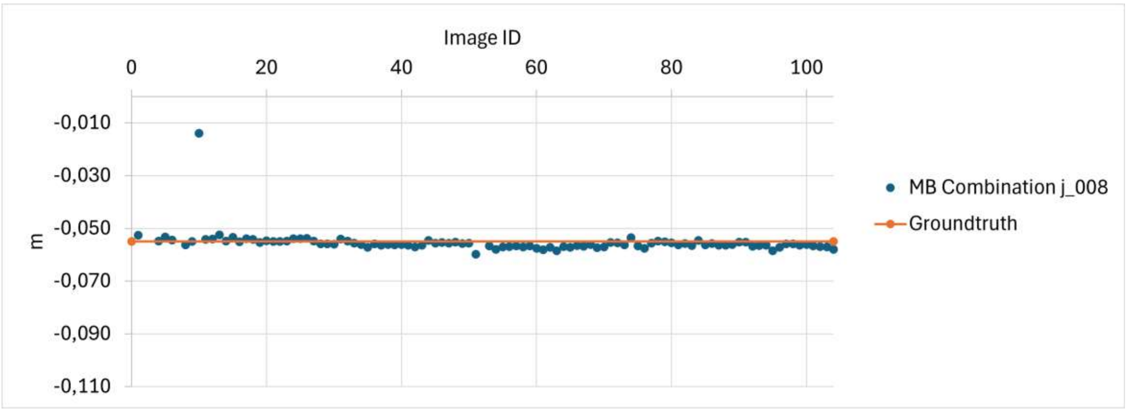


Figure 4.37: Predicted liquid fill level in MediumBottle by the combined method, scene j_008

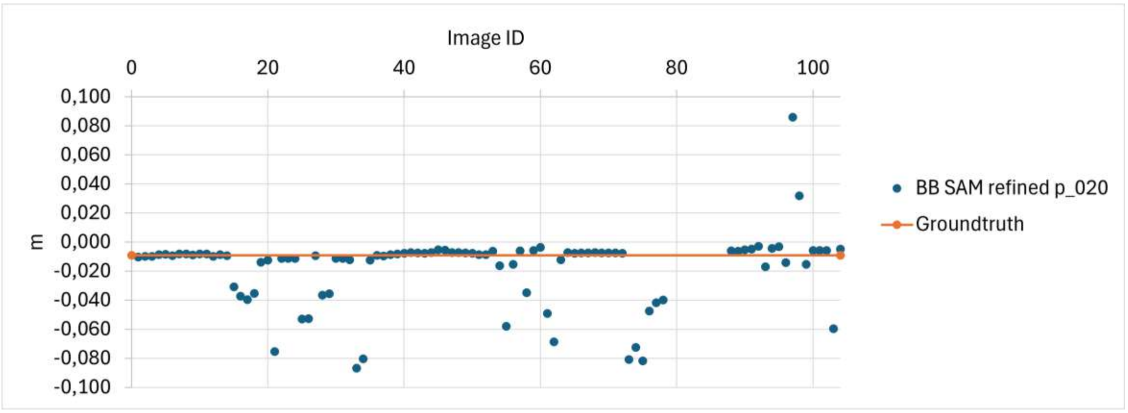


Figure 4.38: Predicted liquid fill level in BigBottle by refined SAM, scene p_020

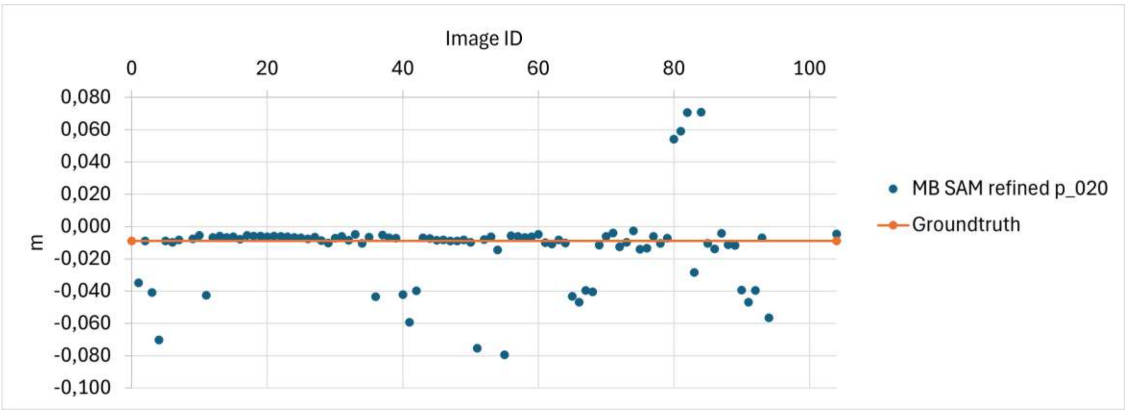


Figure 4.39: Predicted liquid fill level in MediumBottle by refined SAM, scene p_020

5 Difficulties and discussion

Table 5.1 shows the advantages and disadvantages of the liquid mask segmentation methods applied in this thesis.

Method	Advantages	Disadvantages
Combination	More stable than each of the methods alone; Less prone to outliers	If both methods fail or one mask is faulty, the overall result is not good
SAM	Exact when the point placement is good	Relies on point placement which needs to be adjusted based on the object's shape; The to detecting liquid height has to be between the points in the bottle, SAM will always predict a mask between the points; Slower than ROI-NN; Noisy inside the liquid - too little masked
ROI-NN	No additional information is necessary for predicting, only the object mask	Less exact liquid mask prediction than SAM at the borders; Disturbed by unmarked covering objects; Noisy outside the liquid - too much masked

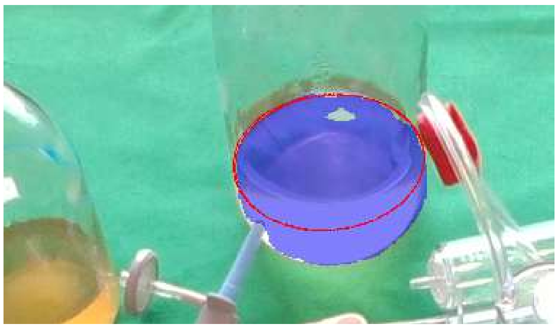
Table 5.1: Comparison of the combined method, SAM liquid mask prediction, and ROI-NN liquid mask prediction

For better visualization, Figures 5.1 to 5.3 show the limitations of the different methods by presenting incorrect predictions, which also appear as outliers in previous results. The liquid masks are overlaid in blue on the original RGB images, with the optimized mesh-plane intersection projected in red.

Figure 5.4 contrasts the segmentation results from SAM and ROI-NN, highlighting the clearer segmentation at the borders for SAM.

5.1 Pose and Gravity vector accuracy

As previously mentioned, optimization and liquid plane detection require exact knowledge of object poses. Even though the poses are generally accurate, slight offsets influence the results. The object masks are generated based on these poses and the corresponding object meshes. If they are not precise, finding the liquid plane in the mesh first and then


(a) BigBottle 97, both NNs did not segment the upper border

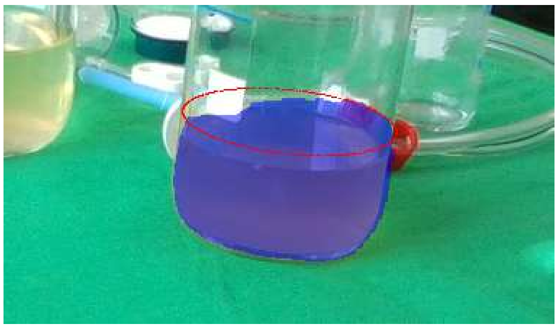
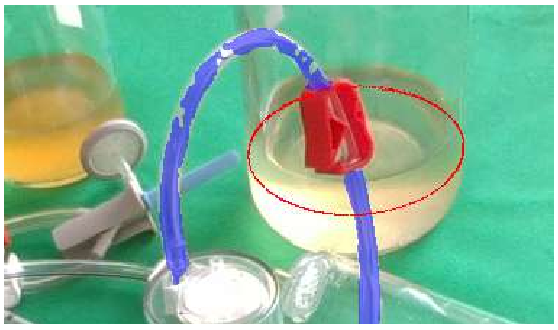

(b) MediumBottle 21, ROI-NN segments too high which is also visible in the overlay

Figure 5.1: Combination method, outlier


(a) BigBottle 57, cable segmented due to point placement

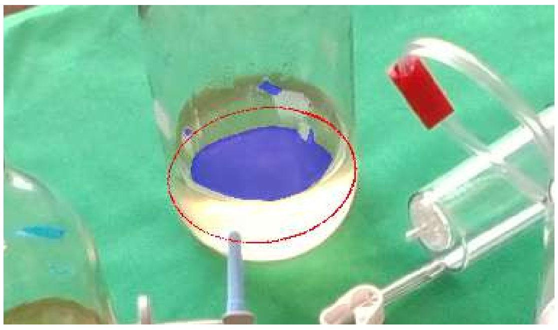
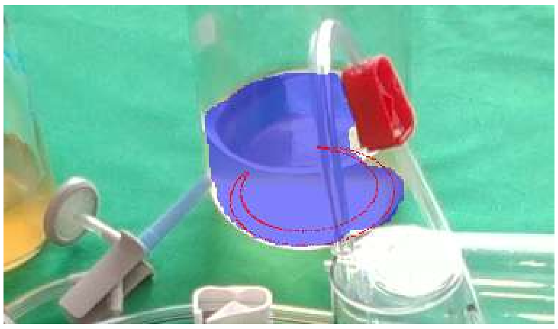

(b) BigBottle 95, only a small liquid area is segmented

Figure 5.2: SAM prediction, outlier


(a) BigBottle 59, unclear edges at the bottom, too far from the border to be removed

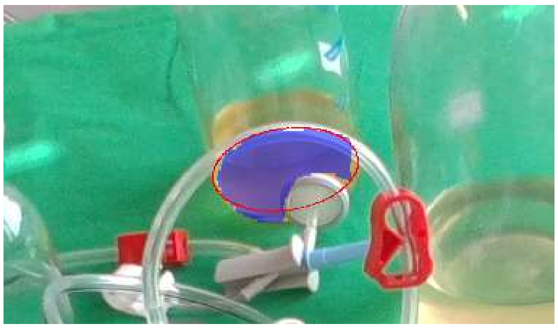

(b) MediumBottle 55, segmentation disturbed by a cable

Figure 5.3: ROI-NN prediction, outlier

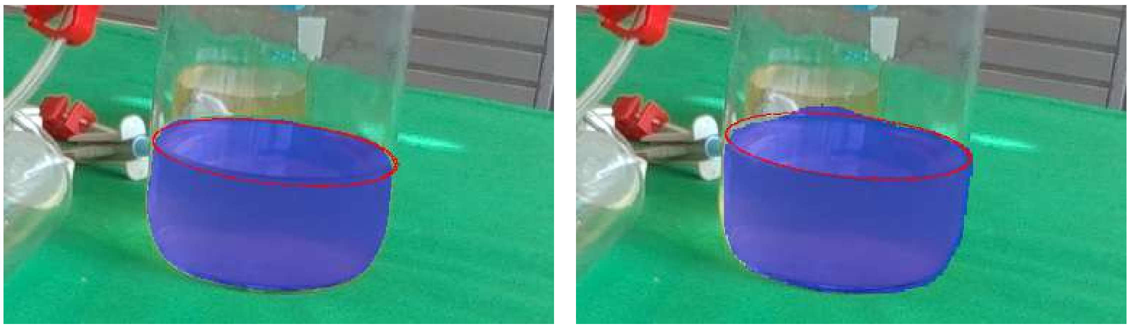

(a) BigBottle 3 SAM, clear segmentation (b) BigBottle 3 ROI-NN, unclear segmentation

Figure 5.4: Segmentation clearness differences SAM and ROI-NN

projecting it to 2D is also affected. The point selection for SAM and the segmentation performed by ROI-NN also depend on accurate poses.

Additionally, the gravity vector is assumed to be aligned with the initial coordinate system, which cannot be guaranteed with complete certainty. Since this alignment affects the optimization direction after the 2D projection, various factors introduce uncertainty.

Figure 5.5 shows offsets in overlaid object masks. They display missing coverage in some areas and too much in others, like in Figure 5.5b on the bottle in front of MediumBottle. Although the offsets are relatively small, all the methods applied in this thesis are sensitive errors in poses.

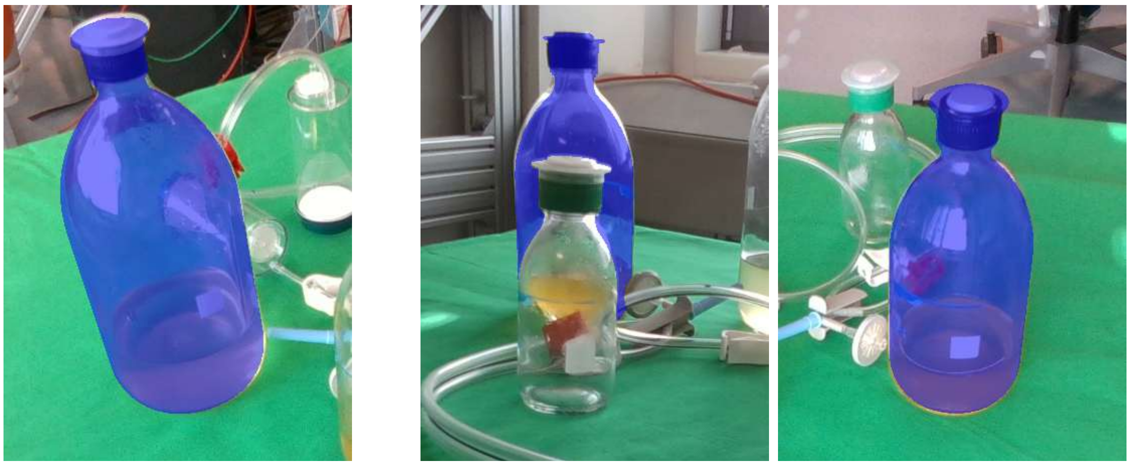

(a) BigBottle 70, mask overlay offset (b) MediumBottle 10, mask overlay offset (c) MediumBottle 73, mask overlay offset

Figure 5.5: Not perfectly aligned objects masks overlaid with RGB images

5.2 Unclear meshes and projection

The correctness of the object meshes also impacts the accuracy of the results. The density of these meshes varies along the height of the objects. Figure 5.7 shows two steps in the optimization loop with the outer circles being the mesh-plane intersections projected to

2D and the inner one the predicted liquid edges. Although the height difference is minimal, changing density in the projected intersection is visible. Even within a single step, the density varies. Figure 5.7b reveals double edges, which are also observed in intersections projected to the RGB images, such as in Figure 5.6. These double edges appear only in BigBottle at a specific height range and disappear when passing through this range, as illustrated in Figure 5.7a. Additionally, when projecting the intersection to 2D, the smaller ends of the projection naturally contain denser points compared to the rest of the projection.

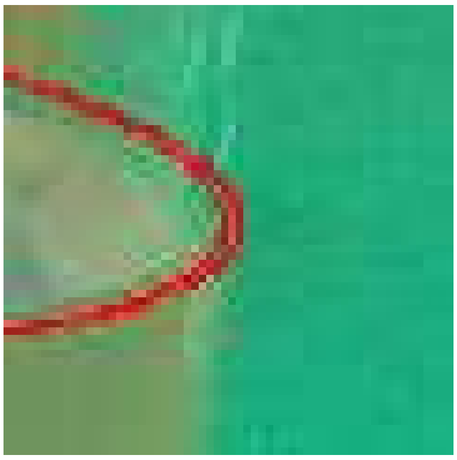


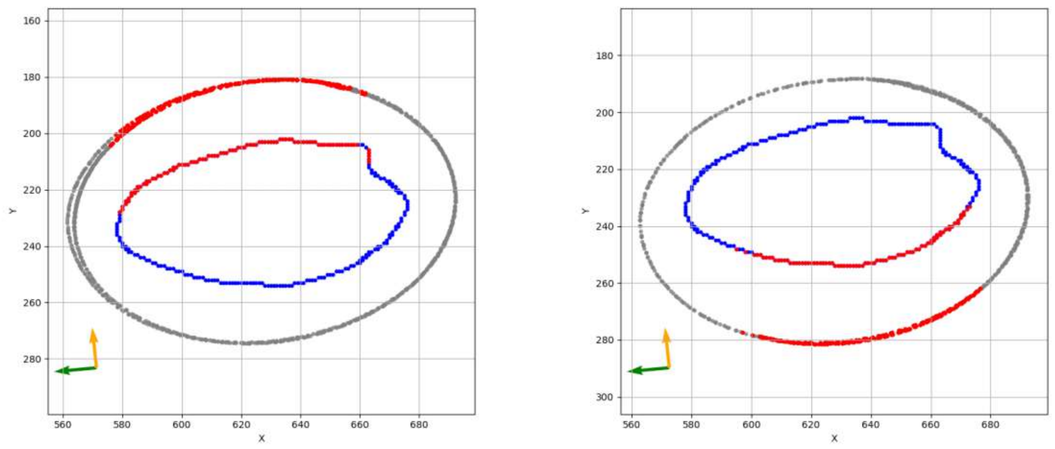
Figure 5.6: BigBottle 2 Groundtruth, double edges

The uneven distribution of meshes and thus the projection, contribute to unstable optimization. As explained in Section 3.2.2, one of the loss functions used for optimization aligns the points of the mesh-plane intersection with the predicted liquid edges generated by the NNs. Due to the changing point density in the mesh-plane intersection, each optimization step yields a different number of matches within specific distance categories. Figure 5.7 provides an example where the matches on the top and bottom are similar due to the shape of the liquid edges. As the optimization progresses, the number of matches varies, and the optimization jumps between top and bottom alignment. Figure 5.2b shows an example result in such cases.

Even without such extreme cases, the inconsistency in matches at each step makes the optimization process less smooth. Consequentially, the process requires more optimization steps and longer run times until the optimized height is obtained.

5.3 Unaccounted distortions in transparent containers

Another factor why the image and the projected optimized liquid fill level do not match is that reflections and refractions from the transparent object walls, as well as the liquid inside, distort the image. The liquid is not necessarily shown at calculated places, as the material and thickness of transparent objects, as well as the viewing angle impact its appearance. This thesis does not address these complex phenomena, which is one of the reasons why the Groundtruth does not match the visible liquid level in all images. Figure 5.8 shows an example where a combination of slightly inaccurate poses, object



(a) BigBottle SAM 95, alignment of the mesh-plane intersection and the liquid edges at the top

(b) BigBottle SAM 95, alignment of the mesh-plane intersection and the liquid edges at the bottom

Figure 5.7: BigBottle SAM 95, alignment of the mesh-plane intersection (grey points) and the liquid edges (blue points). Red points indicate aligned points, blue and grey points are not used for optimization in that step. The yellow arrow points in optimization direction.

meshes, gravity vector, and unaccounted distortions from the transparent object leads to misaligned results.

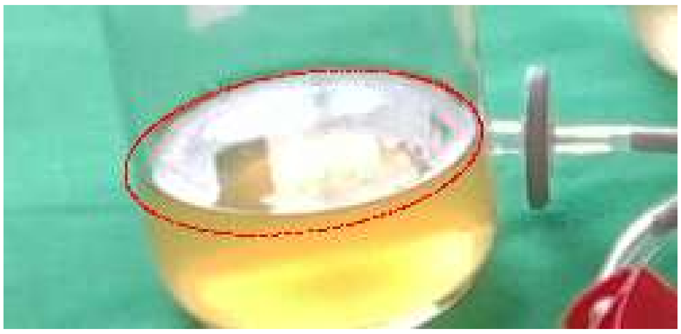


Figure 5.8: MediumBottle 41 combination, liquid edge projection is outside of the object and not perfectly aligned

5.4 Execution times

To compare the speeds of SAM segmentation, ROI-NN segmentation, and the optimization loop, the duration of each method was recorded for BigBottle and MediumBottle. Tables 5.2 and 5.3 present the times for one optimization loop execution, the number of optimization steps per loop, and the calculated time of a single step. From a subset of 20 measurements per bottle using different images, the average, minimum, and maximum values are documented.

BigBottle	Optimization Loop Time	Steps Number	Steps Time
Average	53.04s	140.43	0.38s
Maximum	143.12s	300	0.50s
Minimum	13.35s	79	0.14s

Table 5.2: BigBottle Image ID 21 - 41, measured optimization loop times and steps

MediumBottle	Optimization Loop Time	Steps Number	Steps Time
Average	20.76s	125.10	0.17s
Maximum	49.55s	300	0.19s
Minimum	16.28s	111	0.14s

Table 5.3: MediumBottle Image ID 56 - 76, measured optimization loop times and steps

The time required for a single optimization step for BigBottle is more than twice that of MediumBottle. This is due to BigBottle having a larger size and denser mesh, which naturally leads to longer computation times when calculating the mesh-plane intersection. The projected intersection consequently has more points for BigBottle, which increases the execution time of the loss functions.

Figures 5.9 and 5.10 show the optimization loop time compared to the duration of SAM and ROI-NN liquid mask predictions, excluding post-processing. Both NNs show similar average times for BigBottle and MediumBottle, which was expected since the RGB images all have the same size. However, the significant difference in optimization loop durations across the objects is evident.

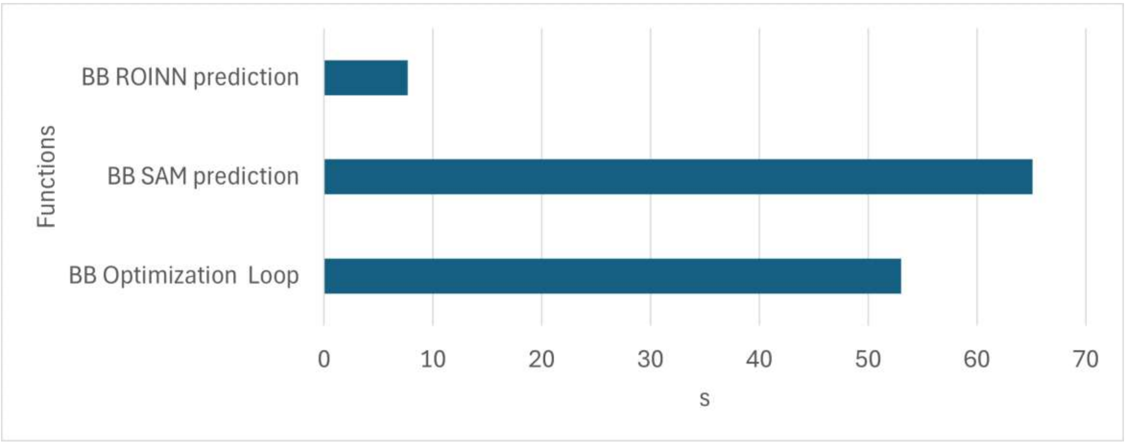


Figure 5.9: BigBottle, mesauring times for SAM prediction, ROI-NN prediction and the optimization loop. Average of times for Image ID 21 - 41

All times were recorded on a laptop with a CPU (AMD Ryzen 5 4500U; 2.3 – 4.0 GHz) and 16GB RAM. Both NNs offer the option to run the code on Nvidia GPUs, and the libraries used for the optimization loop in this thesis are also commonly used on GPUs.

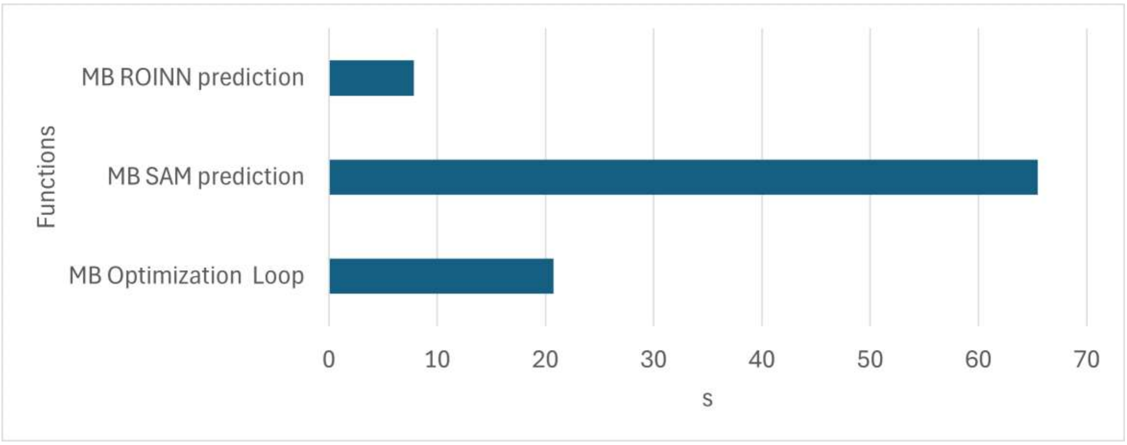


Figure 5.10: MediumBottle, mesauring times for SAM prediction, ROI-NN prediction and the optimization loop. Average of times for Image ID 56 - 76

Using a more powerful computer equipped with an Nvidia GPU would significantly reduce the runtimes for the NN segmentations and the optimization loop.

6 Conclusion

This thesis proposed an effective method for detecting liquid fill levels in known transparent containers, returning results in 2D and 3D. It utilized combinations of state-of-the-art liquid segmentation techniques - specifically SAM and ROI-NN -, refined the detection results, and converted them to 3D using gradient descent optimization with novel loss functions.

Unlike most previous approaches, this method worked in cluttered environments without restrictions regarding the setup, lighting, or background, with improved accuracy. On the datasets it was tested on, it achieved more accurate results than previous state-of-the-art methods for detecting liquid fill levels. The technique successfully detected liquids in two types of transparent containers across more than 200 images, achieving an accuracy exceeding 98%. Such cluttered scenarios are common in real-life applications, and the high detection accuracy aids in precisely handling transparent containers. It also yielded three-dimensional results, which are essential for managing liquid-filled containers.

However, the method encountered challenges when detecting clear liquids, and ROI-NN struggled with large camera rotations. Additionally, due to the point selection required for SAM, it only detected liquid levels between those programmatically specified points.

To address these issues, the first part of the pipeline - liquid segmentation - must be refined to work independently of camera rotation and point selection while providing better results for clear liquids. One potential solution is to train a new network for liquid segmentation using a broader dataset that includes all the challenging cases mentioned.

Bibliography

[1] Grammarly Inc., *Grammarly writing assistant*, `https://www.grammarly.com`, Accessed: 2025, 2025.

[2] OpenAI, *Chatgpt*, Accessed: 2025, 2025. [Online]. Available: `https://chat.openai.com`.

[3] F. Zhu, S. Hu, L. Leng, A. Bartsch, A. George, and A. B. Farimani, „Pour me a drink: Robotic precision pouring carbonated beverages into transparent containers,“ *arXiv preprint arXiv:2309.08892*, 2023.

[4] N. N. S. A. Rahman, N. M. Saad, A. R. Abdullah, and N. Ahmat, „A review of vision based defect detection using image processing techniques for beverage manufacturing industry,“ *Jurnal Teknologi (Sciences & Engineering)*, vol. 81, no. 3, 2019.

[5] H. Zhu, „New algorithm of liquid level of infusion bottle based on image processing,“ in *2009 International Conference on Information Engineering and Computer Science*, IEEE, 2009, pp. 1–4.

[6] Y. Peng, X. Zhang, D. Li, Y. Chen, and Y. Shen, „An infusion liquid level detection method based on improved roi segmentation and horizontal lines modification,“ in *2022 41st Chinese Control Conference (CCC)*, IEEE, 2022, pp. 6208–6215.

[7] Y. Ma and Z. Mao, „Liqd: A dynamic liquid level detection model under tricky small containers,“ *arXiv preprint arXiv:2403.08273*, 2024.

[8] K. J. Pithadiya, C. K. Modi, and J. D. Chauhan, „Comparison of optimal edge detection algorithms for liquid level inspection in bottles,“ in *2009 Second international conference on emerging trends in engineering & technology*, IEEE, 2009, pp. 447–452.

[9] A. Xompero, S. Donaher, V. Iashin, *et al.*, „The corsmal benchmark for the prediction of the properties of containers,“ *IEEE Access*, vol. 10, pp. 41 388–41 402, 2022.

[10] S. Eppel and T. Kachman, „Computer vision-based recognition of liquid surfaces and phase boundaries in transparent vessels, with emphasis on chemistry applications,“ *arXiv preprint arXiv:1404.7174*, 2014.

[11] R. Vine, „Google scholar,“ *Journal of the Medical Library Association*, vol. 94, no. 1, p. 97, 2006.

[12] Z. Liu, F. Liu, Q. Zeng, X. Yin, and Y. Yang, „Estimation of drinking water volume of laboratory animals based on image processing,“ *Scientific Reports*, vol. 13, no. 1, p. 8602, 2023.

[13] M. Devare, „Parallel image processing for liquid level detection,“ in *International Conference on Computing in Engineering & Technology*, Springer, 2022, pp. 372–382.

[14] S. Eppel, H. Xu, Y. R. Wang, and A. Aspuru-Guzik, „Predicting 3d shapes, masks, and properties of materials inside transparent containers, using the transproteus cgi dataset," *Digital Discovery*, vol. 1, no. 1, pp. 45–60, 2022.

[15] S. Eppel, H. Xu, M. Bismuth, and A. Aspuru-Guzik, „Computer vision for recognition of materials and vessels in chemistry lab settings and the vector-labpics data set," *ACS central science*, vol. 6, no. 10, pp. 1743–1752, 2020.

[16] S. Eppel, H. Xu, and A. Aspuru-Guzik, „Computer vision for liquid samples in hospitals and medical labs using hierarchical image segmentation and relations prediction," *arXiv preprint arXiv:2105.01456*, 2021.

[17] A. Kirillov, E. Mintun, N. Ravi, *et al.*, „Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.

[18] J. Canny, „A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[19] S. S. Rumma, „Comparative study of edge detection algorithm techniques with positive and negative details," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 12, pp. 4252–4256, 2014.

[20] I. Sobel, G. Feldman, *et al.*, „A 3x3 isotropic gradient operator for image processing," *a talk at the Stanford Artificial Project in*, vol. 1968, pp. 271–272, 1968.

[21] D. Marr and E. Hildreth, „Theory of edge detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.

[22] D. Ziou and S. Tabbone, „Edge detection techniques-an overview," *Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications*, vol. 8, no. 4, pp. 537–559, 1998.

[23] J. Jing, S. Liu, G. Wang, W. Zhang, and C. Sun, „Recent advances on image edge detection: A comprehensive review," *Neurocomputing*, vol. 503, pp. 259–271, 2022.

[24] H.-D. Cheng, X. H. Jiang, Y. Sun, and J. Wang, „Color image segmentation: Advances and prospects," *Pattern recognition*, vol. 34, no. 12, pp. 2259–2281, 2001.

[25] W. Ladys law Skarbek and A. Koschan, „Colour image segmentation a survey," *IEEE Transactions on circuits and systems for Video Technol-ogy*, vol. 14, no. 7, pp. 1–80, 1994.

[26] W. N. Jasim and R. J. Mohammed, „A survey on segmentation techniques for image processing.," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 17, no. 2, 2021.

[27] Y. Yu, C. Wang, Q. Fu, *et al.*, „Techniques and challenges of image segmentation: A review," *Electronics*, vol. 12, no. 5, p. 1199, 2023.

[28] N. Otsu *et al.*, „A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[29] B. Sankur and M. Sezgin, „Image thresholding techniques: A survey over categories," *Pattern Recognition*, vol. 34, no. 2, pp. 1573–1607, 2001.

[30] H. Yin, A. Aryani, S. Petrie, A. Nambissan, A. Astudillo, and S. Cao, „A rapid review of clustering algorithms," *arXiv preprint arXiv:2401.07389*, 2024.

[31] X. Jin and J. Han, „K-means clustering,“ *Encyclopedia of machine learning*, pp. 563–564, 2011.

[32] G. B. Coleman and H. C. Andrews, „Image segmentation by clustering,“ *Proceedings of the IEEE*, vol. 67, no. 5, pp. 773–785, 1979.

[33] I. A. Yusoff and N. A. M. Isa, „Two-dimensional clustering algorithms for image segmentation,“ *WSEAS transactions on computers*, vol. 10, no. 10, pp. 332–342, 2011.

[34] R. Adams and L. Bischof, „Seeded region growing,“ *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[35] N. Ikonomatakis, K. Plataniotis, M. Zervakis, and A. Venetsanopoulos, „Region growing and region merging image segmentation,“ in *Proceedings of 13th International Conference on Digital Signal Processing*, IEEE, vol. 1, 1997, pp. 299–302.

[36] Z. Karim, N. R. Paiker, M. A. Ali, G. Sorwar, and M. Islam, „Pattern based object segmentation using split and merge,“ in *2009 IEEE International Conference on Fuzzy Systems*, IEEE, 2009, pp. 2166–2169.

[37] Y. Y. Boykov and M.-P. Jolly, „Interactive graph cuts for optimal boundary & region segmentation of objects in nd images,“ in *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, IEEE, vol. 1, 2001, pp. 105–112.

[38] Y. Boykov and V. Kolmogorov, „An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,“ *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[39] F. Yi and I. Moon, „Image segmentation: A survey of graph-cut methods,“ in *2012 international conference on systems and informatics (ICSAI2012)*, IEEE, 2012, pp. 1936–1941.

[40] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[41] S. Naz, H. Majeed, and H. Irshad, „Image segmentation using fuzzy clustering: A survey,“ in *2010 6th international conference on emerging technologies (ICET)*, IEEE, 2010, pp. 181–186.

[42] F. Kurugollu, B. Sankur, and A. E. Harmanci, „Color image segmentation using histogram multithresholding and fusion,“ *Image and vision computing*, vol. 19, no. 13, pp. 915–928, 2001.

[43] M. Li, L. Wang, S. Deng, and C. Zhou, „Color image segmentation using adaptive hierarchical-histogram thresholding,“ *PloS one*, vol. 15, no. 1, e0226345, 2020.

[44] K. S. Tan and N. A. M. Isa, „Color image segmentation using histogram thresholding–fuzzy c-means hybrid approach,“ *Pattern recognition*, vol. 44, no. 1, pp. 1–15, 2011.

[45] A. M. Hafiz and G. M. Bhat, „A survey on instance segmentation: State of the art,“ *International journal of multimedia information retrieval*, vol. 9, no. 3, pp. 171–189, 2020.

[46] J. Long, E. Shelhamer, and T. Darrell, „Fully convolutional networks for semantic segmentation,“ in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[47] W. Ma and J. Lu, „An equivalence of fully connected layer and convolutional layer," *arXiv preprint arXiv:1712.01252*, 2017.

[48] K. He, G. Gkioxari, P. Dollár, and R. Girshick, „Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[49] R. Girshick, J. Donahue, T. Darrell, and J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[50] M. Egmont-Petersen, D. de Ridder, and H. Handels, „Image processing with neural networks—a review," *Pattern recognition*, vol. 35, no. 10, pp. 2279–2301, 2002.

[51] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, „A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, pp. 87–93, 2018.

[52] J. Zhang, „Survey on monocular metric depth estimation," *arXiv preprint arXiv:2501.11841*, 2025.

[53] Y. Ming, X. Meng, C. Fan, and H. Yu, „Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[54] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian, „Monocular depth estimation based on deep learning: An overview," *Science China Technological Sciences*, vol. 63, no. 9, pp. 1612–1627, 2020.

[55] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, „Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[56] L. Yang, B. Kang, Z. Huang, X. Xu, J. Feng, and H. Zhao, „Depth anything: Unleashing the power of large-scale unlabeled data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 371–10 381.

[57] C. Godard, O. Mac Aodha, and G. J. Brostow, „Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 270–279.

[58] Y. Cao, C. Shen, and H. T. Shen, „Exploiting depth from single monocular images for object detection and semantic segmentation," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 836–846, 2016.

[59] J. Liu, H. Ma, Y. Guo, *et al.*, „Monocular depth estimation and segmentation for transparent object with iterative semantic and geometric fusion," *arXiv preprint arXiv:2502.14616*, 2025.

[60] W. Sun, Z. Gao, J. Cui, B. Ramesh, B. Zhang, and Z. Li, „Semantic segmentation leveraging simultaneous depth estimation," *Sensors*, vol. 21, no. 3, p. 690, 2021.

[61] J. Shen and S. Castan, „An optimal linear operator for step edge detection," *CVGIP: Graphical models and image processing*, vol. 54, no. 2, pp. 112–133, 1992.

[62] S. Eppel, „Tracing the boundaries of materials in transparent vessels using computer vision," *arXiv preprint arXiv:1501.04691*, 2015.

[63] S. Eppel, „Tracing liquid level and material boundaries in transparent vessels using the graph cut computer vision approach,“ *arXiv preprint arXiv:1602.00177*, 2016.

[64] Y. Wu, H. Ye, Y. Yang, Z. Wang, and S. Li, „Liquid content detection in transparent containers: A benchmark,“ *Sensors*, vol. 23, no. 15, p. 6656, 2023.

[65] R. Mottaghi, C. Schenck, D. Fox, and A. Farhadi, „See the glass half full: Reasoning about liquid containers, their volume and content,“ in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1871–1880.

[66] Y. Lin and S. Zhou, „Neural numerical estimation of liquid volume in transparent containers via a single rgb-d image,“ in *2024 7th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, IEEE, 2024, pp. 379–384.

[67] H. K. Singh, S. K. Chakroborty, H. Talukdar, N. M. Singh, and T. Bezboruah, „A new non-intrusive optical technique to measure transparent liquid level and volume,“ *IEEE Sensors Journal*, vol. 11, no. 2, pp. 391–398, 2010.

[68] K. Suemori, Y. Komatsu, and T. Nobeshima, „Flange-type liquid-level sensor based on laser light reflection,“ *Sensors International*, vol. 4, p. 100 230, 2023.

[69] C. M. Ranieri, A. V. Foletto, R. D. Garcia, *et al.*, „Water level identification with laser sensors, inertial units, and machine learning,“ *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107 235, 2024.

[70] S. F. Ali and N. Mandal, „Design and development of an electronic level transmitter based on hydrostatic principle,“ *Measurement*, vol. 132, pp. 125–134, 2019.

[71] Z. Wu, Y. Huang, K. Huang, K. Yan, and H. Chen, „A review of non-contact water level measurement based on computer vision and radar technology,“ *Water*, vol. 15, no. 18, p. 3233, 2023.

[72] A. Le, „Ultrasonic sensing basics for liquid level sensing, flow sensing, and fluid identification applications,“ *Texas instruments application report, SNAA220A*, pp. 1–10, 2015.

[73] G. Prathyusha and B. R. Murthy, „Embedded based level measurement and control using float sensor,“ *Int. J. Adv. Res. Electric. Electron. Instru. Eng.*, vol. 4, pp. 2310–2315, 2015.

[74] Y. Ren, B. Luo, X. Feng, Z. Feng, Y. Song, and F. Yan, „Capacitive and non-contact liquid level detection sensor based on interdigitated electrodes with flexible substrate,“ *Electronics*, vol. 13, no. 11, p. 2228, 2024.

[75] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[76] R. K. Arora, *Optimization: algorithms and applications*. CRC press, 2015.

[77] S. Ruder, „An overview of gradient descent optimization algorithms,“ *arXiv preprint arXiv:1609.04747*, 2016.

[78] S. H. Haji and A. M. Abdulazeez, „Comparison of optimization techniques based on gradient descent algorithm: A review,“ *PalArch's Journal of Archaeology of Egypt/Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021.

[79] D. P. Kingma and J. Ba, „Adam: A method for stochastic optimization,“ *arXiv preprint arXiv:1412.6980*, 2014.

[80] P. P. Ghosal, *Non-gradient based optimization*, `https://www.researchgate.net/publication/348732530_Non-Gradient_Based_Optimization`, 2021.

[81] J. Kennedy and R. Eberhart, „Particle swarm optimization,“ in *Proceedings of ICNN'95-international conference on neural networks*, ieee, vol. 4, 1995, pp. 1942–1948.

[82] N. Metropolis and S. Ulam, „The monte carlo method,“ *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[83] J. A. Nelder and R. Mead, „A simplex method for function minimization,“ *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[84] J.-B. Weibel, P. Sebeto, S. Thalhammer, and M. Vincze, „Challenges of depth estimation for transparent objects,“ in *International Symposium on Visual Computing*, Springer, 2023, pp. 277–288.

[85] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, „A review on deep learning techniques applied to semantic segmentation,“ *arXiv preprint arXiv:1704.06857*, 2017.

[86] R. M. Haralick, S. R. Sternberg, and X. Zhuang, „Image analysis using mathematical morphology,“ *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 532–550, 1987.

[87] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, *et al.*, „Meshlab: An open-source mesh processing tool.,“ in *Eurographics Italian chapter conference*, Salerno, vol. 2008, 2008, pp. 129–136.

[88] Dawson-Haggerty et al., *Trimesh*, version 3.2.0, 2025. [Online]. Available: `https://trimesh.org/`.

[89] M. Suchi, *3d-dat: 3d scene annotation and dataset toolkit*, `https://github.com/markus-suchi/3D-DAT`, Accessed: 2025, 2024.

[90] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, „The quickhull algorithm for convex hulls,“ *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[91] S. Eppel, H. Xu, Y. R. Wang, and A. Aspuru-Guzik, „Predicting 3d shapes, masks, and properties of materials, liquids, and objects inside transparent containers, using the transproteus cgi dataset,“ *arXiv preprint arXiv:2109.07577*, 2021.

Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, April 2025

[illegible]

Elisabeth Fabini