Quantifying the Robustness of Image Segmentation Algorithms for Medical Assistance Systems

Prof. Dr. rer. nat. Alexander Windberger (HHN)
Dr. Annika Reinke (DKFZ)

Lennart Kremp
kremp-len@gmx.de
Matr.-Nr.: 209437

September 22, 2023, Heilbronn

Abstract

This thesis presents a comprehensive study and implementation of a specialized framework, referred to as the "RobustnessTool", developed to evaluate the robustness of various image segmentation algorithms, particularly focusing on medical image analysis. The tool aims to provide a standardized measure of algorithm robustness against varied, domain-relevant transformations.

The user can upload any image segmentation algorithm to the tool for evaluation that complies with predetermined criteria, such as being encapsulated within a docker image. The tool operates by allowing users to upload test images and their corresponding reference masks.

The segmentation models are then evaluated based on these transformed images, and their accuracy is measured using the Dice Similarity Coefficient, resulting in a robustness graph for every evaluated transformation.

This framework emphasizes the importance of selecting domain-relevant transformations, in order to generate results that mirror practical, real-world application scenarios. An extensive analysis is performed to evaluate the performance of models on transformed images compared to original, uncorrupted images, which is imperative for discerning the robustness and reliability of segmentation algorithms in diverse conditions.

The framework is demonstrated by evaluating the *Isensee* model (Isensee & Maier-Hein, 2020) submitted to the ROBUST-MIS challenge 2019 (DKFZ et al., 2019a), by members of the German Cancer Research Center.

This thesis, by implementing and assessing a robustness evaluation tool, seeks to contribute to the ongoing advancements in medical image segmentation, offering insights and methodologies that can aid in the development and selection of more reliable and robust image segmentation algorithms, essential for enhancing the accuracy and efficiency of medical diagnoses and treatments.

Remarks and Acknowledgments

I would like to extend my thanks to Prof. Dr. Alexander Windberger for his guidance and support during the course of this project. My thanks also go to Dr. Annika Reinke of the German Cancer Research Center (DKFZ) for providing the image data set and the evaluated model, which greatly assisted the research. This document has benefited from restructuring and wording enhancements made possible by ChatGPT-4 by OpenAI and Microsoft's Bing AI.

Contents

1 Introduction

1.1 Background and Motivation

Image segmentation, a pivotal technique in the medical domain, enables the detection of tissue abnormalities, such as tumors, and assists in segmenting medical imaging from X-Rays, CT scans, and MRIs (Kürbig & Sauter, 2005). Another application of image segmentation is to replace current tracking systems that require additional hardware like magnetic field based tracking (Bianchi et al., 2019) or optical trackers (Koivukangas et al., 2013). While these applications are groundbreaking, the success of image segmentation hinges on two critical pillars - accuracy and robustness. Accurate segmentation ensures that the features of medical images are correctly identified, facilitating precise diagnosis. But in the medical field, not only accuracy is important, but also the robustness of the predictions. For a physician to make informed decisions aided by an image segmentation algorithm, it needs to be reliable in a wide variety of different settings. Optimally, the algorithm rejects all images it can not make a reliable prediction on, and it is additionally ensured that the number of rejected images is as small as possible (Deserno, 2008). The concept of robustness in image segmentation, will be delved into in greater detail in subsequent chapters. To ensure that a given algorithm is fit to be used in the medical field, there is a need for an independent certification. Therefore, there has to be an objective way to quantify the robustness of image segmentation algorithms. This study seeks to find a transparent measure by developing a comprehensive testing framework able to evaluate them in different scenarios. Anyone releasing new image segmentation algorithms for the medical field could then use this framework to certify their product. This helps medical practitioners make informed decisions with an easy-to-read robustness scale and improves patient safety by ensuring only adequate software is used. Developers could benefit from the transparent robustness scale, because it shows their software meets standards for accuracy and robustness, which has long been the norm for medical devices, because it reduces the risk of recalls due to defects and potential lawsuits (Maze, 2022). The platform that is being developed quantifies robustness, has an open interface to enable broad usage and can be extended to test the image segmentation with different types of interference effects. Because the process is automated, it helps to provide a cheap and easy way to test an image segmentation algorithm for its readiness to be deployed in the medical field.

1.2 Problem Statement and Research Objective

Robustness in the context of image segmentation is the ability of an image segmentation algorithm to resist to noise (Vacavant, 2017). Building a framework that accurately quantifies this robustness is challenged by three key points.

First of all, the variability in interference. In the context of medical image segmentation challenges, robustness also means resisting other types of deterioration in image quality. The types of interference effects can vary widely depending on the use case of the algorithm. For example, endoscopic images tend to have vignetting effects because of inhomogeneous lighting and a wide viewing angle. While this poses little problem to a human physician, it can cause difficulties for many computer vision applications (Bergen et al., 2015). When trying to correct this effect by setting the intensity of the light source to a high level, a glare effect is seen from the medical instruments, which is another common interference effect. There are many more of these effects in the real world, so it can be hard to accurately measure the robustness of a given algorithm. This includes less obvious medical interference effects like smoke produced by tissue that is burned by hot medical instruments, in order to stop bleeding. This smoke can substantially decrease the quality of endoscopic images (Zhou et al., 2022), but is not commonly tested in robustness benchmarks.

Furthermore, an accurate result that represents a lot of real-world scenarios requires the evaluation of numerous test images in order for the test to have statistical significance. Each of these test images needs to be transformed to simulate different kinds of interference effects. To facilitate testing on a suitable hardware in a reasonable timeframe, the platform needs to have an efficient way of data processing. At the same time, the application should enable the user to test with a very high amount of test images and sampling steps of the interference effects, which would result in a much longer testing period. The main challenge is to calculate a transparent and easy to understand robustness scale that displays the statistics with optimized comprehensibility without sacrificing essential information in the process. This provides an easy overview of the suitability for a given segmentation challenge. To make the calculation as accurate as possible, many kinds of effects that occur, need to be taken into account. The framework aims to measure the robustness of the given algorithm to different interference effects in varying levels of intensity. The challenge lies in developing a testing mechanism for a diverse array of image segmentation algorithms across various domains, the framework under development must possess an open interface to ensure seamless communication with any model tested by the framework. Furthermore, it needs to have an easy way to add custom interference effects, as so to provide a testing process that is tailored to the domain specific requirements. Lastly, the platform needs to have an easy way of providing the test images, to which the transformations are applied, and their respective references.

The research objective of this framework is to lay the groundwork for an automatic certification process for image segmentation algorithms in the medical field. In the context of this thesis, the benchmarking will focus on a Medical Instrument Segmentation algorithm that works on endoscopic images. Because of the medical focus, the interference effects are mostly relevant to the healthcare domain, but the benchmarking application can be extended to cover many different transformations.

1.3 Structure of the Thesis

The introduction provides an overview of the topic of image segmentation and its significance in the healthcare domain. It also outlines the necessity of objective testing applications for robustness. The "Background" section explains basic terms and concepts related to image segmentation and explores current research in this area. The specific challenges associated with measuring and calculating robustness are detailed in this section. The methodology used in this thesis is described, including the model to be benchmarked, the testing data set, and the platform itself. The implementation of the interface and the augmentation of the data set are explained in detail, as well as the selection of parameters and statistical metrics used to calculate robustness. The benchmarking process is performed and explained from start to finish, and the results are discussed in the context of existing research. The methodology is evaluated for potential limitations, and suggestions for future research and practical applications are provided. The conclusion summarizes the key findings, emphasizing the contribution of this thesis to the field of medical image segmentation

2 Background

2.1 Image Segmentation

Image Segmentation is the process of extracting useful information from an image by partitioning it into objects or regions of interest. There are many different applications for image segmentation that include helping medical science from diagnosis to procedures, video surveillance and enabling object recognition (Mittal et al., 2021), e.g., for determining whether the crop is ready for harvest.

This procedure checks every pixel and determines if it belongs to an object of interest, based on specific characteristics. This results in boundaries between different objects or regions (“Segmentation”, 2005). This makes it easier to work with for the computer, because the image is represented with a few significant segments only, instead of thousands of pixels (Mittal et al., 2021). There are three subcategories of image segmentation techniques, which use different approaches to partition an image into multiple segments. The semantic segmentation assigns a categorical label to each pixel of the image (Guo et al., 2020). This means it decides for every pixel, in which of the given classes it belongs. This technique is used in medical imaging by identifying if a pixel belongs to the category “healthy tissue” or “tumor” (Potnuru & Naick, 2023). This is a special case known as binary segmentation, which simplifies the task to classifying each pixel into one of just two categories: background, usually represented by the value 0, and foreground, represented by the value 1. This streamlined approach is particularly useful in applications where the primary interest is to distinguish a specific object or feature from the surrounding environment. Given its simpler nature, binary segmentation often requires fewer computational resources and can be faster to execute compared to multi-class segmentation methods. The second category is instance segmentation, which evaluates every pixel based on which object instance it belongs to. It also marks the edges of each instance in the image (Sharma et al., 2022). One application would be in a setting where the path of multiple different objects needs to be tracked, like different runners in a sprint, an example of which is provided by Cheng (2022). Finally, there is panoptic segmentation, which combines the procedures above by assigning a class label to each pixel and also detecting each object instance (Kirillov, 2018). Deep Learning Models for image segmentation can use procedures from one or multiple of these categories to achieve their goal. Examples of such models include the SegNet for Image Segmentation (Badrinarayanan, 2015), Mask R-CNN for Instance Segmentation (MathWorks, 2023) and the Panoptic FPN for Panoptic Segmentation (Kirillov, 2019). In the medical domain, the U-Net is seen as the foundational model in the field of medical imaging, particularly for tasks related to semantic segmentation. It is worth mentioning that in practical applications, semantic segmentation models are often adapted for instance segmentation tasks (Shen et al., 2023). This adaptation is typically accomplished through post-processing techniques like connected component analysis, which enables the model to differentiate between individual instances of the same class, such as distinguishing multiple tumors in a single image.

While the performance of a segmentation algorithm can vary depending on the specific image, there are some general factors that influence the results. For example, variations in illumination occur due to inhomogeneous lighting conditions in the image, resulting in different intensity values of the pixels. This can be challenging, because approaches that are based on the intensity value of the pixel, might not cluster two pixels of the same object together, because their values are different. Another challenge for image segmentation is background complexity. This can lead to boundaries not being clearly defined and overall poor performance (Mittal et al., 2021). Background complexity can be introduced by imaging artifacts like noise, which can severely deteriorate the performance of the algorithm. Other artifacts that interfere with image segmentation include motion blur (Agrawal & Raskar, 2007) and glare from the reflection of light on metallic instruments, which itself can cause overexposure and a loss of contrast in the image. Moreover, calibration issues like a wrong white balance of the imaging device or a lossy compression algorithm can deteriorate the results of most segmentation methods. Another interference effect specific to the medical domain, namely endoscopy, is smoke, which is a byproduct of coagulation tools (Reiter, 2020) and diminishes the quality of endoscopic vision. Image segmentation algorithms, that analyze endoscopic images and support the medical practitioners, could have difficulties because of the obstructed field of view.

The requirements for measuring the performance of a given image segmentation algorithm can vary depending on the use case. Therefore, there are several different evaluation parameters that researchers have used for quantitative evaluation of an image segmentation method. Evaluation methods can be categorized as shown in Figure 1.

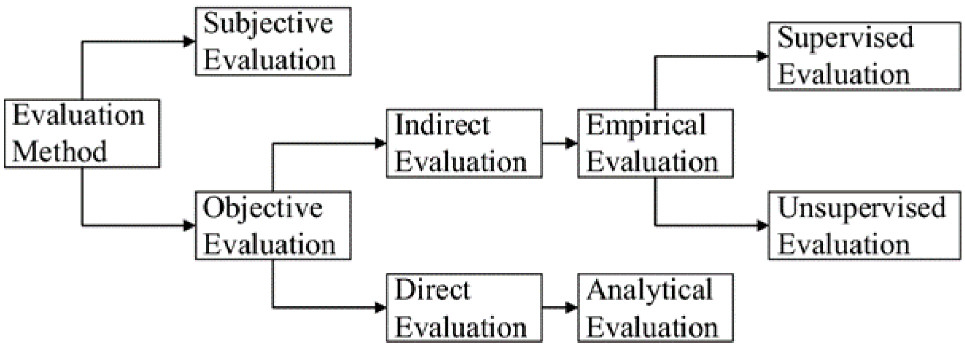

Figure 1: Categories of Performance Evaluation Methods (Wang et al., 2020)

This section will focus on the objective evaluation branch of the evaluation methods, because the subjective evaluation is performed by a human, which makes them unable to be automated. Furthermore, the direct evaluation branch is ignored, because it evaluates the segmentation algorithm itself, instead of its segmentation results. Finally, the unsupervised evaluation category is ignored, because in contrast to the supervised evaluation, which uses reference images also called “references”, it has a lower evaluation accuracy, due to the lack of comparison (Wang et al., 2020). In the following chapter, some of the most common (Müller et al., 2022) metrics in the Supervised Evaluation category, will be explored.

The simplest metrics (1, 2, 3) can be calculated by using the confusion matrix, which shows the number of correctly clustered patterns, consisting of true positives (TP) and true negatives (TN), as well as the wrongly clustered patterns, which include the false positives (FP) and the false negatives (FN). From the confusion matrix, these and more metrics can be calculated:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

The Accuracy (3) shows how often the segmentation method is correct overall, the precision (1) shows how often it is correct when predicting the target class and the recall (2) shows whether it can find all objects of the target class (Team, n.d.). Furthermore, the Sensitivity and Specificity can easily be calculated from the Confusion matrix (Ragan, 2018).

Another popular metric is the intersection over Union (IoU), also known as the Jaccard index (4). It is used to measure the similarity between two arbitrary shapes, by looking at the overlap between a predicted bounding box and the reference bounding box. To be exact, this approach divides the intersection by the union of the bounding boxes to calculate the final value. "A" represents the prediction and "B" represents the reference.

$$IoU = \frac{A \cap B}{A \cup B} = \frac{|I|}{|U|} \tag{4}$$

The Dice Similarity Coefficient also measures the overlap between two structures and is closely related to the IoU. The DSC or the IoU should be used in most cases of image segmentation. The metric is not suited for very tiny structures in combination with a noisy reference, because even a single-pixel difference can hugely impact the metric score (Reinke, 2023). The DSC is often used in the medical field, whereas the IoU is used in other computer vision domains.

Finally, there is the Average Hausdorff distance, which is used to calculate the distance between two point sets. In the context of image segmentation, it is used to compare the prediction of the segmentation algorithm to the reference images. The average Hausdorff distance between two finite point sets X and Y is defined by the following Equation (5).

$$AverageHausdorffdistance = \left(\frac{GtoS}{G} + \frac{StoG}{S} \right) /2 \tag{5}$$

(Aydin et al., 2021)

“Where "GtoS" is the directed average Hausdorff distance from reference to segmentation, "StoG" is the directed average Hausdorff distance from segmentation to reference, "G" is the number of voxels in the reference, and "S" is the number of voxels in the segmentation.” (Aydin et al., 2021). Some studies show that the average Hausdorff distance has a ranking error, making it less suitable to compare segmentation results in some situations. The Balanced average Hausdorff distance is suggested as a replacement, which fixes this issue (Aydin et al., 2021). All of the discussed metrics could be used by the framework to evaluate the performance of the model.

2.2 Robust Image Segmentation

In the context of image segmentation algorithms, robustness refers to the ability of an algorithm to consistently maintain good results in many different circumstances. This includes different settings of the objects of interest, but also artifacts in the input images, like noise, glare, motion blur or any other unwanted effects. In short, a robust system refers to "preserving model performance under naturally-induced image corruptions or alterations.” (Drenkow, 2021). The consequences of image segmentation algorithms that are missing robustness, are far-reaching, especially in the healthcare sector. Missing robustness means that certain changes to image data can completely alter the output of an image classification (Oala et al., 2021). In contrast to a human expert, whose decision will not be altered by small pixel-changes, an algorithm could be fooled by only a small artifact produced by incorrect calibration, movement by the patient, or other interference effects. Because a diagnosis can have far-reaching consequences, an algorithm missing robustness could potentially classify patients as having a tumor, even though they are fully healthy. This could lead to unnecessary radiation and other risks to patient health. An even worse scenario would be the reverse, where a patient with a tumor is sent home and misses the window of effective treatment. For this reason, algorithms in the healthcare sector need to be as transparent as possible, and be tested in as many scenarios as possible, to prevent damage to human health. This can be extremely challenging, because especially machine-learning models are often referred to as “black boxes”, because their complexity is so great, even the researchers who design them can not fully understand how they make predictions (Zewe, 2022). To counteract the challenge of a black box, where one can only see the inputs and the outputs, but not the inner working of the algorithm, a trial and error approach is needed.

To understand if the algorithms predictions are robust enough, a framework is being developed, which slightly varies the input data and measures the change in the output prediction. With a large enough sample size, over time, the framework can measure how resilient the examined algorithm is against common interference effects. Existing studies have applied corruptions like Gaussian noise, impulse noise, motion blur, snow, fog, brightness, compression artifacts and more (Hendrycks, 2019) (Altindis, 2021) to the test images. These artifacts are often drawn from four main categories: Noise, Blur, Weather and Digital. After testing the corrupted images, a mean Corruption Error for each classifier is calculated, by normalizing the average of the top-1 errors of a classifier across all corruptions by the top-1 error of a baseline classifier on the same corruptions.

A limitation of this method is that only common general interference effects are tested. In different fields of application, different artifacts are common. That is why it is not sufficient to test any segmentation algorithm by simply running a predefined list of test artifacts. A framework that accurately measures the robustness of any given image segmentation algorithm, needs to be tailored to the specific domain requirements.

In a paper where 53 studies on robustness were evaluated, it was found that most studies did not provide a formal mathematical definition of robustness, which makes it challenging to establish a universally accepted benchmarking criteria for robustness (Drenkow, 2021). Furthermore, metrics like mean corruption Error and relative Corruption Error rely on the AlexNet model as reference. Using a single model as a benchmark might not be a universally applicable measurement.

While many of the analyzed studies incorporate both different types of corruption and varying severity, they assume an equal likelihood and weighting between these interference effects. This may not be representative of the real world, where certain corruptions could be more or less common. Moreover, some corruptions could have more significant consequences than others. A weighted robustness testing evaluation method could be more relevant for practical applications.

Most studies looked at robustness in the context of image classification, in contrast to only a few studies considering object detection and segmentation. (Drenkow, 2021). Furthermore, it is important to note that the analyzed types of robustness tests do not check for adversarial attacks. These attacks are inputs to machine learning models, that an attacker designed to cause a mistake in the prediction of the model. They work like optical illusions for machines (OpenAI, 2017) and can be undetectable to the human eye. Because the studies only test predefined corruptions to the input image, an algorithm that was evaluated to be robust, could still be fooled by an attacker creating an adversarial example and feeding it to the model as an input image. A real-world example of this would be adding small stickers to a road sign in order to confuse road sign recognition systems of autonomous cars (Ren et al., 2021). The framework that is being developed will also omit this branch of robustness evaluation, because there are already well-developed tools that check for these kinds of attacks (Zizzo et al., 2021).

2.3 Image Transformations

In imaging, particularly in medical imaging, obtaining a perfect, artifact-free image is rare. Variability in equipment, environment and subjects, along with poor calibration or excessive movement by the patient, can introduce alterations to the captured images. Thus, a robust model should not only work with pristine images, but also consistently and correctly interpret images with common corruptions. Image transformations provide an easy and cheap way to simulate these real-world corruptions and thus transparently test the robustness of image segmentation algorithms. The benefit of simulating the corruptions is consistency while testing. While real-world artifacts can be random and unpredictable, image transformations allow the controlled and consistent alterations of images. This standardized procedure allows the comparison of different algorithms based on their performance against the same set of corruptions. While there are many different avenues to apply corruptions to a given test image, this section will focus on popular python libraries, because python has become the most popular programming language for data scientists (Luna, 2023). The most well-known library is OpenCV, which is not just for image transformation but a whole suite of computer vision tasks (OpenCV, 2020). It contains more than 2500 optimized algorithms and has capabilities in image processing, feature detection, object detection and video analysis. PIL or Python Imaging Library was originally developed as a free library for image-processing tasks in Python. Pillow is an actively maintained fork of PIL, which constantly adds new features. Pillow supports Image Manipulations and Enhancements, Drawing, Filtering and more. It is often used for script-based modifications of images (Wikipedia contributors, 2023). The Scikit-image library is another Python package dedicated to image processing, which natively uses NumPy arrays as image objects (3.3. Scikit-image: Image Processing — Scipy Lecture Notes, n.d.). It was released for use in research, education and industry under the liberal Modified BSD open source license and is still actively being updated (Van Der Walt et al., 2014). These libraries provide an easy way to implement different kinds of transformations. Choosing relevant transformations is crucial when testing the robustness of an image segmentation algorithm. The choice of transformation to apply should be based on the most common corruptions expected in the specific domain. For example, a robustness test on the transformation type "rain" would not be relevant for an image segmentation algorithm classifying tumorous tissue on an MRI scan. Another factor to consider is the degree to which the transformation is applied and the number of sampling steps. The amount of, e.g., noise or blur can drastically change classification difficulty to an algorithm. Selecting appropriate upper and lower values for the transformations, along with a reasonable number of sampling steps, is crucial for conducting accurate and objective robustness tests. Furthermore, in real-world scenarios, images might suffer from multiple types of corruptions simultaneously, so testing with a combination of transformation can offer insights into more complex challenges. It is important to note, however, that the complexity increases with the amounts of transformations, sampling steps, and transformation combinations while testing.

2.4 Measures of Descriptive Statistics

The robustness of an image segmentation algorithm is determined from how consistently they perform on different images. To measure this consistency and reliability, some statistical metrics are needed. This section will explore the metrics and methods essential for quantifying robustness. While sophisticated statistical methods will be employed in this research, foundational descriptive statistics serve as the backbone for understanding the data. Here, these descriptive metrics, within the scope of quantifying algorithm robustness, will briefly be contextualized. For measuring the central tendency, there is the mean, which is the most commonly used method for finding the average. It is calculated by adding up all values and dividing the sum by the number of values (Bhandari, 2023). In the context of the evaluation framework, the mean could be used to find the average deterioration of a given simple step of a given transformation, by calculating the mean of every test image at that sampling step. Next there is the median, which is the

value that is in the middle of a data set. It is gathered by ordering each value from smallest to biggest and selecting the number in the middle. If there are two middle numbers, their mean is taken. This can be used like the mean, with the exception that the median is much more stable to extreme values. The last measure of central tendency used is the mode, representing the most frequent value. To find the mode, the data set needs to be ordered from low to high, then the most frequent value is selected. It could be useful to know, what is the most likely deterioration in prediction accuracy for a given transformation. The next category of metrics is the measures of variability or spread. These give a sense of how spread out the values are. The range shows the distance between the most extreme values. To find the range, the lowest value needs to be subtracted from the highest. A high range means that the algorithm is not very robust to this type of interference effect. The second metric is the standard deviation, which is commonly displayed together with the mean. To calculate the standard deviation, the mean has to be gathered first. After that, the mean is subtracted from each value to get the deviation from the mean. Each of these deviations are squared and summed up. Finally, this sum is divided by $n - 1$ and the square root is calculated. The last metric is often displayed together with the median inside a boxplot and is called interquartile range. It measures the spread of the middle half of the distribution (Bhandari, 2023). The median and mean are called point measures, while the corresponding and interquartile range and standard deviation are called dispersion measures.
These measures will be used in the framework to paint a transparent picture of the robustness of the evaluated algorithm. For example, the evaluation graphs for each transformation display the mean for all of the mentioned statistical metrics for all test images and sampling steps of the given transformation. Additionally, other discussed metrics are utilized to to provide a comprehensive overview over the robustness regarding the specific interference effect. With the abundance of statistical information, an experienced statistician or data scientist can get a deep understanding on how susceptible a given segmentation algorithm is when analyzing corrupted images.

3 Methodology

3.1 Research Basis

This study is exploratory in nature, seeking to find an automated approach to quantifying the robustness of image segmentation algorithms. The framework applies different interference effects to images and calculates a robustness score based on the change in prediction accuracy, thereby providing an empirical measure of algorithmic robustness. The data source for building and testing this framework was the Robust Medical Instrument Segmentation (ROBUST-MIS) Challenge 2019. This Challenge was organized by the German Cancer Research Center (DKFZ) and the Heidelberg University Hospital. The Heidelberg Colorectal (HeiCo) data set (Maier-Hein et al., 2021) was recorded with a Karl Storz image 1 laparoscopic camera during daily routine procedures at the Heidelberg University Hospital (Roß et al., 2021). Image resolution was reduced to 960x540 to minimize storage usage. The data set consists of 10,040 images, extracted from 30 videos of minimally invasive surgical procedures. These videos consist of 10 rectal resection procedures, 10 proctocolectomy procedures and 10 procedures of sigmoid resection. Each of these extracted endoscopic video frames, has a labeling mask, with the medical instrument instances marked. The reference masks were generated in a multi-step process (Roß et al., 2021). First, the initial segmentations were performed by the company Understand AI. After the initial annotations were created, the annotations were reviewed for inconsistencies and an annotation guide was developed (Roß et al., 2021)(Appendix B). Next, a team of 14 engineers and four medical students reviewed all the annotations and refined them when necessary. In cases of ambiguity, a specialized team consisting of two engineers and one medical student generated a consensus annotation. Finally, a medical expert reviewed all annotations for possible errors. As the last step, a team comprising both a medical expert and an engineer made the final decisions regarding the labels.

3.2 Image Segmentation

The *Isensee* model based on the 2D UNet (Ronneberger et al., 2015) with residual encoder as the basic architecture, was trained by Isensee and Maier-Hein (2020) with the sum of Dice and cross-entropy loss and deep supervision. "During training, extensive data augmentation [was] used to increase the robustness." (Isensee & Maier-Hein, 2020). Optimizer of the underlying model was the SGD (Roß et al., 2021). The Stochastic gradient descent (SGD) optimizes a function by following noisy gradients with a decreasing step size (Mandt, 2017). This procedure reaches the optimum of the function, or a at least a local optimum, when it is nonconvex. The SGD is one of the most commonly used optimizers (Ruder, 2020) and has become crucial to modern machine learning (Mandt, 2017). To make its prediction, the model didn't use the video data present in the data set or any additional data. The loss function used is the sum of the Dice Similarity Coefficient. The model performs binary segmentation, which is in this context a subcategory of semantic segmentation. It accepts an endoscopic image as input and generates a binary segmentation mask, where '1' represents the presence of

medical instruments, and '0' signifies their absence. The dimensions of the processed image should match the input dimensions, and only one image layer should be present (DKFZ et al., 2019b).
The model allows effective segmentation and identification of surgical instruments in endoscopic images, which is an essential processing step for computer and robotics-assisted interventions (Isensee & Maier-Hein, 2020). By accurately segmenting the instruments, the model can aid in enhancing the precision of these interventions, potentially improving patient outcomes. The robustness of the model also enables it to handle a variety of imaging conditions, making it a versatile tool for different endoscopic procedures.
To reiterate, robustness in a real-world context means that the prediction accuracy is not significantly influenced when interference effects are present in the image. This is especially relevant for artifacts that are common in the algorithm's specific domain. To accurately test the robustness of a given segmentation algorithm, the interference effects need to be selected on the basis of relevancy in the domain. To enable a broad testing spectrum of domains, the framework that is being developed has the capability to apply several transformations to the test images. When testing with the framework, the robustness to these interference effects can be benchmarked out of the box, as shown in Table 1.

The optimal value ranges for the interference effects were found through an empirical approach, involving systematic variation and subsequent evaluation in an experimental setup. This amount of available effects is in line with previous studies on the subject of quantifying robustness (Drenkow, 2021) and covers the three major categories environment (glare, brightness), sensor (noise, motion blur, contrast) and rendering (resolution). Furthermore, it adds effects relevant to the medical domain like smoke or glare from metallic medical instruments. The specific values applied to the image depend on the number of selected sampling steps.

Table 1: Available interference effects

Interference Effect	**Value Ranges**	**Unit**
Glare	0.5-2	Brightness multiplier
Brightness	0.1-0.9	Gamma correction value
Darkness	1.5-5	Gamma correction value
Motion blur	10-20	Radius
Contrast	0.5-2	1/gamma value
Noise	0.01-0.2	Variance of Gaussian Noise
Sharpness	1-10	Edge reinforcement multiplier
Smoke	0.2-1	Transparency of smoke
Resolution	1.5-5	Size divisor
Vignette	0.05-0.4	Vignette size

The transformations were realized with image processing libraries in Python, namely Scikit-image, PIL and OpenCV. The details on how the different transformations were implemented using these libraries will be discussed in the following chapter “Implementation”. When using the framework, the user first has to specify the model to be evaluated. To enable the evaluation of many different kinds of image segmentation algorithms, a unified interface needed to be used. The tool accepts any model, that fulfills the following criteria.

- Needs to be inside a docker image
- Needs to accept two mounted folders at /input and /output
- Include a run_network.sh script that automatically runs the model

(DKFZ et al., 2019b)

Afterwards, the test images and their corresponding reference masks have to be uploaded to the tool. Next, the transformations to be applied and the number of sampling steps can be configured. It is important to choose relevant transformations, that can be seen naturally in images of the specific domain. After starting the testing process, the framework starts creating the test data set. This consists of every original test image provided by the user. Additionally, there is one modified image for each transformation and sampling step. This means when uploading 20 test images and selecting 5 transformations with 10 sampling steps, 1020 images are used for testing. This number is calculated by multiplying the number of test images with the number of transformations and sampling steps, and finally adding the original number of test images. Out of the 1020 images, 1000 have transformations in different intensities and 20 are the original test images. Now every image is segmented using the model to be evaluated. In the context of the ROBUST-MIS *Isensee* model, this means that 1020 binary segmentation masks in the form of PNG images are created, where every “1” represents a pixel of a medical instrument. After predicting the segmentation of every test image, the evaluation phase begins. Every prediction is compared to the reference using the Dice Similarity Coefficient. The DSC has been chosen as the evaluation metric, because it is most relevant in the medical domain. In future versions of the framework, all of the discussed metrics could be available to the user. This would add even more customizability and provide more accurate results, because the evaluation metric could be tailored to the domain of the evaluated model. This results in X accuracy scores for every data point, where X is the number of original test images and a data point is a transformation at a specific sampling step. Next, the mean of all accuracy scores belonging to one data point is calculated. When comparing it to the mean accuracy of the predictions on the images that were not corrupted, the mean deterioration rate of every transformation at every intensity is found. Afterwards, the median of all the different intensities of one transformation is calculated. The median is chosen, so that outliers don't skew the robustness measurements. If the mean was chosen, an extreme accuracy value like 0 could completely change the robustness result. This could be the case in an image that was already dark to begin with, and got its brightness lowered even further, resulting in a black image. This should not result in a far worse robustness rating, if the accuracy in the previous sampling steps was far better, because even the most robust image segmentation algorithm can not segment an image that is completely black. The evaluation results in a robustness graph and numerous statistical metrics for every transformation. The standard deviation above the robustness graph is the most relevant metric showing how robust the model is. A high standard deviation means that the model is better at working with some transformed images than others, which in turn means it is not generalizable and thus less robust. To provide even more information for experienced users, in the future the framework could be extended to let the user change the default graph, which shows the mean score for every transformation at every intensity, to display a variety of different statistical measures. These could include the mean, the median, the mode, and the minimum and maximum of the accuracy values. The implementation section will go into more detail on how they are calculated.

3.3 Limitations and Assumptions

The most important limitation of the framework is the time it takes to analyze an algorithm. While the processing of the test images has been optimized and is performed in a relatively short time period, the prediction times take much longer. This heavily depends on the speed of the model to be evaluated. The rate for the U-Net for analyzing a single image of 240x256 on a host with dual AMD CPUs and two GTX1080ti graphic cards is 2.87s (Qin et al., 2022). When extrapolating this time for a single image to a typical test scenario of 1020 images (20 test images, 5 transformations, 10 sampling steps), then prediction phase alone takes 48.79 minutes on average. Furthermore, it is important to note that an image size of 240x256 is roughly 12% of the size of the images used while testing the framework of 960x540. This means that an evaluation of a segmentation algorithm with test images of a typical size, can take a very long time. Another limitation is the heavy data dependence. The framework does not provide a data set out of the box, because the test images heave to be from the same domain as the model to be evaluated. This means that the user needs to provide the test images and corresponding references. If the test images provided are of poor quality or not relevant to the specific domain, the robustness ranking has less relevance, because it does not represent a typical data set, the algorithm actually segments in the real world. Furthermore, the interference effects are artificially introduced, which means that they may not exactly match the naturally occurring effects in real world images. And finally, the last limitation is the lack of external validation. Because only the robustness metrics of the different transformations are cross-validated, the model's robustness ranking may not be applicable on an entirely independent data set, if the testing data set was too small or too homogenous.

There are several assumptions this framework makes. It assumes that the data it is working with is **accurate and relevant** to the domain, as well as having **correct corresponding references**. Furthermore, the tool assumes that the algorithm to be evaluated **can be evaluated** using one of the **existing metrics**. The same is true for assuming that the test images uploaded are **relevant to the specific domain**. Lastly, the tool assumes that the robustness of image segmentation algorithms can be benchmarked by introducing **artificial interference effects** to the test images.

4 Implementation

4.1 Architecture

The code for the RobustnessTool is open source and can be found **here**. The experimental framework for quantifying the robustness of image segmentation algorithms has been built using the PyCharm Integrated Development Environment, because it is an all-in-one tool to build a Python backend and Vue.js frontend. Furthermore, the IDE has built in tools to help with refactoring, code assistance and debugging (Rahmonov, 2022). The Python backend uses the Flask web framework for handling the requests from the Frontend. The frontend uses AJAX to make these requests. In Table 2 a short overview of the available endpoints is presented.

Table 2: Available Endpoints

URL	Method	Description
/api/add-docker-container	PUT	Add new model for testing
/api/delete-docker-container	DELETE	Delete model & results
/api/get-docker-containers	GET	Get all available models
/api/set-ground-truth	PUT	Upload references
/api/set-test-images	PUT	Upload test images
/api/load-container-results	POST	Load the model's test results
/api/transform-images	POST	Applies selected interference effects to the test images
/api/build-docker	POST	Builds the container holding the model to be evaluated
/api/image-exists	POST	Checks if selected model is already present on server
/api/run-tests	POST	Runs model on transformed test data
/api/evaluate-results	POST	Stores and displays the test results and relevant metrics
/api/available-transformations	GET	Interference effects this version of the framework supports

This application architecture is able to operate in two distinct modes: a server-client mode and a backend-only mode. In the server-client mode, the Python backend and Vue.js frontend are hosted on a remote server. This enables the users to access the framework as a service without any installation requirements on their local machines. This makes the process lightweight and user-friendly. The second option is the backend-only mode, which allows users to run both the backend and frontend components on their local device. This has the advantage of eliminating the need for uploading large data files, since the application runs on the same machines where the data is stored. This can make a notable difference when testing large models, as users can avoid uploading .tar files of the models spanning multiple gigabytes. Because image segmentation models are often very large, the backend-only architecture has been chosen for the framework. The application uses the Model View Controller architecture, because it works well with web applications written in JavaScript (GeeksforGeeks, 2023) and provides a separation of concerns, which makes it straightforward to manage. Furthermore, the framework is built with modularity in mind, making it easier to extend its functionality and do further research on. In Figure 2, an overview of the most important components used, is shown.

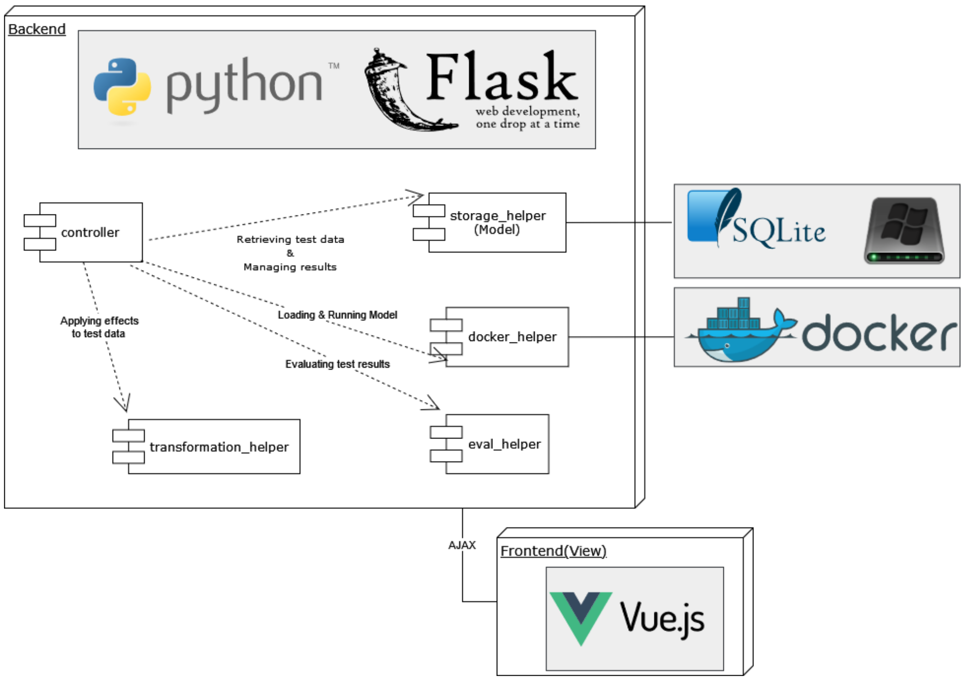

Figure 2: A component diagram showing the structure of the backend of the RobustnessFramework and the components it communicates with. The controller manages all communication of the frontend to the components of the backend. The storage_helper manages all interactions with the file system and the database. The docker_helper handles the communication with the models via the docker API. The transformation_helper applies the image transformations and the eval_helper evaluates the model's accuracy using the Dice Similarity Coefficient. (Wikipedia contributors, 2023f)(Wikipedia contributors, 2023e) (Wikipedia contributors, 2023b) (Wikipedia contributors, 2023a)(VectorLogoZone, n.d.)

The storage_helper class has access to the file system, where the test data is stored. Additionally, it also manages a SQLite Database, which tracks the existing containers and their results. SQLite has been chosen as the Architecture for the Database, because it is small, fast and requires no setup or configuration (SQLite, n.d.). Because the database only contains timestamps of the test runs and references to the model and result locations, it needs to be lightweight with little overhead. The frontend has been kept simple, using Vue.js because it is easy to learn, lightweight and very flexible (Shapel, 2023). The available transformations provided by the backend and displayed by the frontend, can easily be extended by adding the new transformation effect to the list of supported transformations, located in the file system of the application and then modifying the transformation_helper class accordingly.

4.2 Image Transformations

This section will go into detail on how the various interference effects are implemented.

Glare

The glare transformation aims to simulate the reflection of light on a metallic instrument. This could be the case in endoscopic imaging, where the light source can reflect on the metallic tools like a biopsy forceps, which is among other procedures used to diagnose gastric cancer (Matsuo et al., 2018). This effect is crucial for creating realistic visual conditions similar to those encountered during endoscopic procedures.
Algorithm 1 receives two parameters, the image_array, which is the test image to be converted represented as a NumPy array, and the flare_intensity, which is the strength of the glare effect. To apply this effect, the Pygame library is used. First, the test image is converted to a Pillow image and the library is initialized. Afterwards, a surface, which is a Pygame object for representing images (Pygame, n.d.), is created from the test image. Then two different PNG files are loaded (Figure 3), which are the two separate parts of the interference effect.

Algorithm 1 Pseudocode of the glare transformation in Python

```
1: function ADD_CUSTOM_LENS_FLARE(image_array, flare_intensity)
2:     Initialize Pygame library
3:     Create Pygame surface from the PIL Image
4:     Load flare overlays from assets folder
5:     Resize the flare overlays to match the image dimensions
6:     Adjust brightness of flare overlays according to flare_intensity
7:     Create new Pygame surface for the modified image
8:     Copy the original image onto the new surface
9:     Add first flare effect to the new surface
10:    Add second flare effect to the new surface
11:    Convert the modified surface back to a NumPy array
12:    return modified NumPy array
13: end function
```

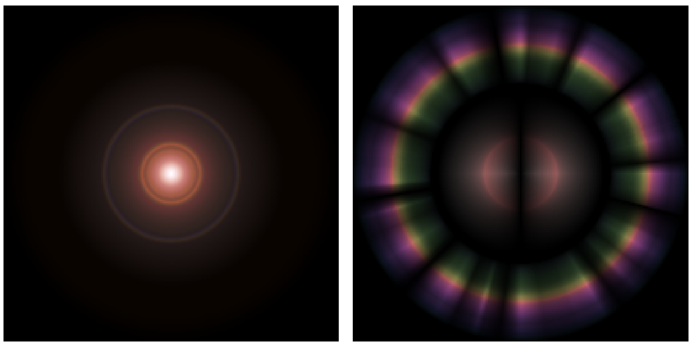

Figure 3: Component-Assets of the glare effect. Base glare (left) and outer glare(right)

These glare images are resized to align with the dimensions of the original test image. The base glare takes the full size, while the second is scaled down to 40% of the original dimension, in order to serve as a secondary reflection. Afterwards, the parameter “flare_intensity” determines the brightness of the loaded assets, by scaling their pixel values accordingly. Finally, the glare images are blended onto a new surface, holding the modified image. The BLEND_RGB_ADD flag is set during this process, to use additive RGB Blending. In additive blending, the pixel values of the test image and the glare images are added at each pixel where the glare effect is placed (Processing.org, n.d.). This results in the brightness of the underlying image increasing at those pixels, simulating the effect of light glare. At the end, the modified Pygame surface is converted back into a NumPy array and is returned. Figure 4 shows the minimum and maximum intensity of the glare effect. The code is inspired by **this** GitHub repository, which is under the MIT license.

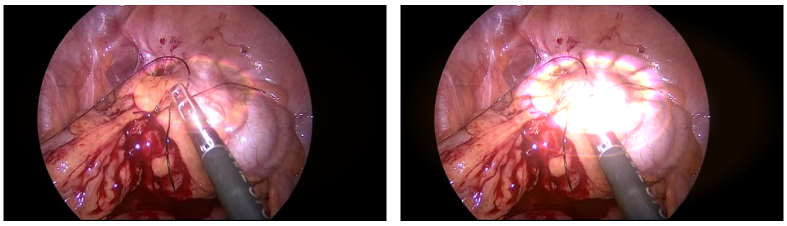

Figure 4: Glare effect with flare_intensity=0.5 (left) and glare effect with flare_intensity=2 (right)

Brightness and Darkness

The objective of the lighting modulation effects is to test the robustness of the algorithm against variations in lighting conditions. In endoscopic imaging, varying brightness levels can occur due to changes in the depth of the scope insertion or the orientation of the endoscope's light source. Furthermore, different tissue types can absorb light differently (Sato, 2021). Algorithm 2 takes two arguments, the original test image and an intensity factor, for adjusting the brightness.
It uses the adjust_gamma_of_image function from the exposure module of the Skimage library. It adjusts the gamma of the image based on the given factor. "Gamma correction is, in the simplest cases, defined by the following power-law expression (6):

$$V_{\text{out}} = V_{\text{in}}^{\gamma} \tag{6}$$

(Wikipedia contributors, 2023c)

Because the gamma correction is a non-linear operation, the effects "brightness" and "darkness" use the same adjust_image_brightness function, with different value ranges. Values between 0 and 1 will increase image brightness, while values >1 will decrease it. It is worth noting that reasonable gamma values can vary between test data sets. These values have been empirically found to be appropriate in most cases, but in cases of extremely dark or bright original test images, it can make sense to adjust the value ranges of these two interference effects accordingly.

Figure 5 shows the minimum and maximum intensities of the brightness effect, while Figure 6 shows the minimum and maximum of the darkness effect.

Algorithm 2 Pseudocode of the brightness adjustment transformations in Python

```
1: function ADJUST_IMAGE_BRIGHTNESS(image, factor)
2:     return ADJUST_GAMMA_OF_IMAGE(image, gamma=factor)
3: end function
```

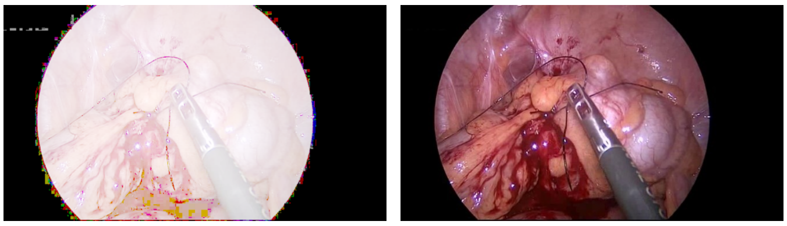

Figure 5: Brightness effect with factor=0.1 (left) and brightness effect with factor=0.9 (right)

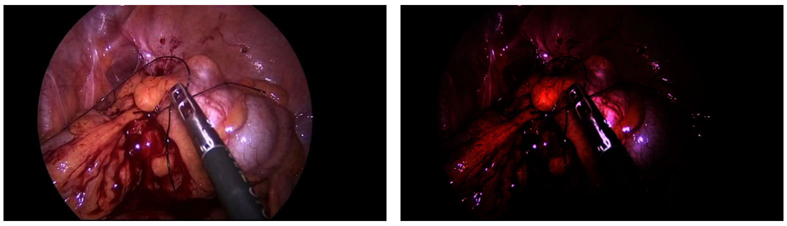

Figure 6: Darkness effect with factor=1.5 (left) and darkness effect with factor=5 (right)

Motion Blur

The objective of this effect is to simulate fast movements of the camera during imaging procedures like endoscopy. During such procedures, the camera might move quickly, either because of the actions of the medical practitioner or due to patient movements. These fast movements can result in blurring of the captured image (Yang et al., 2020).

The apply_motion_blur function (Algorithm 3) takes two parameters, the original test image and the radius, which defines the extent of the blur. The function generates a kernel of zeros, using NumPy, with the dimensions being represented by the radius parameter squared. The middle of this kernel is filled with ones. Then, this kernel is normalized by dividing each element by the radius. Finally, the kernel is applied to the test image by the filter2D method of OpenCV, which blurs the image in the direction of the row of ones in the kernel. By default, the motion blur is applied from left to right

(Figure 7).

Algorithm 3 Pseudocode of the motion blur transformation in Python

```
function APPLY_MOTION_BLUR(image, radius)
    if radius $\leq 0$ then
        Print "Radius must be greater than zero."
        return image
    end if
    Generate motion blur kernel of size (radius x radius)
    Normalize the kernel
    Apply the kernel to the image to get the blurred image
    return the blurred image
end function
```

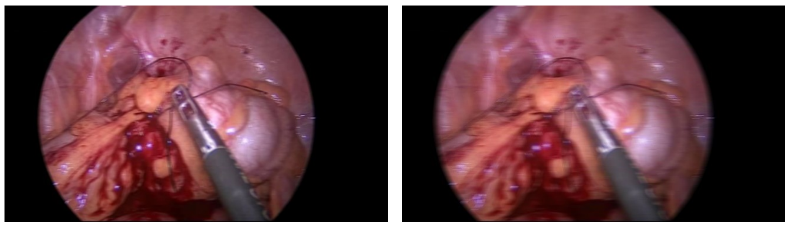
Figure 7: Motion blur effect with radius=10 (left) and radius=20 (right)

Noise

The objective of introducing noise as an interference effect is to simulate random noise commonly observed in medical imaging. Noise can be introduced due to device's limitations, transmission interference or low lighting conditions (Goyal, 2018).

The add_noise function (Algorithm 4) takes the original test image and an intensity factor as parameters. The intensity factor, in the context of noise, is the level of noise variance. It uses the random_noise utility from the Skimage library to generate Gaussian noise with the desired variance. The generated noise is then added to the image, creating a noisy version (Figure 8).

Algorithm 4 Pseudocode of the noise transformation in Python

```
1: function ADD_NOISE(image, factor)
2:     Generate random Gaussian noise with variance ‘factor‘
3:     Add the noise to the original image
4:     return the noisy image
5: end function
```

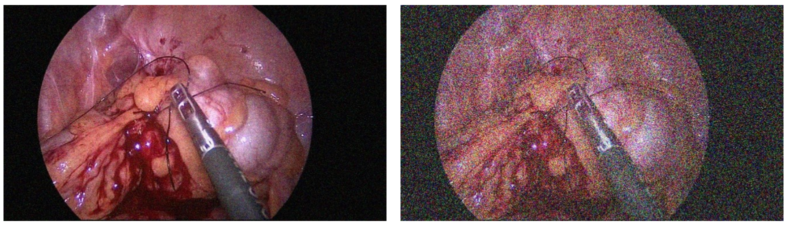
Figure 8: Noise effect with variance=0.01(left) and noise effect with variance=0.2(right)

Contrast

The objective of applying changes to the image contrast is to simulate different methods of imaging. Especially in endoscopic imaging, different techniques produce varying levels of contrast (He et al., 2021). A robust image segmentation algorithm needs to be able to produce reliable results in a realistic range of contrast augmentations.
The adjust_contrast function (Algorithm 5) modifies the contrast of an image using the gamma parameter with the exposure module of the Skimage library, which is similar to the augmentation of the image brightness, with the difference being the gamma parameter, which is set to 1/factor. The resulting images can be seen in Figure 9.

Algorithm 5 Pseudocode of the contrast transformation in Python

```
1: function ADJUST_CONTRAST(image, factor)
2:     Adjust the image contrast using gamma correction, with gamma set
       to 1/factor
3:     return the contrast-adjusted image
4: end function
```

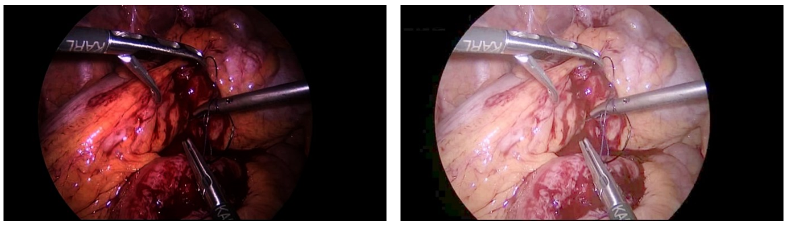

Figure 9: Contrast decreased with factor= 0.5 (left) and contrast increased with factor=2 (right)

Sharpness

The objective of applying a sharpness transformation is to simulate the effect of overly-sharp images that can occur in imaging. This can happen due to hardware variations, but especially by software post-processing or digital zoom features, which artificially sharpen the image (Wikipedia contributors, 2023d). This increase in sharpness could make it difficult to differentiate between actual anatomical details and the introduced artifacts.
Algorithm 6 takes the original image and an intensity factor as parameters. It uses the filters module of Skimage, by employing the unsharp_mask function, which sharpens the image with a desired radius of the area and a sharpness multiplier. The radius is by default set to one, and the sharpness multiplier is dependent on the intensity factor. The effects of the minimum and maximum edge reinforcement multiplier are shown in Figure 10.

Algorithm 6 Pseudocode of the sharpness transformation in Python

```
1: function ADJUST_IMAGESHARPNESS(image, factor)
2:     Apply unsharp mask to the image with radius = 1.0 and strength = factor
3:     return the sharpened image
4: end function
```

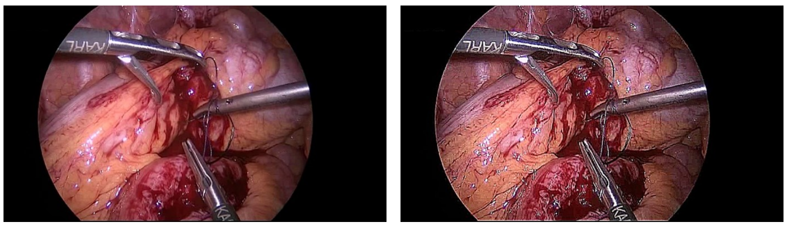

Figure 10: Sharpness effect with factor= 1(left) and sharpness effect with factor=10 (right)

Smoke

The objective of this interference effect is to simulate the smoke produced by coagulation tools during minimally invasive procedures like endoscopy. Algorithm 7 takes the original test image and an intensity factor as parameters. First, it loads the stored smoke texture asset (Figure 11). Next, the smoke texture is resized to match the dimensions of the input image and both textures are converted to NumPy arrays for further manipulation. Afterwards, a random transparency mask for the smoke texture is generated and the smoke effect is blended into the original image using this mask and the factor parameter. The transparency of the smoke asset can be modulated from 0.2 to 1 (Figure 12).

Algorithm 7 Pseudocode of the smoke transformation in Python

```
1: function ADD_SMOKE(image, factor)
2:     Load the smoke texture and resize it to match the input image dimensions
3:     Convert both images to NumPy arrays
4:     Generate a smoke mask with random transparency values for the smoke texture
5:     Normalize the smoke mask
6:     Blend the smoke texture into the original image using the mask and alpha factor
7:     return the modified image
8: end function
```

Figure 11: The smoke texture asset used to create the interference effect.

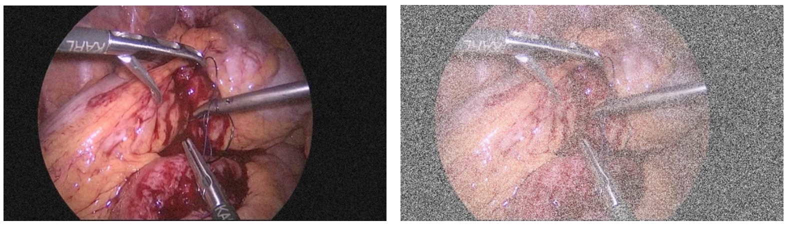
Figure 12: Smoke effect with transparency = 0.2 (left) and smoke effect with transparency = 1 (right)

Resolution

The objective of lowering the resolution is to simulate different equipment used during the imaging process. Another reason could be deliberately lowered data transmission rates in telemedicine scenarios. For example, during tele-endoscopy, real-time transmission of images is critical, but bandwidth might be limited. This function, as shown in Algorithm 8, modifies the test image by reducing its resolution based on the factor parameter. The new dimensions of the image are calculated by dividing the original dimensions by the intensity factor. The image is then resized using bilinear interpolation, provided by the resize function of the PIL library. Afterwards, the image is resized back to its original size, leading to information loss, or a lower resolution, during the resizing process. This can be observed in Figure 13.

Algorithm 8 Pseudocode of the resolution transformation in Python

```
1: function LOWER_RESOLUTION(image, factor)
2:     Calculate new dimensions based on resolution factor
3:     Resize the image using bilinear interpolation
4:     Restore the original dimensions
5:     Convert the image back to a NumPy array
6:     return the modified image
7: end function
```

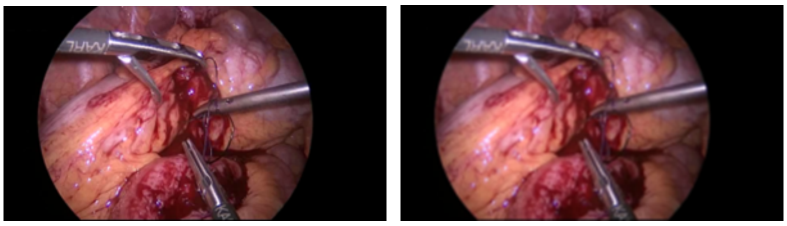
Figure 13: Resolution decreased with divisor =1.5 (left) and resolution decreased with divisor = 5 (right)

Vignette

The objective of this interference effect is to simulate the darkening of the corners or edges of the test image. This can be the case in endoscopic procedures with inhomogeneous lighting conditions. The function add_vignette (Algorithm 9) uses the intensity parameter to generate a black ellipse on a transparent mask. The area inside this ellipse remains transparent, forming a "window" onto the test image. The image and mask are both converted to RGBA format to ensure they have alpha channels. These alpha channels are crucial as they allow for the compositing of the image and the mask, thereby creating the vignette effect. The mask, initially created as a grayscale image, is used to control the level of darkness applied to the test image. The darker areas of the mask will produce the vignette effect, darkening the corners of the test image, while leaving the center visible.
This approach ensures that the periphery of the image darkens as the intensity of the vignette effect increases, simulating the effect of inhomogeneous lighting conditions often seen in endoscopy. This can be observed in Figure 14.

Algorithm 9 Pseudocode of the vignette transformation in Python

```
1: function ADD_VIGNETTE(image, intensity)
2:     Create a mask with dimensions same as the image
3:     Calculate the ellipse parameters based on intensity
4:     Draw a black ellipse onto the mask, leaving a transparent window in
       the center
5:     Convert image and mask to RGBA format
6:     Composite the image and the mask to darken the image corners,
       leaving the center as is
7:     return the modified image with vignette effect
8: end function
```

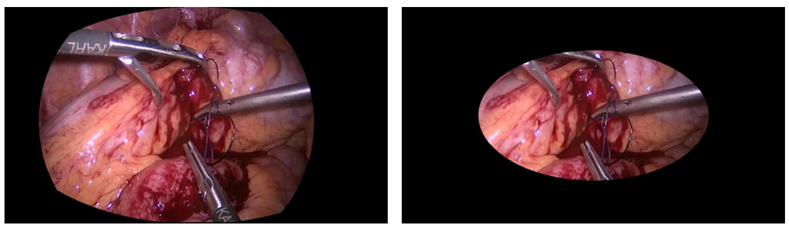

Figure 14: Vignette with size= 0.05 (left) and vignette with size= 0.4 (right)

4.3 Docker

The model to be evaluated should be contained inside a docker container. The specific requirements are based on the Submission Instructions of the ROBUST-MIS 2019 Challenge (DKFZ et al., 2019b). To set up an image segmentation model, which uses the GPU, inside a docker container, the nvidia-docker needs to be installed. This provides the necessary nvidia runtime, to run a given model. Instructions on how to set this up can be found **here**. It is worth noting that this only works on Linux Operating Systems. The framework will use the docker API to be able to test any image segmentation model, regardless of specific implementation. To achieve this, the framework executes this command when using the model:

```
sudo docker run --gpus 1 --runtime nvidia --ipc=host -v
    "INPUT_DIR:/input" -v "OUTPUT_DIR:/output" IMAGE_NAME
    /usr/local/bin/run_network.sh
```

This will mount an input folder, containing the test data and the corresponding references. Furthermore, it mounts an output folder, where the predictions made by the model will be stored. Then the "run_network.sh" script is called. This script has to be included in the provided docker container, and must be written in such a way that it will run the specific model automatically. This enables the framework to use the same docker command, to evaluate any model. It is important to set reasonable model parameters, such as batch size, in order for the framework to be able to evaluate the model on the given hardware.

4.4 Data Flow

In this chapter, the data flow within the application will be described, to give a general overview of the data processing pipeline. After starting the application, the user can create a new Docker container with a custom name. Then, the image segmentation model needs to be provided. Because the application runs in the backend_only mode, a file choose window will appear, triggered from the Python backend. This means that the user can specify the path to the model, without needing to upload huge amounts of data. The application then registers the container and the path to the model, as well as other meta information, in the SQLite Database. This marks the end of the registration process. Afterwards, the application checks if there are already results presents for the registered container and displays them if applicable. If no results are stored in the database, the user can upload the test images and their corresponding references with two separate file choose windows launched from the backend. Both data sets need to be in a separate .tar folder. The naming conventions for test images and corresponding references should adhere to specific naming formats.

- **Test Images**

```
raw.png/jpg, raw1.png/jpg, raw2.png/jpg, ..., rawX.png/jpg
```

- **Corresponding References**

```
solution.png/jpg, solution1.png/jpg, solution2.png/jpg,
    ..., solutionX.png/jpg
```

The application then creates a testing environment for the container, by creating folders where the test data and references will be stored. After providing the two different tar folders, the user can select the desired interference effects to test the model with. The frontend requests the available transformations from the backend. These can easily be extended by implementing the desired interference effect in the transformation_helper class and adding the transformation label to the transformations.txt file. The combination of interference effects depend on the actual occurrence of these effects in the real-world scenario of the given model. Furthermore, the user can specify the sampling steps for every transformation. A higher value for the sampling steps means that the application will divide the value range for the given interference effect into more data points. For example, the maximum sampling rate of 10 for the interference effect sharpness, which has a value range of 1-10, means that the application will create 10 variations of the original test images with increasing sharpness. The first variation would have a sharpness factor of 1, while the last would have a sharpness factor of 10. The actual intensity values applied are calculated by using the intensity factor to linspace the value range. The linspace function of the NumPy library divides the value range in equal parts, depending on number of sampling steps. It is worth noting that a higher amount of transformations and sampling steps will increase the time necessary to complete the evaluation. After specifying the desired parameters, the "Run Tests" button will be clickable. This will first of all transfer the information on the amount of transformations and sampling steps to the backend, which will use this to apply the desired effects using the implementation discussed in previous chapters. The framework will store these images in the "transformations" folder inside the testing environment created, when adding a new container. For every transformation and sampling step, a new folder is created, holding all the original test images with the specific transformation applied. After successfully transforming the test data, the application extracts the name of the model to be evaluated from the manifest.JSON file, located in the .tar file containing the model. Then, it checks if the model is already present on the system. If it is not, the "docker load input TAR_PATH" command is used to make the docker container usable. After successfully completing this operation, the actual testing phase begins. This starts by creating the rest of the testing environment, namely the testing folder, which will be mounted as the input folder. Similarly, the output folder is created. All the transformed test images will be copied in the correct format to the testing folder. Subsequently, the docker command discussed previously will be used to start the docker container and specify the directories above. The command is launched in a new shell and it is assumed that the environment where the application is running has the nvidia runtime installed. Once this process is finished, the model predictions will be located in the output folder, and the evaluation phase will begin. Here, every model prediction will be compared to the corresponding reference using the evaluation metric. In the context of the ROBUST-MIS model, the Dice Similarity Coefficient is used. This provides an accuracy rating for every test image at every sampling step for every transformation. This data is stored in a 3D Array. For every sampling step of a transformation, statistical metrics are calculated. These metrics are the mean, median, standard deviation, variance, minimum and the maximum. Additionally, for every transformation, the mean of every statistical metric is calculated over all the sampling steps. Finally, the 3D results, the labels for the transformations applied, and the calculated metrics are stored inside a JSON file. The JSON file is located inside the "results" folder in the testing environment and is additionally registered in the results table of the SQLite Database. When the evaluation phase has finished successfully, the frontend requests the results for the current container from the backend. This is provided by loading the newest JSON result registered in the database and returning it to the frontend. The frontend will display a graph for every transformation, where the x axis represents the number of sampling steps, and the y axis represents the mean of accuracy of every test image at that sampling step. Above this graph, the mean of every statistical metric for all the sampling steps is displayed for a deeper insight of the robustness regarding this interference effect.

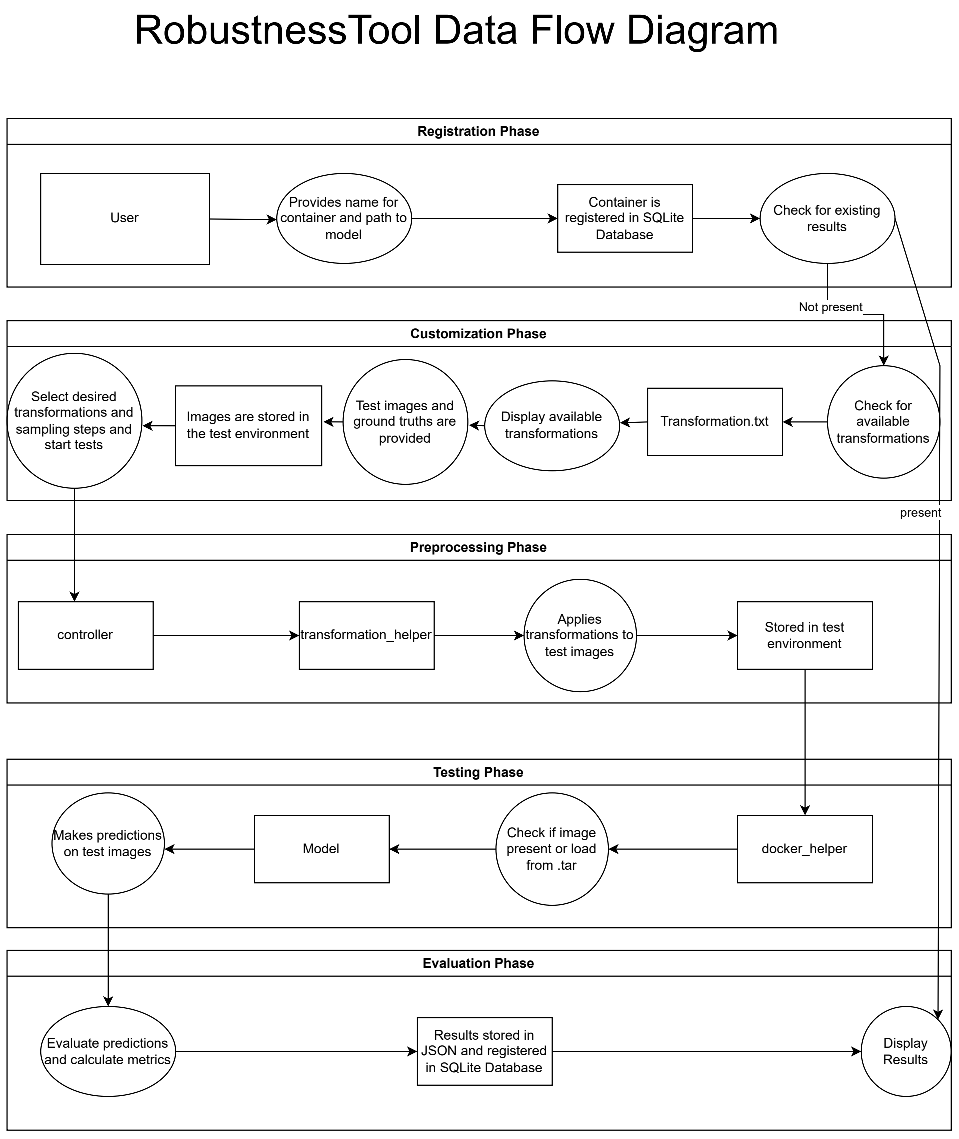
RobustnessTool Data Flow Diagram
Registration Phase
User
Provides name for container and path to model
Container is registered in SQLite Database
Check for existing results
Not present
present
Customization Phase
Check for available transformations
Transformation.txt
Display available transformations
Test images and ground truths are provided
Images are stored in the test environment
Select desired transformations and sampling steps and start tests
Preprocessing Phase
controller
transformation_helper
Applies transformations to test images
Stored in test environment
Testing Phase
docker_helper
Check if image present or load from .tar
Model
Makes predictions on test images
Evaluation Phase
Evaluate predictions and calculate metrics
Results stored in JSON and registered in SQLite Database
Display Results

Figure 15: Data Flow Diagram of the RobustnessTool Framework. The Diagram shows the main phases of the application: Registration, Customization, Preprocessing, Testing and Evaluation.

5 Results

In this chapter, the evaluation results for the quantification of robustness for the ROBUST-MIS image segmentation model in the context of medical endoscopic imaging are presented. The model was evaluated regarding its robustness against various types of interference effects, namely: noise, glare, brightness, darkness, contrast, smoke, vignette, motion blur, and resolution. Using a set of 20 endoscopic test images containing medical instruments, extracted from the ROBUST-MIS data set (DKFZ et al., 2019a), the model's robustness was examined. The relationship between the sampling steps and the applied intensity value, which is specific to the interference effect, can be observed in Table 1.

5.1 Overview

The model was generally very robust against the chosen interference effects. The best result was achieved while withstanding the sharpness effect, as can be seen in the sharpness section, with just a 0.12 total drop of mean accuracy over the 10 sampling steps. This results in a very low standard deviation of 0.09, which is the most relevant metric when evaluating robustness. As can be seen in the darkness section, the model was least robust regarding the darkness effect, where a total drop of 0.45 in mean accuracy, resulting in a standard deviation of 0.33, was observed.

It is important to understand that the metrics on top of the transformation graph are the mean over every sampling step. This means that for example the metric “Minimum” does not show the minimum value of the graph displayed, but the mean of every minimum of every sampling step.

5.2 Transformations

Noise

The model shows considerable resilience to noise interference, maintaining a variance of 0.03 and a standard deviation of 0.15, as indicated in Figure 16. It retains high predictive accuracy, while only slightly decreasing by 0.03 between the first and third sampling step, representing noise variance values of 0.01 and 0.05 respectively. The mean accuracy continues to remain relatively stable until the degradation of the signal-to-noise ratio reaches a threshold that obscures key features of the image (Figure 8). Specifically, this threshold becomes evident during the final sampling step, where the mean accuracy declines sharply from 82.2% to 64.8%. This substantial decrease in performance occurs within a narrow range of Gaussian noise variance, specifically between 0.179 and 0.2. Though the variations in noise levels between these specific sampling steps are visually indistinguishable, rising noise levels progressively modulate pixel values. This ultimately leads to a critical point where the model can no longer segment medical instruments with its prior accuracy. This critical point arises under conditions of significantly elevated Gaussian noise variance. An overview of all noise transformation sampling points reveals an average accuracy of 0.84 and a median accuracy of 0.93. Although this demonstrates the model's overall robustness in the presence of noise, it appears somewhat less resilient when subjected to other environmental factors like sharpness, glare, or contrast.

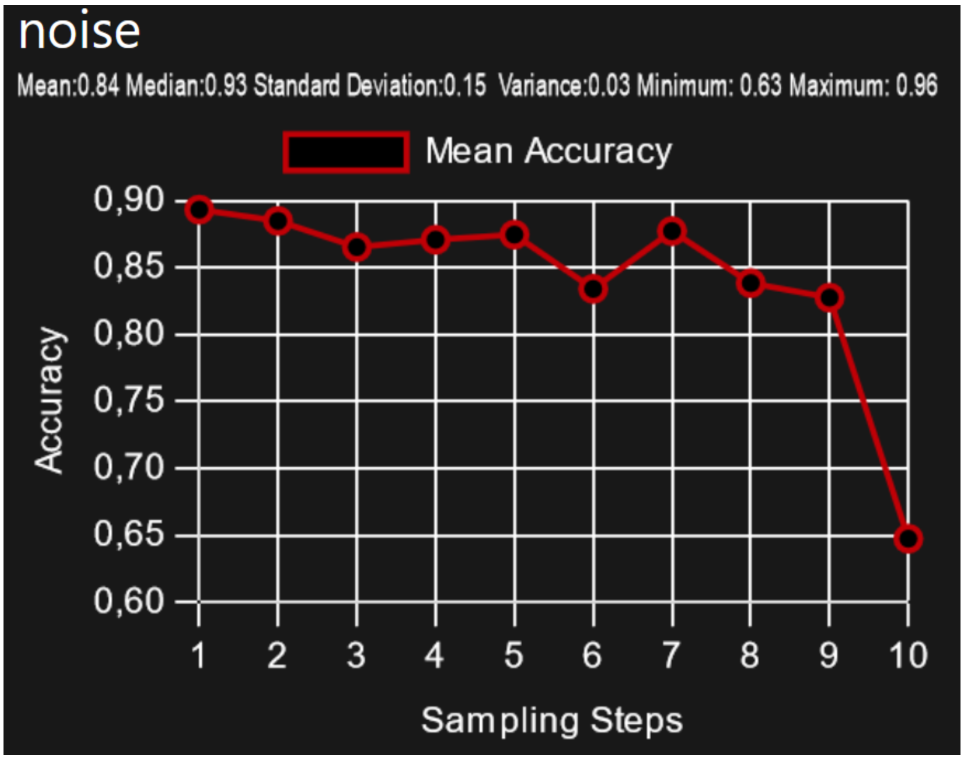

Figure 16: Evaluation results of the noise transformation. The sampling steps range from the variance of Gaussian noise of 0.01 to 0.2.

Contrast

In the test performed, the contrast of the test images is adjusted to a range of 0.5 to a maximum modulation value of 2. The result is an initially lower mean accuracy of 0.87 compared to other sampling points (see Figure 17). After an initially stable period, the accuracy only slightly continues to drop to a local minimum of 0.887, which is still significantly higher than the accuracy observed with a low contrast. This suggests the model is more sensitive to low contrast than to high contrast. Despite this, the model generally maintains strong resilience to fluctuations in image contrast, boasting a variance of 0.1 and a standard deviation of 0.11. It consistently delivers high accuracy across all sampling stages, achieving a noteworthy mean minimum accuracy of 0.74. On average, the model performs well when faced with varying levels of contrast, attaining a mean accuracy of 0.89 and a median of 0.96. Among all the types of interference examined, the model's resistance to contrast changes ranks as one of the most robust. This is also represented by the span of the mean maximum and the mean minimum being only 0.23.

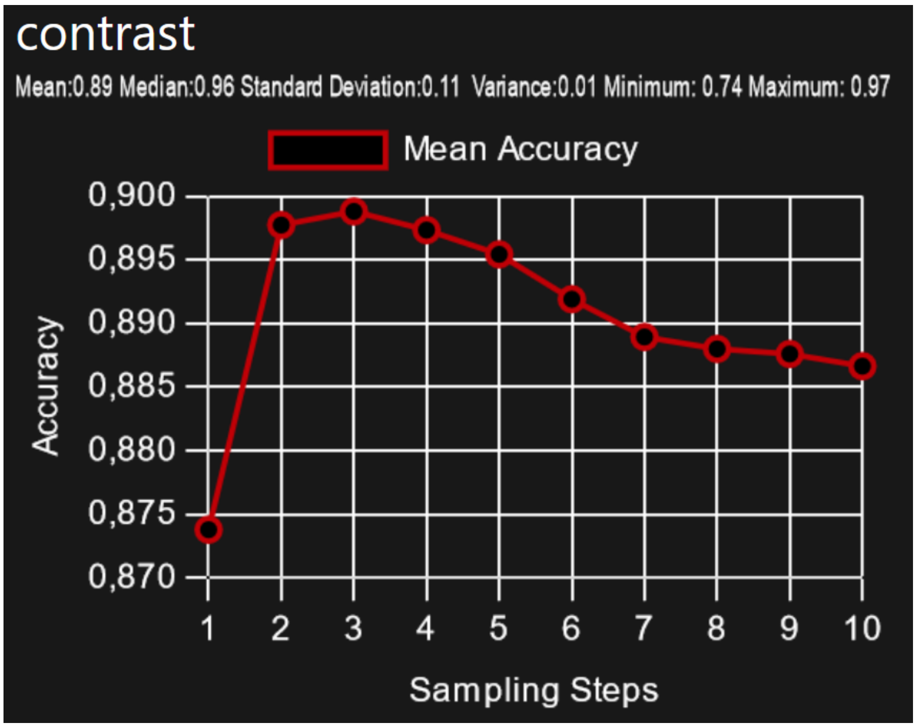

Figure 17: Evaluation results of the contrast transformation. The sampling steps range from a 1/gamma value of 0.5 to 2.

Brightness

When evaluating the impact of brightness interference, it is important to consider the nonlinear gamma correction function utilized. Specifically, the first sampling step significantly impacts image brightness, observable in Figure 18. In this figure, the initial sampling step exhibits the lowest mean accuracy, registering at 0.55, while subsequent images maintain a notably high degree of accuracy. After a substantial increase of 0.29 in accuracy, coinciding with a gamma value shift from 0.1 to 0.18, the rate of accuracy improvement slows. A modest 0.031 increase in accuracy is noted when the gamma value further rises to 0.27. These robustness results align with those observed for the noise interference effect, displaying a variance of 0.03 and a standard deviation of 0.14. The data suggests that the model is highly resilient to brightness modulation, up to a certain threshold gamma value where essential image features become obscured, as depicted in Figure 5. Beyond this point, pixel values often max out at 255, leading to loss of information. This is most important at the edges, where a loss of information can have extreme consequences on the model's ability to accurately segment the object of interest. Furthermore, high gamma values can change the texture and color of objects, making it more difficult to detect them. Otherwise, the model shows minimal difficulty in handling images affected by brightness interference.

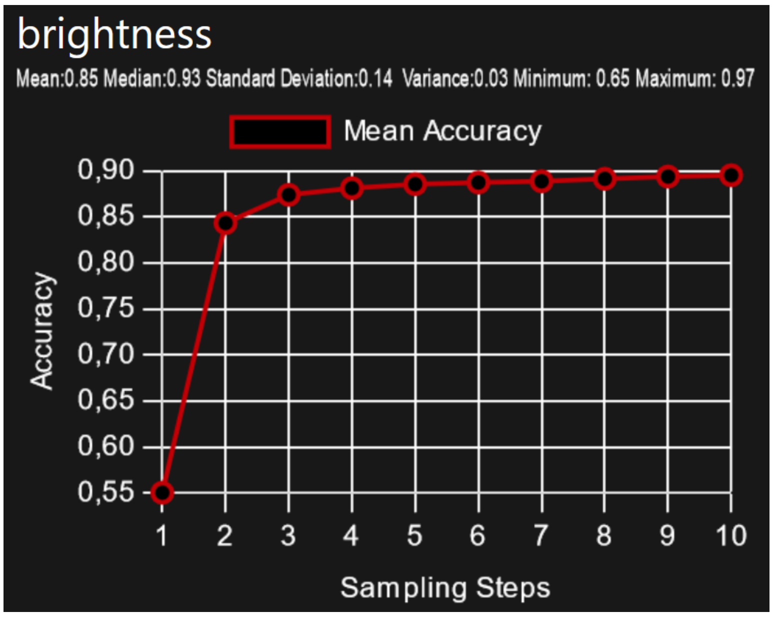

Figure 18: Evaluation results of the brightness transformation. The sampling steps range from a gamma correction value of 0.1 to 0.9.

Darkness

The model demonstrates reduced robustness when confronted with darkness interference, displaying a high variance of 0.12 and a standard deviation of 0.33, as highlighted in Figure 19. This manifests in a wide range of accuracies across different sampling steps, with the average minimum accuracy reaching as low as 0.18 and the average maximum soaring to 0.91, resulting in the largest observed accuracy span of 0.73. One plausible reason for this discrepancy is that darkness quickly removes crucial image information (Figure 6), thereby complicating the task of medical tool segmentation for the model. Interestingly, the initial gamma adjustment from 1.5 to 1.8 does not significantly impact the mean accuracy. The most significant drop in mean accuracy is recorded between the second and fourth sampling points, plummeting from 0.89 to 0.63. This downturn occurs in the gamma value range of 1.8 to 2.6. Despite these challenges, the model maintains a mean accuracy of 0.64 and a median accuracy of 0.83 across all sampling steps. It is worth noting that the accuracy levels midway through the sampling process are extremely stable. The increase in gamma value from 2.6 to 3.83 did not significantly impact the mean accuracy. However, in the final stretch leading up to the maximum gamma value of 5, the model's accuracy once again took a notable hit, dropping to its lowest mean accuracy level of 0.45.

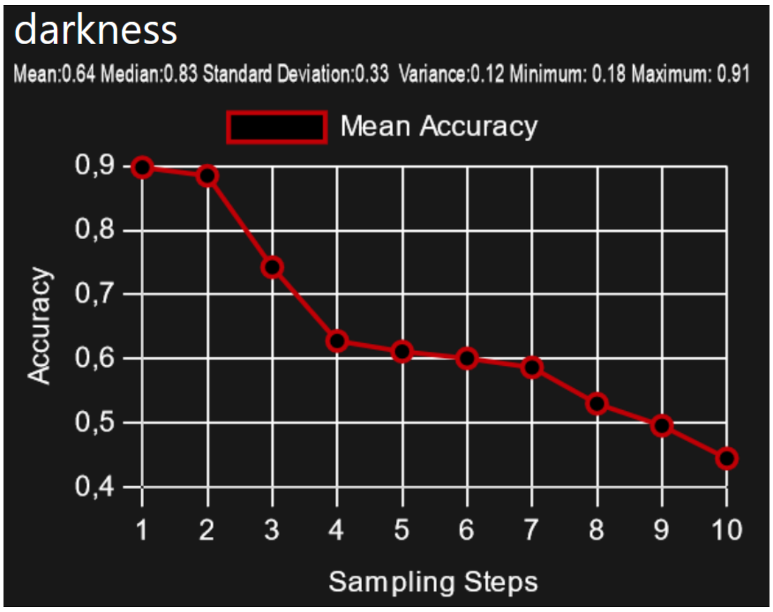

Figure 19: Evaluation results of the darkness transformation. The sampling steps range from a gamma correction value of 1.5 to 5.

Sharpness

The evaluated model is very robust regarding the sharpness interference effect, with a variance of 0.01 and a standard deviation of 0.09, as per Figure 20. Furthermore, the model produced accurate results, with the average minimum being 0.77. It is important to note that applying a weak sharpness effect to the image, actually increased the mean accuracy score. This could be the result of the sharpness effect reinforcing the edges (Figure 10) of the medical tools to be segmented. A sharper distinction between the object of interest and the rest of the pixels, makes it easier for the model to segment the transformed test image. The mean accuracy improved from 0.899 at an edge reinforcement multiplier value of 1, to a mean accuracy score of 0.902 at an edge reinforcement multiplier value of 4. Afterwards, the accuracy continued to drop to the lowest recorded mean accuracy of 0.887 at an edge reinforcement multiplier value of 10. This could be the result of the sharpness effect increasing the noise in the test image. This could create artificial boundaries, making it more difficult to discern the edges of the object of interest. Another issue of an overly extreme sharpness effect is edge saturation, where the pixel values of the reinforced edges can reach their maximum, leading to information loss.

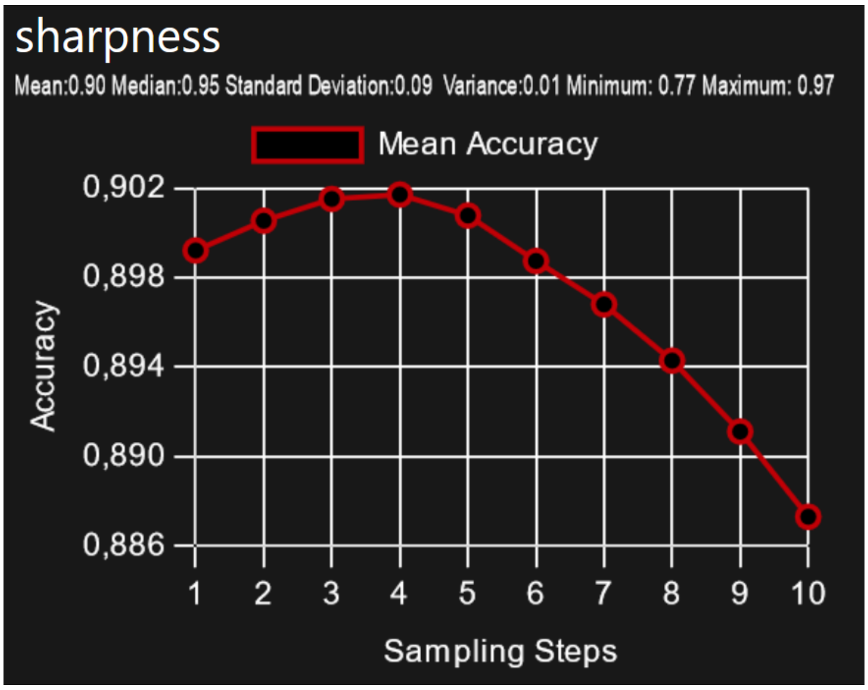

Figure 20: Evaluation results of the sharpness transformation. The sampling steps range from an edge reinforcement value of 1 to 10.

Smoke

The test outcomes, as illustrated in Figure 21, suggest that the model is vulnerable to the effects of smoke interference. Despite the seemingly low variance of 0.03 and standard deviation of 0.16, the evaluated algorithm lacks robustness regarding the smoke effect. The model's accuracy sees a steep decline at the third sampling step and subsequently stabilizes, but at a substantially reduced level of accuracy. The most significant drop in mean accuracy is observed between the transparency values of 0.29 and 0.38, plunging from 0.84 to 0.28. This accounts for a 0.56 decrease in mean accuracy within a single sampling step. Figure 22 provides a visual representation using one sample image from the set of 20 test images, captured at the second and third smoke transformation sampling steps. Following this decline, the mean accuracy plateaus, settling into a narrow range between 0.194 and 0.197, from the fourth to the seventh sampling steps. This corresponds to a smoke transparency range of roughly 0.46 to 0.73. Intriguingly, the mean accuracy starts to recover beyond this point, ultimately reaching an accuracy of 0.38 at the maximum transparency value of 1. This marks a 0.18 improvement in accuracy compared to the end of the plateau, despite the increase in intensity of the smoke interference effect.

As for the reasons behind this unexpected improvement in accuracy, further investigation is needed to identify or rule out potential factors contributing to this phenomenon.

While a medical practitioner will have no issues detecting the medical instruments even at higher smoke sampling steps, the model performs poorly. Analogous to its behavior with noise, the model appears to reach a critical threshold at the third smoke sampling step. At this juncture, a significant number of pixel values are altered to white smoke effect, impeding the model's ability to accurately perform automated segmentation.

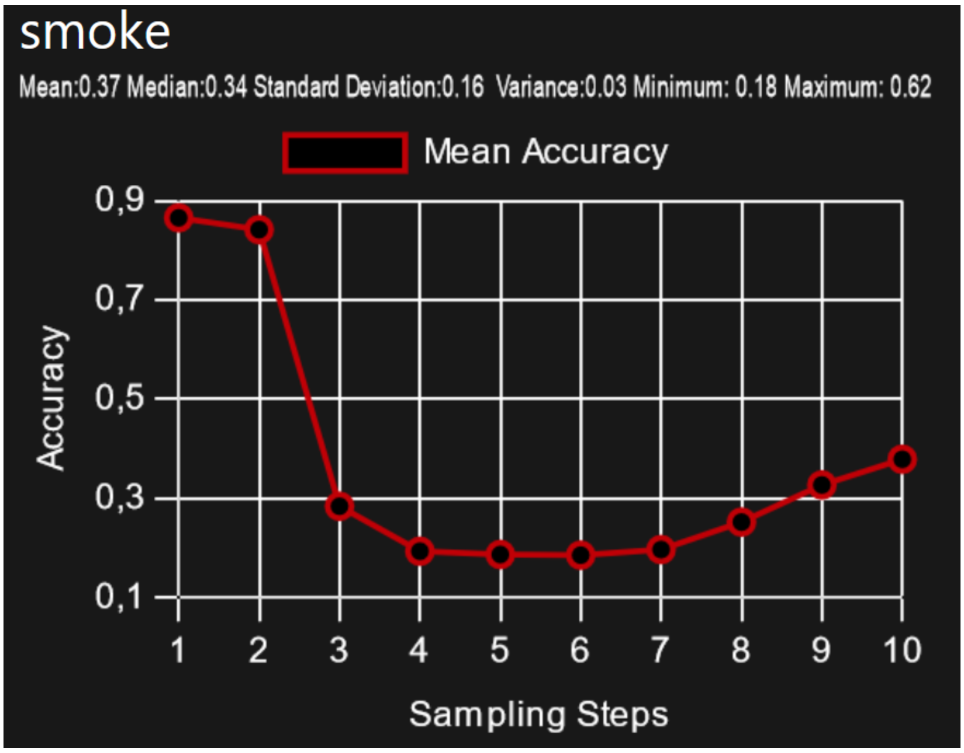

Figure 21: Evaluation results of the smoke transformation. The sampling steps range from a smoke transparency value of 0.2 to 1.

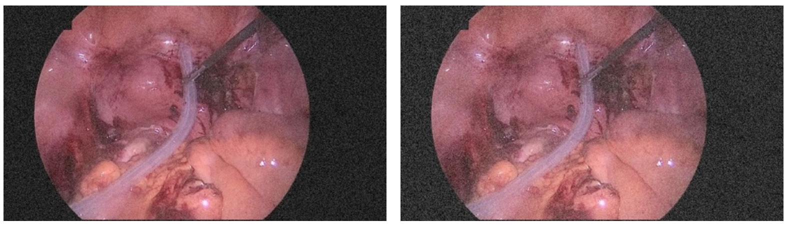

Figure 22: Smoke transformation at sampling step 2 (left) and sampling step 3 (right)

Glare

The model under evaluation demonstrates a high degree of resilience to glare interference, as substantiated by a low variance of 0.01, illustrated in Figure 23. The mean accuracy only slowly and minimally decreases with increasing levels of glare brightness. Only after the sixth sampling step does the accuracy begin to drop slightly faster, decreasing from 0.87 to 0.8 within 4 sampling steps. Beyond its robustness, the model also excels in terms of accuracy, with the mean minimum accuracy standing at 0.69. Although the range between the mean minimum and maximum accuracies is somewhat broader compared to what is observed under the sharpness effect, the model's performance remains consistently strong across all sampling points. It boasts a high mean accuracy of 0.86 and a median accuracy of 0.94. One plausible explanation for these exceptionally robust results could be that the glare effect occupies a relatively smaller portion of the image (Figure 4) compared to other forms of interference. This suggests that the model is more adept at handling disruptions that are confined to smaller regions within the frame, because less information is lost. A bigger glare effect has a higher chance of obscuring details like the edges of an object of interest. Future research could explore the impact of variable glare sizes and locations to assess whether the model maintains its robustness under more complex glare conditions. This additional investigation could provide further insights into the model's capabilities and limitations concerning glare interference.

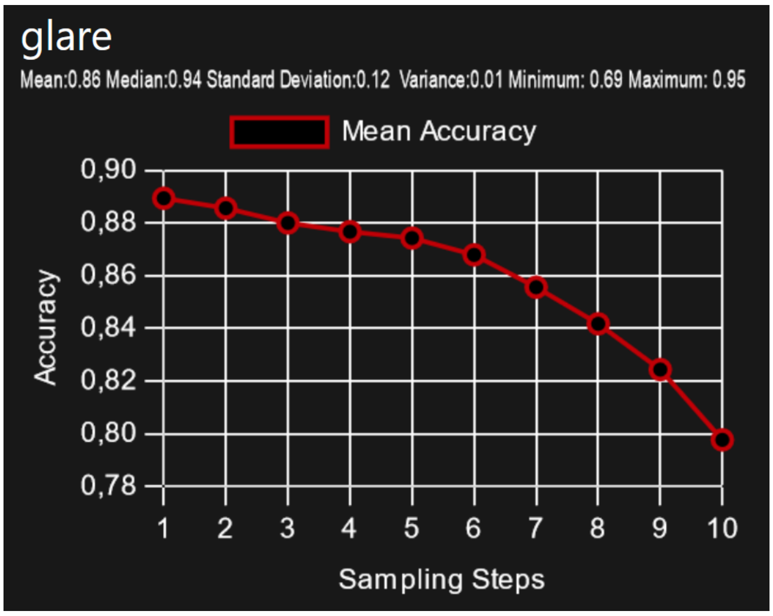

Figure 23: Evaluation results of the glare transformation. The sampling steps range from a brightness multiplier value of 0.5 to 2.

Resolution
As displayed in Figure 24, the model demonstrates considerable resilience to changes in resolution, maintaining a mean accuracy of 0.86 even when reduced to one-fifth of the original resolution. This represents only an insignificant decline of 0.04 in accuracy compared to the initial sampling point at a resolution divisor of 1.5. This results in low variance and standard deviation values, at 0.01 and 0.11 respectively. The model's robustness with respect to resolution is largely a function of the feature sizes it is designed to extract. In scenarios where the targeted features are small, reductions in resolution could severely compromise predictive accuracy. However, in the context of this model, which focuses on segmenting medical instruments that are relatively large in relation to the overall image, degrading (Figure 13) the resolution has only a minimal impact on the mean accuracy. Figure 24 further illustrates an almost linear decline in mean accuracy with increasing resolution divisor values, suggesting that a proportional loss of image information occurs as resolution decreases, without any abrupt shifts in accuracy.

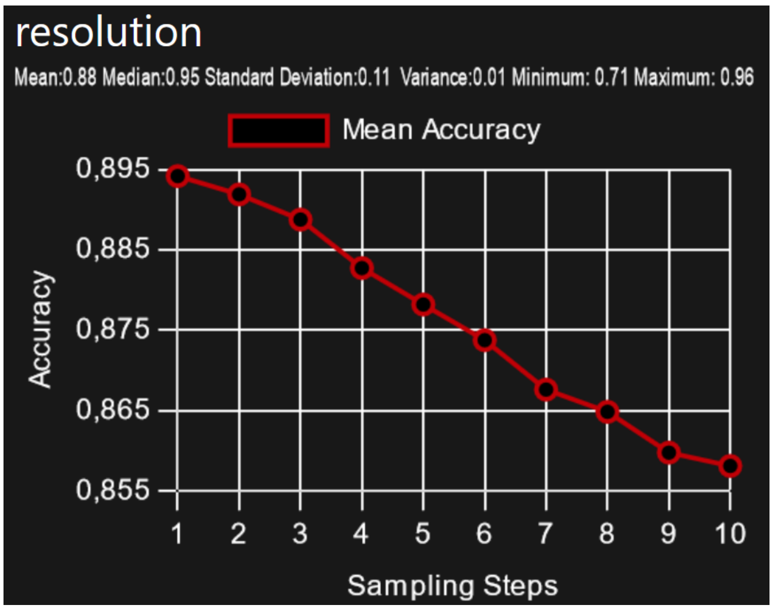

Figure 24: Evaluation results of the resolution transformation. The sampling steps range from a size divisor of 1.5 to 5.

Vignette and Motion Blur
Due to compatibility limitations, the evaluated model could not be assessed for its robustness against the interference effects vignette and motion blur. When running tests with the applied transformations, the docker container holding the evaluated model would become unresponsive without producing a clear error. One plausible explanation would be that the observed anomalies may be attributable to the occurrence of a division by zero, or a null value arising in some part of the computation. Another reason could be that the hardware used to run the model was insufficient for analyzing the applied effects. When conducting the evaluation with the vignette and motion blur effects applied, the memory of the GTX 970, used for testing, was fully utilized, indicating that the video memory of the graphics card was operating at its maximum capacity leading to a potential memory overflow. Further investigations employing alternative image segmentation models on more adequate hardware are required to determine the relevance of these effects to robustness assessments.

6 Discussion

6.1 Interpretation of Findings

In the course of evaluating the robustness of the image segmentation model, several important findings were made that have significant implications for its applicability in medical contexts. The model performed well when subjected to moderate to high levels of Gaussian noise. However, a critical drop-off point in mean accuracy suggests that it is not suitable for scenarios where exceptionally high levels of noise are expected, such as in low-light conditions or when using older endoscopic equipment that might introduce electronic noise. Contrast variations also yielded crucial insights. The model seems to be notably challenged under low-contrast conditions, a scenario where important anatomical details could be obscured, thus compromising diagnostic or surgical procedures. Conversely, high-contrast conditions appeared less problematic, underlining the model's specific domain of robustness. This means that the model should not be used when analyzing ultrasound images of soft tissue and the blood pool, as low contrast is to be expected (Yusefi & Helfield, 2022). Dim lighting, another aspect explored, proved to be a significant hurdle for the model's performance. This shortcoming particularly limits its usability in procedures involving closed body cavities or in any other minimally invasive operations that may present reduced lighting conditions. This also includes endoscopic imaging, where darker conditions are very common. It is important to note that the implemented interference effect applies the darkness effect using a gamma correction, which is not entirely realistic and could be improved with further research. The slight increase in the accuracy of the model when faced with moderate sharpness increases suggests that a sharpness effect applied in the pre-processing could improve the performance of the evaluated model. The smoke effect, a common byproduct in procedures involving coagulation, was found to significantly degrade the model's performance. This means that when using coagulation tools while employing this model, the results should be interpreted with caution. This study failed to confirm a definite interpretation of the improvement evident in higher intensities of the smoke effect. One possible explanation for the observed improvement in performance at higher levels of smoke intensity could be the already low baseline accuracy of around 20%. In the realm of such low accuracy values, even small changes or anomalies could appear as significant improvements. At higher levels of smoke intensity, it is conceivable that less crucial details in the images were obscured, leading to a slightly better performance. Another reason could be that the model could be trained with images containing no noise and a very high amount of noise. The smoke effect is similar to the noise effect and if the training images did not contain a medium amount of noise, it could explain the worse accuracy ratings on smoke transformations of medium intensity. However, it is important to note that these findings were based on a limited sample of 20 test images, so the role of chance in these results should not be discounted. Although the small sample size was sufficient for preliminary findings, it was not adequate for a comprehensive evaluation. This limitation was most evident in the evaluation of the model's performance under noise interference, where no clear trend could be established until a high amount of noise. However, due to time constraints and hardware limitations, a more extensive evaluation was not feasible. In summary, while the model demonstrates remarkable resilience to a range of interference factors, its limitations under specific conditions — extremely high noise, low contrast, dim or extremely bright lighting, and the presence of smoke — suggest that it may not be universally applicable for all endoscopic procedures. Further evaluations are advised, particularly in scenarios where these conditions could be present.

6.2 Evaluation of the Methodology

The type of interference effects implemented, were chosen, because of their common presence in real-world scenarios of endoscopic imaging. While the chosen effects cover a wide range of these common artifacts, they are not a comprehensive list and could be extended, especially when testing image segmentation models out of the medical domain. Types of interference effects that could be included are color distortions, which could be caused by variations in lighting conditions, shadows, artifacts caused by processing algorithms, especially in MRI and CT images, depth-of-field blur, which can occur in endoscopic imaging and is different from motion blur. Furthermore, speckle noise could be simulated, because it is a common interference effect

in ultrasound imaging. Speckle noise is granular interference that degrades the quality of ultrasound images.
To test the application, the data set of the ROBUST-MIS challenge was used. Because of its extensive size, it represents the real-world endoscopy settings very well, with a great variability of medical instrument locations and different environmental factors like lighting. While the statistical metrics used in the study were fundamental, they were appropriate for measuring the kinds of robustness relevant to image segmentation models. In particular, standard deviation and variance proved to be good indicators of robustness. Further research could explore more relevant statistical metrics in the context of robust image segmentation. Because the statistical metrics and the displayed graphs use the means of multiple test images, they are statistically significant and can adequately represent the robustness of the given algorithm to any given interference effect. Testing with even more images could further improve the significance, while increasing the time necessary for testing. Because the underlying model of the *Isensee* model, the U-net, has numerous parameters, and the resolution of the test images is quite large, the computational challenge involved in the testing process was significant. The evaluation was performed on a system with an AMD 5600X, 16 GB of DDR4 RAM and one Nvidia GTX 970. With the standard batch size of the evaluated model of 16, the model would crash during the testing process, when using a total amount of images higher than about 100. After decreasing the batch size to 8, the evaluation time was increased, but the model would reliably complete the testing process. Testing on a system with more and higher-end GPUS could enable the evaluation with more test images while maintaining a smaller time frame.

6.3 Future Research

Future research into the topic could compare multiple segmentation algorithm using this robustness framework. When evaluating with different data sets and different models, the quality of the quantification framework could be tested more in depth. Furthermore, the application could be used in a real world scenario of testing two similar segmentation model regarding a predefined interference effect. The chosen interference effect has to occur in the subsequent imaging procedure. Then the robustness rating of the framework could be compared to the real world performance of these segmentation models. The model with a higher robustness rating for the given interference effect should perform better in this test scenario. Another possibility of future research would be to extend the framework into an automated scoring system. This could work like a traffic light system, where a robust algorithm is certified and receives a green rating. Algorithms that are generally robust but are easily influenced by some interference effects, could receive a yellow label, with extra emphasis on the problematic effect.

This would greatly increase the usability and intuitiveness of the framework. Should it be utilized for licensing algorithms in the future, the steps needed for it to be approved by medical governing bodies for actual clinical use, could be explored. This could mean partnering with medical organizations for pilot testing and validation. Furthermore, expert clinical opinions on the framework's applicability, usability, and relevance could be obtained. Another important point is to work on the compliance of the framework to the medical domain, by implementing different standards like the ISO 9241-210 for creating user-friendly interfaces or the ISO 29119 standard, which focuses on the verification and validation process of software. Furthermore, it needs to be ensured that the framework adheres to Data Protection regulations such as the GDPR, to be able to be used in a medical setting. An important option of improving the framework and tailoring it to the real-world needs of its users could be to implement a feedback loop, where users who have tested the application can rate it and describe where its limitations lie, in order for it to be further improved. Another avenue of research would be to add compatibility tests to the framework. Interference effects that cause the model to malfunction completely, could produce an exception that informs the user about the incompatibility. This means that the framework does not only test the robustness, but also provides an error detection functionality. Currently, the application can already store multiple test results for a single model. With future research, the framework could be extended to include a dashboard of the model's robustness improvements over time. This could allow developers to use this framework to improve the robustness of new image segmentation models. Existing interference effects could also be improved, especially the glare, smoke and resolution effect. The glare effect could be improved, by randomizing the size and location of the effect. This could provide a more realistic testing scenario and insures that the results are not skewed, because the location of the effect is always in a location of the image, where no important structures are located. The smoke effect could be improved by implementing a more realistic interference effect that is significantly different from the noise effect. Finally, the resolution effect could be improved, by using the nearest neighbor method to rescale the test image. Instead of a blur that the currently used bilinear interpolation produces, the nearest neighbor method would result in a pixelated image, better representing actual drops in bandwith.

7 Conclusion

7.1 Summary of Key Findings

The thesis successfully addressed its objective of quantifying the robustness of image segmentation algorithms in a medical context, specifically focusing on the *Isensee* model submission of the ROBUST-MIS challenge. Because of the application of this model, the focus was heavily on analyzing endoscopic images. While evaluating the model, a diverse range of interference factors including noise, glare, darkness, contrast, smoke, vignette, motion blur and resolution were applied to the test data set. Because the evaluated model was already designed in the context of a Robustness Challenge, it was robust regarding the interference effects, with extreme darkness and noise effects being the exception. The model was most robust against sharpness, with a variance of 0.01 and a standard deviation of 0.09. It was least robust against the darkness effect, with a variance of 0.12 and a standard deviation of 0.33. To provide deeper insight into the robustness quantification, meaningful metrics like the mean, median, standard deviation, variance and extreme values were calculated for each transformation. Especially the variance metric was found to be a good indicator of robustness. However, the variance should not be used as the sole measurement of robustness, because a model that always scores a bad accuracy score will also exhibit low variance. In that case, the robustness would be high, but the model would still not be fit to be used in real-world scenarios.

7.2 Contributions and Impact

One contribution of this thesis is the creation of an evaluation framework for testing the robustness of image segmentation algorithms in a medical imaging context. This framework lays the foundation for further research into a tool that could potentially be used for the licensing of medical image segmentation algorithms. Because of the critical nature of medical applications, an objective label for robustness is imperative. This framework already quantifies the robustness against common interference effects into a comprehensible metric, which is helpful for medical practitioners when deciding which assistance model to use, depending on the environment of the procedure. Furthermore, because this framework offers deep statistical insights into the performance variations under multiple interfering conditions, developers could use this tool while creating new image segmentation algorithms, paving the way for improvements in robustness. To make further research into the topic easier, this framework is designed to be modular and can be extended in a couple of ways. First of all, more interference effects could be added to cover a wider range of interference effects in the real word. Furthermore, multiple evaluation metrics could be added, to enable accurate evaluation tailored to the specific segmentation algorithm. Further research could also be invested in finding more accurate value ranges for the interference effects and improving the realism and efficiency of the transformations. If further research in this area is undertaken, it could pave the way to more reliable medical image analysis and an increased awareness not only for the accuracy of such systems, but also the robustness.

References

Agrawal, A., & Raskar, R. (2007). Resolving objects at higher resolution from a single motion-blurred image. https://doi.org/10.1109/cvpr.2007.383030

Altindis, S. F. (2021, September 2). *Benchmarking the robustness of instance segmentation models.* https://arxiv.org/abs/2109.01123

Aydin, O. U., Taha, A. A., Hilbert, A., Khalil, A. A., Galinovic, I., Fiebach, J. B., Frey, D., & Madai, V. I. (2021). On the usage of average hausdorff distance for segmentation performance assessment: Hidden error when used for ranking. *European Radiology Experimental*, *5*(1). https://doi.org/10.1186/s41747-020-00200-2

Badrinarayanan, V. (2015, November 2). *Segnet: A deep convolutional encoder-decoder architecture for image segmentation.* https://arxiv.org/abs/1511.00561

Bergen, T., Wittenberg, T., & Münzenmayer, C. (2015). Shading correction for endoscopic images using principal color components. *International Journal of Computer Assisted Radiology and Surgery*, *11*(3), 397–405. https://doi.org/10.1007/s11548-015-1273-3

Bhandari, P. (2023). Descriptive statistics | definitions, types, examples. *Scribbr.* https://www.scribbr.com/statistics/descriptive-statistics/

Bianchi, F., Masaracchia, A., Barjuei, E. S., Menciassi, A., Arezzo, A., Koulaouzidis, A., Stoyanov, D., Dario, P., & Ciuti, G. (2019). Localization strategies for robotic endoscopic capsules: A review. *Expert Review of Medical Devices*, *16*(5), 381–403. https://doi.org/10.1080/17434440.2019.1608182

Cheng, L. (2022, March 31). *Human instance segmentation and tracking via data association and single-stage detector.* https://arxiv.org/abs/2203.16966

Deserno, T. (2008). Medical image processing. *Optipedia, SPIE Press.*

DKFZ, Heidelberg University Hospital, & Nationales Zentrum für Tumorerkrankungen Heidelberg (NCT). (2019a). *Robust-mis challenge 2019.* https://www.synapse.org/#!Synapse:syn18779624/wiki/592660

DKFZ, Heidelberg University Hospital, & Nationales Zentrum für Tumorerkrankungen Heidelberg (NCT). (2019b). *Submission instructions.* https://www.synapse.org/Portal/filehandle?ownerId=syn18779624&ownerType=ENTITY&fileName=SubmissionInstructions.pdf&preview=false&wikiId=592666

Drenkow, N. (2021, December 1). *A systematic review of robustness in deep learning for computer vision: Mind the gap?* https://arxiv.org/abs/2112.00639

GeeksforGeeks. (2023). *Benefit of using mvc.* https://www.geeksforgeeks.org/benefit-of-using-mvc/

Goyal, B. (2018, September 21). *Noise issues prevailing in various types of medical images.* https://biomedpharmajournal.org/vol11no3/noise-issues-prevailing-in-various-types-of-medical-images/

Guo, D., Pei, Y., Zheng, K., Yu, H., Lu, Y., & Wang, S. (2020). Degraded image semantic segmentation with dense-gram networks. *IEEE Transactions on Image Processing*, *29*, 782–795. https://doi.org/10.1109/TIP.2019.2936111

He, Z., Wang, P., Liang, Y., Fu, Z., & Ye, X. (2021). Clinically available optical imaging technologies in endoscopic lesion detection: Current status and future perspective. *Journal of Healthcare Engineering*, *2021*, 1–27. https://doi.org/10.1155/2021/7594513

Hendrycks, D. (2019, March 28). *Benchmarking neural network robustness to common corruptions and perturbations.* https://arxiv.org/abs/1903.12261

Isensee, F., & Maier-Hein, K. H. (2020). Or-unet: An optimized robust residual u-net for instrument segmentation in endoscopic images. *ResearchGate.* https://www.researchgate.net/publication/340963424_OR-UNet_an_Optimized_Robust_Residual_U-Net_for_Instrument_Segmentation_in_Endoscopic_Images

Kirillov, A. (2018, January 3). *Panoptic segmentation.* https://arxiv.org/abs/1801.00868

Kirillov, A. (2019, January 8). *Panoptic feature pyramid networks.* https://arxiv.org/abs/1901.02446

Koivukangas, T., Katisko, J., & Koivukangas, J. (2013). Technical accuracy of optical and the electromagnetic tracking systems. *SpringerPlus*, *2*(1). https://doi.org/10.1186/2193-1801-2-90

Kürbig, J., & Sauter, M. (2005). *Seminar: Bildsegmentierung und computer vision ws 2005/6.* Retrieved August 12, 2023, from https://www.mathematik.uni-ulm.de/stochastik/lehre/ws05_06/seminar/ausarbeitung_sauter.pdf#:~:text=Das%20Verfahren%20der%20Bildsegmentierung%20findet%20Anwendung%20in%20zahlreichen,auch%20zur%20Segmentierung%20von%20R%C3%B6ntgenbildern%20und%20der%20Computertomographie.

Luna, J. C. (2023, March 10). *Top programming languages for data scientists in 2023.* https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022

Maier-Hein, L., Wagner, M., Roß, T., Reinke, A., Bodenstedt, S., Full, P. M., Hempe, H., Mindroc-Filimon, D., Scholz, P., Tran, T. N., Bruno, P., Kisilenko, A., Müller, B., Davitashvili, T., Capek, M., Tizabi, M. D., Eisenmann, M., Adler, T., Gröhl, J., . . . Müller-Stich, B. P. (2021). Heidelberg colorectal data set for surgical data science in the sensor operating room. *Scientific Data*, *8*(1). https://doi.org/10.1038/s41597-021-00882-2

Mandt, S. (2017, April 13). *Stochastic gradient descent as approximate bayesian inference*. https://arxiv.org/abs/1704.04289

MathWorks. (2023). *Getting started with mask r-cnn for instance segmentation*. https://www.mathworks.com/help/vision/ug/getting-started-with-mask-r-cnn-for-instance-segmentation.html

Matsuo, Y., Yasuda, H., Kato, M., Kiyokawa, H., Ozawa, M., Sato, Y., Ikeda, Y., Ozawa, S., Yamashita, M., Fukushima, T., Yamamoto, H., Takagi, M., & Itoh, F. (2018). Endoscopic small-capacity forceps increase the pathological diagnosis of gastric indefinite neoplasia. *The Turkish journal of gastroenterology*, *29*(4), 481–487. https://doi.org/10.5152/tjg.2018.17347

Maze, A. (2022). *The importance of iso 13485 certification to medical device manufacturing | smithers*. https://www.smithers.com/resources/2022/december/importance-of-iso-13485-certification

Mittal, H., Pandey, A. C., Saraswat, M., Kumar, S., Pal, R., & Modwel, G. (2021). A comprehensive survey of image segmentation: Clustering methods, performance parameters, and benchmark datasets. *Multimedia Tools and Applications*, *81*(24), 35001–35026. https://doi.org/10.1007/s11042-021-10594-9

Müller, D., Soto-Rey, I., & Kramer, F. (2022). Towards a guideline for evaluation metrics in medical image segmentation. *BMC Research Notes*, *15*(1). https://doi.org/10.1186/s13104-022-06096-y

Oala, L., Murchison, A. G., Balachandran, P., Choudhary, S., Fehr, J., Leite, A. W., Goldschmidt, P., Johner, C., Schörverth, E. D. M., Nakasi, R., Meyer, M., Cabitza, F., Baird, P., Prabhu, C., Weicken, E., Liu, X., Wenzel, M., Vogler, S., Akogo, D. A., … Wiegand, T. (2021). Machine learning for health: Algorithm auditing amp; quality control. *Journal of Medical Systems*, *45*(12). https://doi.org/10.1007/s10916-021-01783-y

OpenAI. (2017). *Attacking machine learning with adversarial examples*. https://openai.com/research/attacking-machine-learning-with-adversarial-examples

OpenCV. (2020, November 4). *About - opencv*. https://opencv.org/about/

Potnuru, M., & Naick, B. S. (2023). Semantic segmentation of mri images for brain tumour detection with shufflenet-based unet. *SN computer science*, *4*(5). https://doi.org/10.1007/s42979-023-01878-y

Processing.org. (n.d.). *Pimage documentation* [Accessed: 2023-08-28]. https://processing.org/reference/PImage_blend_.html

Pygame. (n.d.). *Pygame.surface — pygame v2.6.0 documentation* [Accessed: 2023-09-02]. https://www.pygame.org/docs/ref/surface.html

Qin, J., Liu, T., Wang, Z., Liu, L., & Fang, H. (2022). Gct-unet: U-net image segmentation model for a small sample of adherent bone marrow cells based on a gated channel transform module. *Electronics*, *11*(22), 3755. https://doi.org/10.3390/electronics11223755

Ragan, A. (2018). Taking the confusion out of confusion matrices - towards data science. https://towardsdatascience.com/taking-the-confusion-out-of-confusion-matrices-c1ce054b3d3e

Rahmonov, J. (2022). Pycharm for productive python development (guide). *realpython.com.* https://realpython.com/pycharm-guide/

Reinke, A. (2023, February 3). *Understanding metric-related pitfalls in image analysis validation.* arXiv: 2302.01790. arXiv.org. https://arxiv.org/abs/2302.01790

Reiter, W. (2020). Improving endoscopic smoke detection with semi-supervised noisy student models. *Current Directions in Biomedical Engineering*, *6*(1). https://doi.org/10.1515/cdbme-2020-0026

Ren, H., Huang, T., & Yan, H. (2021). Adversarial examples: Attacks and defenses in the physical world. *International Journal of Machine Learning and Cybernetics*, *12*(11), 3325–3336. https://doi.org/10.1007/s13042-020-01242-z

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Roß, T., Reinke, A., Full, P. M., Wagner, M., Kenngott, H., Apitz, M., Hempe, H., Mindroc-Filimon, D., Scholz, P., Tran, T. N., Bruno, P., Arbeláez, P., Bian, G., Bodenstedt, S., Bolmgren, J. L., Bravo-Sánchez, L., Chen, H., González, C., Guo, D., . . . Maier-Hein, L. (2021). Comparative validation of multi-instance instrument segmentation in endoscopy: Results of the robust-mis 2019 challenge. *Medical Image Analysis*, *70*, 101920. https://doi.org/10.1016/j.media.2020.101920

Ruder, S. (2020). An overview of gradient descent optimization algorithms. *ruder.io.* https://www.ruder.io/optimizing-gradient-descent/#gradientdescentvariants

Sato, T. (2021). Txi: Texture and color enhancement imaging for endoscopic image enhancement. *Journal of Healthcare Engineering*, *2021*, 1–11. https://doi.org/10.1155/2021/5518948

Segmentation. (2005). In *Digital image processing* (pp. 449–462). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-27563-0_16

Shapel, M. (2023). 10 reasons why vue.js is best for app development [+ benefits]. *SaM Solutions.* https://www.sam-solutions.com/blog/why-vue-js/

Sharma, R., Saqib, M., Lin, C., & Blumenstein, M. (2022). A survey on object instance segmentation. *SN computer science*, *3*(6). https://doi.org/10.1007/s42979-022-01407-3

Shen, Y., Zhang, D., Zheng, Y., Li, Z., Fu, L., & Ye, Q. (2023). Training-free instance segmentation from semantic image segmentation masks.

SQLite. (n.d.). *About sqlite.* https://www.sqlite.org/about.html

Team, E. A. (n.d.). *Accuracy vs. precision vs. recall in machine learning: What's the difference?* [Accessed: 2023-08-12]. https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall/

Vacavant, A. (2017). A novel definition of robustness for image processing algorithms. *Lecture Notes in Computer Science*, 75–87. https://doi.org/10.1007/978-3-319-56414-2_6

VectorLogoZone. (n.d.). *Vue.js logo.* Retrieved August 28, 2023, from https://www.vectorlogo.zone/logos/vuejs/vuejs-ar21.svg

Wang, Z., Wang, E., & Zhu, Y. (2020). Image segmentation evaluation: A survey of methods. *Artificial Intelligence Review*, *53*(8), 5637–5674. https://doi.org/10.1007/s10462-020-09830-9

Wikipedia contributors. (2023a). *Docker container engine logo* [Accessed: 28.08.2023]. https://upload.wikimedia.org/wikipedia/commons/4/4e/Docker_%28container_engine%29_logo.svg

Wikipedia contributors. (2023b). *Flask logo* [Accessed: 28.08.2023]. https://upload.wikimedia.org/wikipedia/commons/3/3c/Flask_logo.svg

Wikipedia contributors. (2023c). *Gamma correction* [Accessed: 02.09.2023]. https://en.wikipedia.org/wiki/Gamma_correction

Wikipedia contributors. (2023d). *Python imaging library* [Accessed: 02.09.2023]. https://en.wikipedia.org/wiki/Python_Imaging_Library

Wikipedia contributors. (2023e). *Python logo* [Accessed: 28.08.2023]. https://upload.wikimedia.org/wikipedia/commons/f/f8/Python_logo_and_wordmark.svg

Wikipedia contributors. (2023f). *Sqlite logo* [Accessed: 28.08.2023]. https://upload.wikimedia.org/wikipedia/commons/3/38/SQLite370.svg

Yang, X., Chen, Y., Tao, R., Zhang, Y., Liu, Z., & Shi, Y. (2020). Endoscopic image deblurring and super-resolution reconstruction based on deep learning. *2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, 168–172. https://doi.org/10.1109/ICAICE51518.2020.00039

Yusefi, H., & Helfield, B. (2022). Ultrasound contrast imaging: Fundamentals and emerging technology. *Frontiers in Physics*, *10*. https://www.frontiersin.org/articles/10.3389/fphy.2022.791145/full

Zewe, A. (2022, May 5). *Machine learning explainability.* MIT News. https://news.mit.edu/2022/machine-learning-explainability-0505

Zhou, Y., Hu, Z., Xuan, Z., Wang, Y., & Hu, X. (2022). Synchronizing detection and removal of smoke in endoscopic images with cyclic consistency adversarial nets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *PP*(99), 1–12. https://ieeexplore.ieee.org/abstract/document/9878179

Zizzo, G., Fraser, K., Hameed, M. Z., & Purcell, M. (2021). *Adversarial robustness toolbox* [Accessed: 2023-09-14]. IBM Research. https://research.ibm.com/projects/adversarial-robustness-toolbox