

Politecnico di Torino

Politecnico di Torino

Department of Control and Computer Engineering

Master degree in Computer Engineering
Artificial Intelligence and Data Analytics

Analysis of the Application of Computer Vision and Advanced Models (such as GPT-Vision) Based on Large Language Models (LLMs) for Damage Detection and Classification Projects

---

Author:
Emilio Joseph Grieco

Supervisor:
prof. Andrea Bottino

Academic Year 2025-26

Abstract

Automated quality inspection plays a central role in modern manufacturing, where consistent product quality must be guaranteed within the tight cycle times of the assembly line. This thesis addresses the automatic detection of defects in the positioning of a sealing gasket on a tractor cabin frame, a step that is currently performed manually and inspected only prior to roof mounting, when a misaligned gasket can still be corrected before it becomes costly to fix.

Two fundamentally different technological strategies are designed, implemented, and compared. The first is an engineered computer vision pipeline that combines a deep learning–based segmentation model (the Segment Anything Model) with classical image-processing techniques: the gasket and the metal frame are segmented into binary masks, their upper boundaries are extracted as polylines, and the local deviation between them is compared against an empirically learned tolerance envelope to produce an `OK/DEFECT` verdict. A dedicated automatic point-selection algorithm, based on texture and colour cues, makes the segmentation robust to the moderate acquisition shifts that occur in a production environment. The second strategy reframes the problem as an in-context, few-shot visual classification task, delegated to a general-purpose Vision-Language Model (`gpt-4o`) that is guided solely by natural-language instructions and a small set of labelled reference images, without any task-specific training. A systematic experimental protocol varies the number of reference images and the richness of the prompt to characterise the model's behaviour.

The two approaches are evaluated on a common test set and compared from both a technical and an economic perspective. Technically, the engineered pipeline proves highly accurate and interpretable, reaching an F1-score of 0.98 with perfect recall and a single false positive, whereas GPT-4o also attains perfect recall but a lower precision (best F1-score of 0.85), improving only when the prompt is enriched with detailed per-image annotations and several reference images. Economically, both automated solutions reduce the direct inspection cost by one to two orders of magnitude with respect to a dedicated human operator, yet they embody opposite cost structures—an upfront-heavy computer vision asset versus a pay-as-you-go VLM service—so that the preferable option depends on production volume, planning horizon, and non-monetary factors such as data sovereignty. The resulting analysis provides practical guidance on when each strategy is the more appropriate choice for an industrial defect-detection task of this kind, and points to a hybrid solution as a promising direction.

Keywords: defect detection; quality inspection; computer vision; image segmentation; Vision-Language Models; GPT-4o; manufacturing automation.

Contents

1 Introduction **6**
1.1 Context and Motivation 6
1.2 Problem Statement 6
1.3 Objectives and Approach 6
1.4 Contributions 7
1.5 Thesis Structure 7

2 Technological Perspective: Classical and Deep Learning-Based Computer Vision vs Vision-Language Models **9**
2.1 Application Context 9
2.2 Computer Vision-Based Approach 10
2.2.1 Segmentation via DNN-based model 10
2.2.2 Polylines extraction from binary masks 11
2.2.3 Defect Quantification using polylines 12
2.2.4 Geometric Assumptions 14
2.2.5 Automatic Point Selection via Classical Computer Vision 14
2.3 Defect Detection Using Vision-Language Models (GPT-4o) 16
2.3.1 Experimental Variables 16

3 Solution results **18**
3.1 Computer vision-based approach result 18
3.1.1 Test Protocol 18
3.1.2 Class Separation and Threshold Selection 18
3.1.3 Quantitative Results 19
3.1.4 Discussion 19
3.2 Impact of Prompting and Visual Context in Vision-Language Models . . . 20
3.2.1 Evaluation Metrics 20
3.2.2 Results for Different Prompt Qualities 21
3.2.3 Uncertainty Analysis 25
3.2.4 Additional Considerations 27
3.2.5 Summary of Experimental Findings 29
3.3 Comparison of the Two Approaches 29

4 Economic Analysis: Economic Sustainability of Automated Defect Detection in Industrial Contexts **31**
4.1 Cost Model and Assumptions 31
4.2 Manual Inspection by a Human Operator 32

4.3 Cost of the VLM-Based Approach (OpenAI GPT-4o) . . . 33
4.4 Cost of the Computer Vision-Based Approach . . . 33
4.5 Total Cost of Ownership and Break-Even Analysis . . . 34
4.6 Incorporating the Cost of Errors . . . 35
4.7 Three-Way Comparison Summary . . . 36
4.8 Discussion of Economic Sustainability . . . 37

5 Conclusions **38**
5.1 Summary of the Work . . . 38
5.2 Main Findings . . . 38
5.3 Limitations . . . 39
5.4 Future Work . . . 39
5.5 Closing Remarks . . . 40

A The Segment Anything Model (SAM) **41**
A.1 Motivation and Key Idea . . . 41
A.2 Architecture at a Glance . . . 41
A.3 Types of Prompts . . . 42
A.4 Effect of the Number of Points . . . 42
A.5 Prompting Strategy Adopted in This Thesis . . . 43

B Classical Computer Vision Techniques for Automatic Point Selection **44**
B.1 The Role of Point Selection in the Pipeline . . . 44
B.2 Texture Analysis via the Laplacian Operator . . . 45
B.2.1 Texture as a Discriminative Cue . . . 45
B.2.2 The Laplacian as a Second-Order Derivative . . . 45
B.2.3 Discrete Approximation . . . 46
B.2.4 From Laplacian Response to a Texture Measure . . . 47
B.2.5 Thresholding into a Binary Texture Mask . . . 47
B.2.6 Practical Considerations . . . 47
B.3 Color Analysis in the CIELAB color Space . . . 48
B.3.1 Why Not Work in RGB? . . . 48
B.3.2 The CIELAB Color Space . . . 48
B.3.3 Measuring Color Similarity: the ΔE Distance . . . 49
B.3.4 Building the Color Mask . . . 49
B.3.5 Strengths and Limitations of the Color Cue . . . 50
B.4 Fusion of the Two Cues and Safe Point Selection . . . 50
B.5 Summary . . . 51

C The GPT-4o Vision-Language Model **52**
C.1 Historical Context 52
C.2 What Is a Vision-Language Model? 52
C.3 Architecture at a Glance 53
C.4 In-Context Learning and Few-Shot Prompting 53
C.5 The Role of the Prompt 54
C.6 Relevance to This Thesis 54

List of Figures

1 Acquired 4K image of the frame region, with the privacy mask applied to obscure any visible faces. The gasket is the sponge-like profile running along the top of the metal frame. 10
2 Upper-boundary polylines extracted from the binary masks of the frame and the gasket. The vertical distance between the two curves, measured at each horizontal position, defines the deviation profile. 12
3 Empirical "OK gap" envelope obtained by aggregating the deviation profiles of the 70 correctly positioned reference samples. Deviations falling inside the band are considered acceptable. 13
4 Final inference result overlaid on the acquired image: gasket regions within the OK envelope are shown in green, out-of-envelope (defective) regions in red. 14
5 Comparison of GPT-4o performance across LOW, MEDIUM, and HIGH prompt qualities as a function of the number of reference images per class. While recall remains consistently close to or equal to 1.00, precision and F1-score increasingly benefit from richer prompt descriptions and larger annotated reference sets, with the HIGH-quality configuration achieving the strongest overall performance. 22

1 Introduction

1.1 Context and Motivation

Modern manufacturing increasingly relies on automated quality inspection to guarantee consistent product quality, reduce rework, and keep pace with the throughput demanded by contemporary assembly lines. In the agricultural machinery sector, tractors are assembled through a sequence of tightly scheduled operations, each of which must be completed within a fixed cycle time. One such operation is the mounting of the roof onto the cabin frame, which is sealed by a rubber gasket placed along the frame profile. Because the gasket is positioned manually, its final placement is inherently variable and depends on operator skill, fatigue, and the pace of the line. A misaligned, folded, or incompletely seated gasket compromises the seal between the roof and the cabin and, once the roof is mounted, can only be corrected through costly disassembly and remounting.

Detecting such defects *before* the roof is assembled is therefore highly valuable: at that moment the fault is still cheap and quick to fix, the product quality is protected, and the risk of a sealing failure reaching the customer is avoided. This creates a concrete need for an automated inspection system that can deliver a clear, actionable `OK/DEFECT` verdict on the gasket placement, quickly enough to fit within the station's cycle time and without requiring a specialized operator to interpret the result.

1.2 Problem Statement

The problem addressed in this thesis is the automatic detection of defects in the positioning of a sealing gasket on a tractor cabin frame, from images acquired at the assembly station immediately prior to roof mounting. The task is challenging for several reasons: the defects of interest are small, local deviations of the gasket profile, often only a few millimeters in size; the gasket and the metal frame must be distinguished reliably despite variations in illumination and material appearance; and the acquisition conditions are not perfectly repeatable, since the tractor may be stopped slightly further forward or backward, introducing shifts between images. Any viable solution must also respect the operational constraints of a production line (limited cycle time, robustness to normal variability, and interpretability of the output) as well as data-protection requirements, since operators may be visible in the acquired images.

1.3 Objectives and Approach

The objective of this work is to design, implement, and evaluate a system capable of detecting gasket-positioning defects, and to compare two fundamentally different techno-

logical strategies for solving it:

1. a **computer vision-based approach**, combining classical image-processing techniques with deep learning-based segmentation to produce an explicit geometric measurement of the gasket position and compare it against a learned tolerance envelope;
2. an approach based on **Vision-Language Models (VLMs)**, specifically `gpt-4o`, which reframes the task as an in-context, few-shot visual classification problem guided by natural-language instructions and a small set of labelled reference images.

These two strategies embody contrasting philosophies. The first is an engineered, measurement-driven pipeline that offers precise, interpretable, and controllable behaviour at the cost of dedicated development and calibration. The second applies a general-purpose multimodal model with little or no task-specific development, trading engineering effort for reliance on a large pre-trained model. Comparing them makes it possible to assess not only which achieves better detection performance, but also what each costs to build and operate (an increasingly relevant question as powerful foundation models become readily available).

1.4 Contributions

The main contributions of this thesis are the following:

- the design and implementation of a complete computer vision pipeline for gasket-defect detection, including deep-learning segmentation, boundary (polyline) extraction, an empirically learned tolerance envelope, and a robust automatic prompt-point selection algorithm based on classical texture and color cues;
- the design and implementation of a VLM-based alternative, together with a systematic experimental protocol that varies the number of reference images and the richness of the prompt;
- a comparative evaluation of the two approaches from both a technical and an economic perspective, providing practical guidance on when an engineered pipeline or a general-purpose VLM is the more appropriate choice for an industrial inspection task of this kind.

1.5 Thesis Structure

The remainder of this thesis is organized as follows. **Section 2** presents the technological perspective, describing in detail the design and implementation of both the computer

vision pipeline and the VLM-based approach. **Section 3** reports and analyses the experimental results on a common test set, showing that the computer vision pipeline reaches an F1-score of 0.98 with perfect recall, while the VLM also achieves perfect recall but a lower precision, and examining the sensitivity of the VLM to prompt quality and reference selection. **Section 4** examines the problem from an economic perspective, comparing manual inspection, the computer vision pipeline, and the VLM in terms of total cost of ownership, break-even volume, and the cost of classification errors. Finally, **Section 5** summarizes the findings and outlines possible directions for future work.

2 Technological Perspective: Classical and Deep Learning-Based Computer Vision vs Vision-Language Models

2.1 Application Context

The industrial partner operates in the agricultural machinery sector and specializes in the production of tractors. One of the key steps in the assembly line involves mounting the roof onto a metal frame equipped with a sealing gasket. Before the roof can be positioned, an operator must manually place the gasket along the designated frame profile. The gasket ensures a watertight and airtight seal between the roof and the cabin frame; if it is misaligned, folded, or not fully seated along the profile, the subsequent roof assembly can trap the gasket in an incorrect position, compromising sealing performance and, in the worst case, requiring the roof to be removed and remounted. Because the gasket is placed by hand, its final position is inherently variable and depends on operator skill, fatigue, and the pace imposed by the assembly line.

The objective of this work is to detect defects in the positioning of the gasket *prior* to roof assembly, so that a defective placement can be corrected while it is still cheap and easy to do so. This constraint is important: the inspection must fit within the cycle time of the station and must produce a clear, actionable `OK`/`DEFECT` verdict without slowing the line or requiring a specialized operator to interpret the result. Detecting the defect at this stage avoids costly downstream rework, protects the quality of the final product, and reduces the risk of sealing failures reaching the customer.

Two alternative solutions were developed and compared:

1. a computer vision-based approach;
2. an approach leveraging Vision-Language Models (VLMs).

The first solution combines classical computer vision techniques with deep learning-based segmentation, producing an explicit geometric measurement of the gasket position; the second exploits the multimodal reasoning capabilities of GPT-4o, treating the problem as a visual classification task guided by natural-language instructions and a small set of reference examples. The two approaches embody two very different philosophies: an engineered, measurement-driven pipeline versus a general-purpose model applied with little or no task-specific development, and the remainder of this chapter describes how each was designed and implemented. Their comparative performance and cost are analyzed in the following chapters.

2.2 Computer Vision-Based Approach

A high-resolution (4K) camera is installed to capture the portion of the frame where the gasket is applied. The high resolution is not a luxury but a requirement: the defects of interest manifest as small, local deviations of the gasket profile, often of the order of a few millimeters, and resolving them reliably along the entire length of the frame demands a dense, sharp image in which the thin boundary of the gasket is clearly distinguishable from the underlying metal. A privacy mask is applied to obscure any potentially visible faces, so that operators working around the station are never identifiable in the stored images; this is a deliberate design choice to comply with data-protection requirements (GDPR) and to make the dataset safe to retain and process. The acquired images are then processed to extract the geometry of both the metal frame and the gasket.

At a high level, the pipeline proceeds through the following stages: (i) image acquisition and privacy masking; (ii) segmentation of the frame and the gasket into binary masks; (iii) extraction of their upper boundaries as polylines; (iv) computation of the local deviation between the two boundaries; and (v) comparison of this deviation against a learned tolerance envelope to produce the final `OK`/`DEFECT` decision. Each of these stages is detailed in the following subsections.

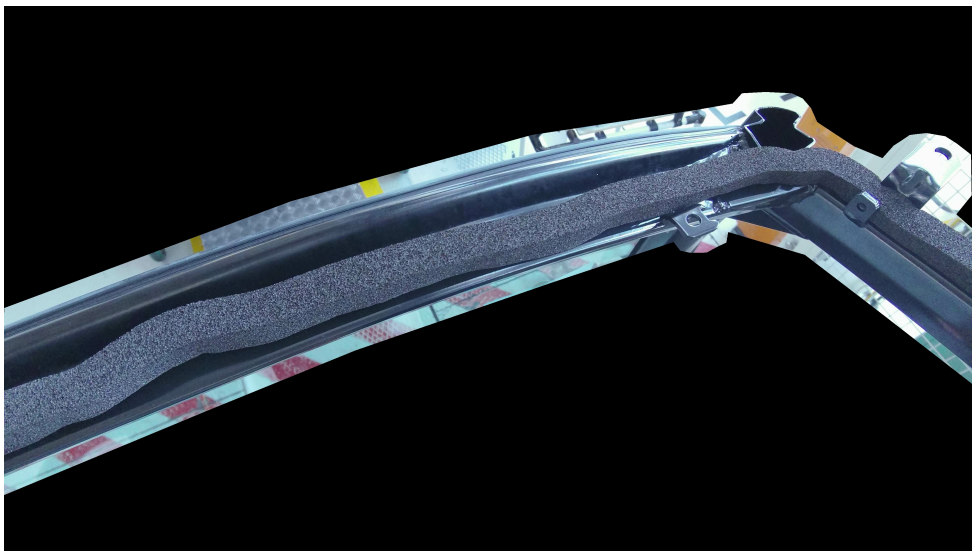


Figure 1: Acquired 4K image of the frame region, with the privacy mask applied to obscure any visible faces. The gasket is the sponge-like profile running along the top of the metal frame.

2.2.1 Segmentation via DNN-based model

The first step is to obtain binary masks of both the frame and the gasket, since all subsequent geometric reasoning is carried out on these masks rather than on the raw

image. To do so, we perform image segmentation using SAM (Segment Anything Model), a foundation model based on deep neural networks. SAM is a *promptable* segmentation model: rather than being trained to recognize a fixed set of classes, it takes an image together with a set of prompts, such as points, boxes, or coarse masks (if you are not familiar with SAM check Appendix A), and returns the segmentation mask of the object indicated by those prompts. Having been trained on an extremely large and diverse dataset, it generalizes in a zero-shot manner to objects and domains it has never seen during training, which makes it well suited to an industrial setting where a large, manually annotated dataset of gaskets and frames is not available.

This zero-shot capability is the main reason SAM was preferred over training a dedicated segmentation network from scratch: building a supervised segmenter would have required collecting and pixel-wise annotating a substantial number of images, an effort that is expensive and slow in a production environment. We empirically verified that by providing two prompt points on the gasket and two on the frame, both at the left and right extremes, SAM produces the corresponding binary masks, preserving the borders (which are precisely what this task is interested in) with a quality that is sufficient for our goals. Placing one pair of points on each object, distributed between the left and right sides of the image, gives SAM enough spatial context to segment the full extent of the elongated gasket and frame, rather than only a local patch around a single point.

2.2.2 Polylines extraction from binary masks

Once the binary masks of the frame and the gasket are available, the geometry relevant for defect detection is condensed into two curves, one per object, describing their upper boundaries. The extraction proceeds column by column: for each horizontal image position (each x coordinate), the algorithm scans the corresponding column of the binary mask and identifies the topmost foreground pixel, i.e. the highest point belonging to the object at that horizontal location. Collecting these topmost points across all columns yields, for each mask, an ordered sequence of points that traces its upper profile: a polyline. Columns in which the mask has no foreground pixel (for example, near the extremes or where the privacy mask removes content) are handled by interpolating between the neighbouring valid columns, so that both polylines are defined on a common, regularly sampled horizontal grid and can be compared point by point. A light smoothing is applied to suppress the pixel-level jaggedness introduced by segmentation, without erasing the genuine local deviations that constitute the defects of interest.

These polylines are used to represent the upper boundaries of both the frame and the gasket. Comparing the two curves at each horizontal position, by measuring the vertical distance between the gasket boundary and the frame boundary, produces a deviation

profile that quantifies, along the entire length of the frame, how far the gasket departs from the reference metal profile. This deviation profile is the single most important intermediate result of the pipeline: every subsequent decision is derived from it.

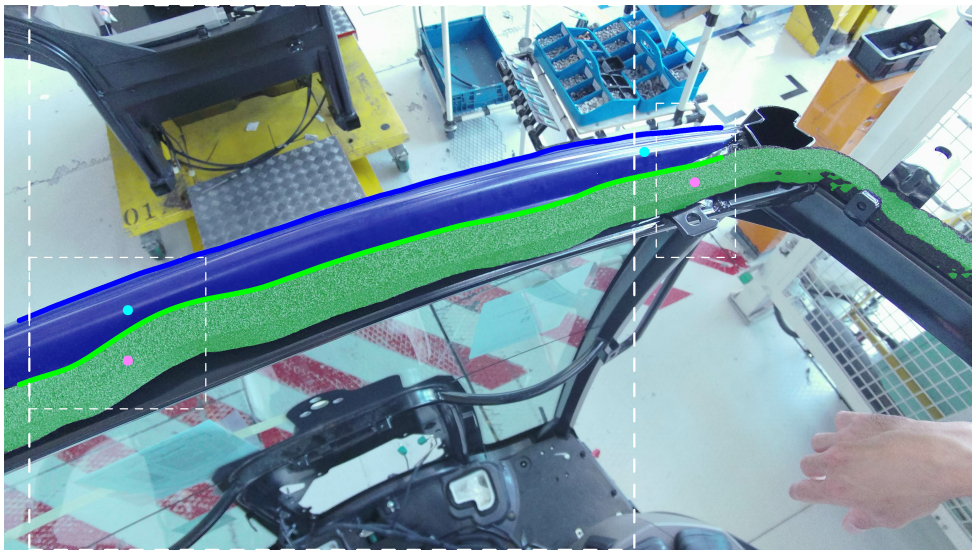


Figure 2: Upper-boundary polylines extracted from the binary masks of the frame and the gasket. The vertical distance between the two curves, measured at each horizontal position, defines the deviation profile.

2.2.3 Defect Quantification using polylines

Once the deviation profile between gasket and frame is obtained, two steps are required:

1. defining an acceptable tolerance range (the "OK gap");
2. classifying the gasket as `OK` or `DEFECT`.

To determine the acceptable gap, a domain expert produced multiple examples of correctly positioned gaskets, along with an equal number of defective cases. Among all these samples (101 `OK` and 33 `DEFECT`), 70 of the correctly positioned ones were used to construct the reference tolerance. The rationale is that the natural, defect-free variability of a correctly placed gasket (due to manual placement, slight differences between frames, and acquisition noise) should itself define what "acceptable" means, rather than a single hand-picked threshold.

By aggregating the 70 correctly positioned samples, an empirical *OK deviation envelope* was defined. Concretely, the deviation profiles of all 70 `OK` samples are aligned on the same horizontal grid, and at each horizontal position the range of deviations observed across the samples is used to delimit a band of acceptable values: a lower and an upper

bound that together enclose the region in which a correctly positioned gasket is expected to lie. This band is wide where correct gaskets naturally vary more and narrow where they are consistent, giving a position-dependent tolerance that a single global threshold could not capture.

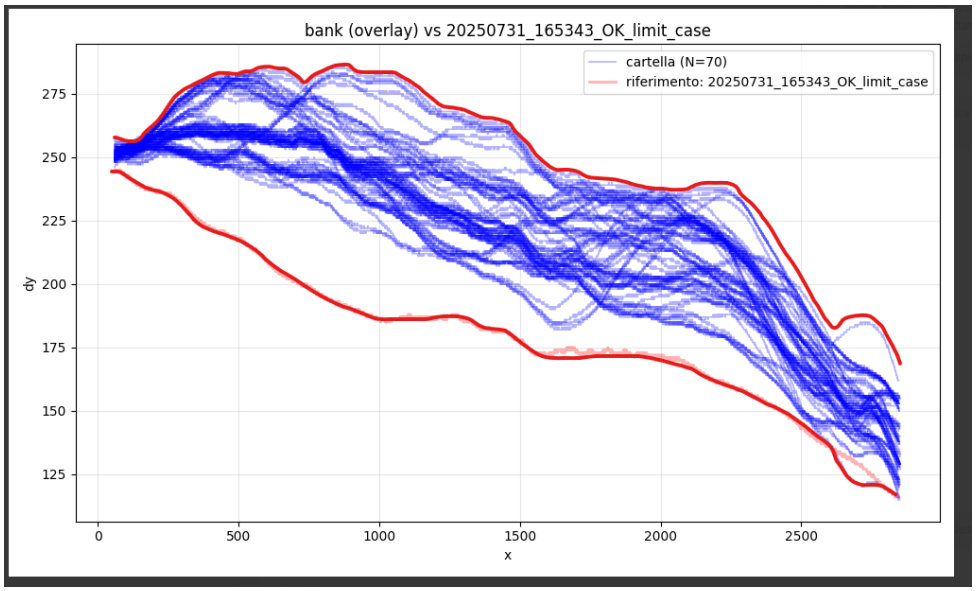


Figure 3: Empirical "OK gap" envelope obtained by aggregating the deviation profiles of the 70 correctly positioned reference samples. Deviations falling inside the band are considered acceptable.

During inference, points on the gasket profile whose deviation lies outside this envelope are marked as defective. The defect ratio is then computed as:

$$\text{defect ratio} = \frac{\text{number of defective points}}{\text{total number of points}}.$$

If the defect ratio exceeds a predefined threshold, empirically selected by observing the model's behaviour during inference and tuned accordingly, the sample is classified as `DEFECT`; otherwise, it is classified as `OK`. Using a ratio rather than the presence of a single out-of-band point makes the decision robust to isolated segmentation artefacts, while still reacting promptly when a genuine, spatially extended misalignment is present.

The final output can be visualized directly on the acquired image: the portions of the gasket whose deviation remains within the envelope are highlighted in green, while the portions that fall outside it are highlighted in red, giving the operator an immediate, interpretable indication of both the verdict and its location along the frame.

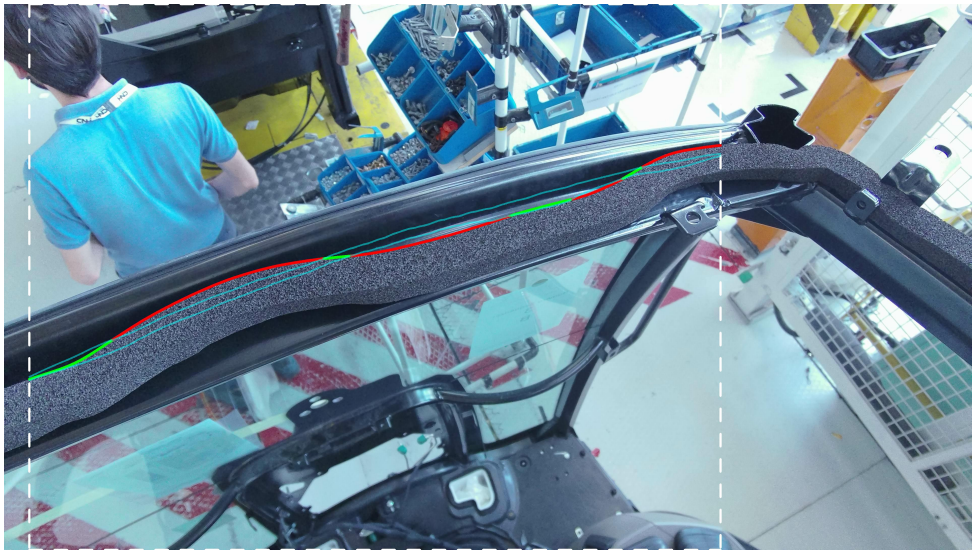


Figure 4: Final inference result overlaid on the acquired image: gasket regions within the OK envelope are shown in green, out-of-envelope (defective) regions in red.

2.2.4 Geometric Assumptions

During inference, each tractor is assumed to be imaged with a consistent perspective and zoom, since the camera is fixed and the tractor is always positioned at approximately the same station. In practice, however, some vertical and horizontal shifts between images are unavoidable, because the operator may stop the tractor slightly further forward or backward, or slightly higher or lower, in different instances. As a result, the same physical point on the frame does not always project onto exactly the same pixel coordinates from one acquisition to the next.

These shifts are a problem for the segmentation step, which relies on four prompt points (two on the gasket and two on the frame) to obtain the binary masks: prompt points placed at fixed image coordinates would, under a shift, no longer fall on the intended object and could land on the background or on the wrong structure. We solved this by introducing an automatic point-selection algorithm that detects both the gasket and the frame in each image and places the prompt points accordingly, remaining robust to moderate shifts. The tolerated range is of course limited, because an excessive displacement, which would in any case indicate an anomalous acquisition, is not accommodated, but within the normal operating range the algorithm reliably relocates the prompt points onto the correct objects.

2.2.5 Automatic Point Selection via Classical Computer Vision

The four prompt points used by SAM are automatically identified within their respective regions of interest (ROIs) using a combination of texture and color cues derived from

classical computer vision. Their computation is described below; a detailed treatment of both techniques is provided in Appendix B.

Texture analysis via Laplacian operator The sponge-like surface of the gasket exhibits a fine-grained, irregular micro-texture, characterized by rapid local variations in intensity. These high-frequency components are effectively captured by the Laplacian operator, which approximates the second spatial derivative of the image and therefore responds strongly to abrupt changes in grey levels.

Given a greyscale, contrast-enhanced version of the ROI, I, the Laplacian response is computed as:

$$L = \text{cv2.Laplacian}(I, \text{CV_64F}), \qquad \text{variance} = L^2.$$

Pixels with variance $> \tau_{\text{texture}}$ are retained, producing a binary texture mask that highlights regions where the intensity fluctuates rapidly, such as the porous rubber gasket, while largely suppressing the smoother, metallic areas of the frame.

Color analysis in CIELAB space To further refine the localization of the gasket, a color-based mask is computed in the CIELAB color space, which is approximately perceptually uniform, meaning that Euclidean distances in this space correspond closely to differences perceived by the human eye. Each pixel in the ROI is compared to one or more reference color patches representative of the gasket material, and a similarity score (the Euclidean distance ΔE) is evaluated. Pixels whose color lies within a predefined distance from the reference are classified as gasket candidates.

Fusion of texture and color cues and safe point selection The final candidate regions are obtained by combining the texture-based and color-based masks. This fusion exploits both the characteristic sponge-like micro-texture and the specific color of the gasket, leading to a robust identification of the gasket area within two predefined ROIs: a left ROI and a right ROI along the frame. Relying on two complementary cues is what makes the detection robust: texture alone can be misled by other rough regions, and color alone can be misled by lighting changes, but their intersection is characteristic of the gasket.

Once the gasket has been detected in each ROI, the algorithm computes the most central point within the corresponding candidate region (the centroid of the largest connected component). This yields two "safe" points lying well inside the gasket, one on the left side and one on the right side of the image, minimizing the risk of sampling near boundaries, metallic parts, or noisy regions. These two safe points are then used as prompt points for SAM on the gasket side, providing stable and reliable anchors for the

subsequent segmentation step, even under moderate variations in illumination, viewpoint, or vertical shifts.

2.3 Defect Detection Using Vision-Language Models (GPT-4o)

The second approach evaluated in this study leverages a Vision-Language Model (VLM), specifically `gpt-4o`, to perform the same defect detection task addressed by the computer vision pipeline. A VLM is a model that jointly processes images and text, allowing it to "reason" about visual content in response to natural-language instructions. The goal is to assess whether a general-purpose multimodal model, without any task-specific training, can reliably discriminate between correctly positioned gaskets and defective ones through visual reasoning guided by prompt instructions. This is fundamentally different from the computer vision pipeline: instead of engineering an explicit geometric measurement, the task is reframed as an *in-context, few-shot* classification problem, in which the model is shown a handful of labelled examples directly in the prompt and asked to classify a new image by analogy, without any parameter update or fine-tuning. A general overview of `gpt-4o` and of the Vision-Language Model concepts relevant to this work is provided in Appendix C.

The VLM is provided with:

- a set of reference images belonging to the `OK` class,
- a set of reference images belonging to the `DEFECT` class,
- a target image to classify,
- an instructional prompt describing the task and the application context.

Its task is to classify the target image as `OK` or `DEFECT`. The appeal of this approach lies in its minimal development cost: if a general-purpose model can solve the task competently from a few examples and a well-written prompt, it sidesteps the segmentation, calibration, and tuning effort required by the engineered pipeline.

2.3.1 Experimental Variables

To systematically study the behaviour of the VLM, two factors are varied, corresponding to the two main levers available to a practitioner using such a model out of the box: how many examples it is shown, and how much guidance it is given.

1. the number of reference images per class;
2. the quality and richness of the prompt.

Four configurations are considered for the number of references: 1, 2, 4, and 6 images per class (for both `OK` and `DEFECT`). Increasing the number of references gives the model more evidence about the natural variability of each class, but also consumes more of the context and increases the cost of each query; studying several values makes it possible to observe how much the model actually benefits from additional examples. For each configuration, references are randomly sampled multiple times (specifically, 70 different reference selections, as detailed in Section 3) to avoid selection bias, so that the reported behaviour reflects the configuration rather than a particularly favourable or unfavourable choice of examples.

Three prompt types are designed, differing in length, contextual detail, and semantic richness:

LOW-quality prompt. A concise description of the task, without domain-specific context (76 lines, 201 words). Example: *"TASK: You are given reference images belonging to two classes. Your goal is to analyze a target image and decide if it is OK or DEFECT based on references."*

MEDIUM-quality prompt. Extends the LOW prompt by adding domain context (the gasket on a metallic frame, the relevance of alignment, and the nature of the reference images; 91 lines, 333 words). Example: *"CONTEXT: sealing gasket mounted on a metallic tractor frame. All black masked regions are background, not part of the object."*

HIGH-quality prompt. Includes everything from MEDIUM and, additionally, a detailed description for each reference image (130 lines, 1021 words). Example: *"Description: The gasket shows a defect on the left side of the image: it is curved upward excessively compared to the correct position on the metal frame."*

The HIGH-quality prompt is the only one containing explicit semantic annotations for the reference examples, effectively teaching the model *why* each reference belongs to its class rather than merely showing it the label. This progression, from a bare task statement, to domain context, to fully annotated examples, is designed to isolate the contribution of prompt engineering to the model's performance. The performance differences across these configurations are analyzed in the following chapters.

3 Solution results

This chapter reports and analyses the results obtained by the two approaches introduced in Section 2: the computer vision-based pipeline and the Vision-Language Model (VLM) based on GPT-4o. To make the two solutions directly comparable, both are assessed with the same family of classification metrics: precision, recall, and F1-score, formally defined in Section 3.2.1. The defect class (`DEFECT`) is treated as the positive class throughout, since the primary objective of the inspection system is to detect defective gaskets. It is worth stressing from the outset that, in this industrial setting, the two error types are *not* equally costly: a false negative (a defective gasket that passes inspection) is far more critical than a false positive (a correct gasket flagged for re-checking), because the former can propagate a sealing fault into the finished product, whereas the latter only triggers a quick manual verification.

3.1 Computer vision-based approach result

3.1.1 Test Protocol

To evaluate this approach, a subset of the images produced by the operators was held out as an independent test set, disjoint from the samples used to build the *OK gap* envelope described in Section 2. The test set consists of 64 images, balanced between the two classes: 31 correctly positioned gaskets (`OK`) and 33 defective ones (`DEFECT`). Each test image is processed by the full pipeline (segmentation, polyline extraction, and deviation computation against the OK envelope) and is summarized by a single scalar, the *defect ratio*, i.e. the fraction of gasket points falling outside the acceptance envelope. A sample is classified as `DEFECT` when its defect ratio exceeds a decision threshold, and as `OK` otherwise.

3.1.2 Class Separation and Threshold Selection

The results show a clear separation between the two classes in terms of defect ratio. All 31 `OK` test samples were classified below a defect ratio of 33.9%; more precisely, excluding a single outlier at 33.9%, every remaining `OK` sample exhibited a defect ratio below 20%. Conversely, all 33 `DEFECT` samples exhibited a defect ratio above 20%, with the least severe defective case reaching 28.95%.

This distribution reveals a wide separation margin between the bulk of the two classes: the non-outlier `OK` samples all lie below 20%, while the defective samples all lie above 28.95%, leaving an empty band in between. Consequently, any decision threshold chosen in the interval [20%, 28.95%) yields exactly the same classification outcome on this test set: one false positive (the `OK` outlier) and zero false negatives. Such a broad, stable

operating range is a strong indicator of robustness, since the final performance does not hinge on a delicately tuned value. The threshold is therefore fixed at 20%, a conservative choice that keeps a comfortable safety margin with respect to the least severe defect (28.95%).

The only misclassification, the `OK` outlier at 33.9%, is consistent with the desired behaviour of the system: it is a false positive, not a false negative. In other words, the single mistake made by the pipeline is the "safe" kind of error, in which a correct gasket is flagged for re-inspection rather than a defective one being let through.

3.1.3 Quantitative Results

Applying the 20% threshold to the test set produces the confusion matrix reported in Table 1 and the per-class metrics reported in Table 2.

	Predicted OK	**Predicted DEFECT**
Actual OK	30	1
Actual DEFECT	0	33

Table 1: Confusion matrix of the computer vision pipeline on the test set of 64 images, using a defect-ratio threshold of 20%.

Class	**Precision**	**Recall**	**F1-score**	**Support**
`DEFECT` (positive)	0.97	1.00	0.98	33
`OK` (negative)	1.00	0.97	0.98	31
Overall accuracy		0.984 (63/64)		

Table 2: Per-class performance of the computer vision pipeline on the test set, at a defect-ratio threshold of 20%.

With the `DEFECT` class as positive, the pipeline attains a **recall of** 1.00 (all 33 defective gaskets are detected, i.e. no false negatives) and a **precision of** 0.97 (33 of the 34 samples flagged as defective are truly defective), for an **F1-score of** 0.98 and an overall accuracy of 0.984. The perfect recall is the most important outcome for the intended application, as it means that, on this test set, no defective gasket would reach the roof-assembly stage undetected.

3.1.4 Discussion

The strength of this approach lies in the combination of high accuracy, a wide and stable decision margin, and full interpretability: because the verdict is derived from an explicit

geometric measurement, every decision can be traced back to the specific regions of the gasket that fall outside the acceptance envelope, and it can be visualized directly on the image through the green/red overlay presented in Section 2. This transparency is a decisive advantage in an industrial quality-control context, where an operator must be able to understand and trust the reason behind a rejection.

The main caveats concern the size and nature of the evaluation. The test set, although balanced and independent, is relatively small (64 images), and the single false positive suggests that rare, atypical `OK` configurations can still be pushed above the threshold. Nevertheless, the clear class separation and the conservative error direction (false positives rather than false negatives) make the pipeline well suited to the target task.

3.2 Impact of Prompting and Visual Context in Vision-Language Models

In this section we analyze the impact of prompt quality and visual context on the behaviour of GPT-4o for the gasket defect detection task introduced in Section 2.

The experiments are organized around the configurations described in the technological perspective: three prompt quality levels (LOW, MEDIUM, HIGH) and four values for the number of reference images per class (1, 2, 4, 6). For each combination, the evaluation is repeated over 70 different reference-image selections, and the reported metrics correspond to averages across all 70 runs. This repeated sampling is essential: because the reference images are provided in-context and act as informal examples of each class, the specific set of references chosen for a given run can influence the outcome, and averaging over 70 selections yields a value representative of the typical behaviour rather than of a lucky or unlucky draw.

The VLM is evaluated on exactly the same test set used to assess the computer vision pipeline, namely the 64 held-out images (31 `OK` and 33 `DEFECT`) described in Section 3.1, so that the two approaches are compared on a fully common ground.

3.2.1 Evaluation Metrics

For each configuration we compute the following metrics:

- **Precision**, defined as the fraction of predicted `DEFECT` samples that are actually defective;
- **Recall**, defined as the fraction of truly defective samples that are correctly detected;
- **F1-score**, defined as the harmonic mean of precision and recall.

Denoting by TP, FP, and FN respectively the number of true positives, false positives, and false negatives (with `DEFECT` as the positive class), these quantities are given by

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}, \qquad \text{F1} = 2\,\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

These metrics allow us to characterize whether the model tends to *miss defects* (low recall) or to *over-detect* them (low precision), which is particularly relevant in an industrial quality-control scenario. In the context of gasket inspection, a missed defect (*false negative*) may lead to a faulty assembly passing the inspection stage, whereas an excessive number of false alarms (*false positives*) can increase rework and manual verification costs. Precision and recall explicitly separate these two aspects: model's ability to detect all defective samples and the reliability of positive predictions.

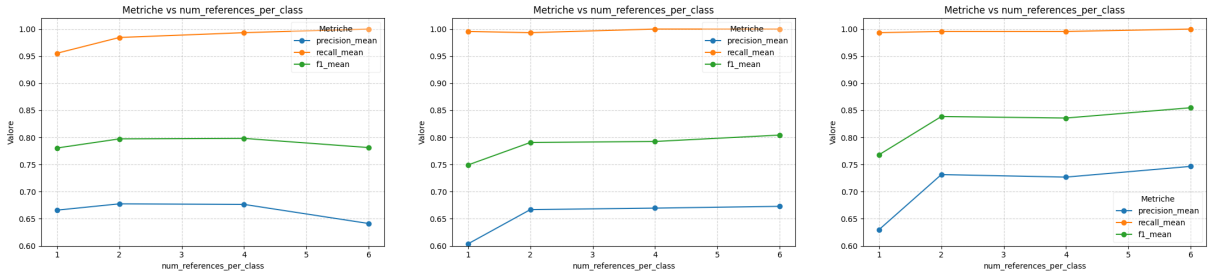
The **F1-score** provides a single scalar summary that balances both effects. Being the harmonic mean of precision and recall, it takes high values only when *both* are sufficiently high, and drops rapidly if either false negatives or false positives dominate.

In contrast, simpler metrics such as overall accuracy would be less informative in this setting. Accuracy aggregates all correct predictions into a single value and may hide asymmetric behaviours, for example configurations that achieve very high recall at the cost of a large number of false positives.

3.2.2 Results for Different Prompt Qualities

The impact of prompt quality and number of reference images is summarized in three plots, one for each prompt level (LOW, MEDIUM, HIGH). For every configuration, metrics are averaged over the 70 runs with different selections of reference images, so that the reported values are indicative of the typical behaviour of the model rather than of a specific choice of references.

The analysis focuses on how precision, recall, and F1-score evolve as the number of reference images per class increases, and on how this evolution changes across prompt qualities. A recurring pattern emerges across all three prompt levels: recall saturates almost immediately at its maximum value, so that precision (and hence F1-score) becomes the quantity that actually discriminates between configurations.



(a) Performance of GPT-4o for different numbers of reference images per class under a LOW-quality prompt. Recall quickly saturates at its maximum value, while precision and F1-score remain relatively stable, indicating a persistent tendency towards false-positive defect predictions.

(b) Performance of GPT-4o for different numbers of reference images per class under a MEDIUM-quality prompt. The addition of domain-specific context produces a modest improvement in precision and F1-score, while recall remains effectively saturated across all configurations.

(c) Performance of GPT-4o for different numbers of reference images per class under a HIGH-quality prompt. Detailed descriptions of the reference images substantially improve precision and F1-score as the number of references increases. Perfect recall is maintained across all configs.

Figure 5: Comparison of GPT-4o performance across LOW, MEDIUM, and HIGH prompt qualities as a function of the number of reference images per class. While recall remains consistently close to or equal to 1.00, precision and F1-score increasingly benefit from richer prompt descriptions and larger annotated reference sets, with the HIGH-quality configuration achieving the strongest overall performance.

LOW Prompt Quality

Observations.

- Recall is high across all settings, rising from ≈0.95 with a single reference per class to 1.00 at six references. The increase is gradual: two references already yield ≈0.98, but perfect recall is reached only at the largest setting.
- Precision is the distinctive trend. After a negligible bump between one and two references, it declines as more examples are added, falling from ≈0.68 at two references to ≈0.64 at six. Overall, additional references do not help precision: they worsen it.
- The F1-score follows precision. It peaks near 0.80 at two-four references and then decreases at six, remaining in the narrow band 0.78-0.80.

Interpretation. With a LOW-quality prompt, the model receives only a minimal task description and no explicit domain context. In this setting, GPT-4o adopts a conservative defect-detection policy: when uncertain, it classifies borderline cases as `DEFECT`. This behaviour explains the combination of:

- very high, and still increasing, recall (almost all truly defective samples are detected);
- low precision with a *decreasing* trend in the number of references (more `DEFECT` predictions are false alarms as extra images are added).

The model thus behaves like an inspector who "does not want to miss any defect", even at the cost of flagging several `OK` gaskets as defective. Without instructions, extra reference images reinforce this bias rather than correcting it: recall is pushed toward 1.00, while precision deteriorates. At six references the drop is explicit. Unguided visual examples add noise, not a decision boundary. The bottleneck is the lack of task specification; more images make the over-flagging worse.

MEDIUM Prompt Quality

Observations.

- Recall is effectively saturated from the first reference onward, remaining in the range 0.99-1.00 with negligible variation.
- Precision shows the opposite behaviour to the LOW setting: it *increases* with the number of references, from ≈0.60 at one reference to ≈0.67 at two, and then continues a slight upward drift toward 0.67-0.68 at six. False positives are still frequent, but additional examples now improve precision instead of degrading it.
- The F1-score tracks this increasing precision: a jump from about 0.75 to 0.79 between one and two references, followed by a modest rise toward 0.80.

Interpretation. The MEDIUM-quality prompt augments the LOW prompt with domain-specific context: the task concerns a sealing gasket mounted on a metallic tractor frame, black-masked regions are background, and the references include both correct and defective placements. This information changes how extra images are used. In the LOW case, more references drive precision down; here they drive it up. Context makes the second (and subsequent) reference usable as a comparison, rather than as further fuel for over-flagging.

The remaining pattern is otherwise similar. The contrast with LOW is the main result: once the model is told *what* it is looking at, additional references start to help precision instead of hurting it. High-level context is still not enough for fine-grained geometric reasoning about gasket (frame alignment) without explicit criteria for *why* a configuration is `OK` or `DEFECT`, the gain remains limited and plateaus after the second reference, but the sign of the precision trend has reversed.

HIGH Prompt Quality

Observations.

- Recall remains consistently equal to 1.00 across all reference counts: all defective samples are detected.
- Precision improves significantly with more reference images:
  - 1 reference: 0.63,
  - 2 references: 0.73,
  - 4 references: 0.73,
  - 6 references: 0.75.
- The F1-score exhibits a clear positive trend, increasing from 0.77 (1 reference) to 0.85 (6 references).

Interpretation. The HIGH-quality prompt differs qualitatively from the previous two: in addition to the task description and domain context, it provides a detailed textual description for *each* reference image, explicitly explaining why a given example is considered `OK` or `DEFECT`. In practice, these descriptions act as in-context annotations that guide the model's reasoning process.

Under this richer prompt, GPT-4o still maintains perfect recall but now becomes notably more selective in its `DEFECT` predictions:

- precision increases by more than 10 percentage points when moving from 1 to 2 references;
- precision remains high and continues to grow slightly up to 0.75 with 6 references;
- the F1-score reaches its best values, indicating a more balanced trade-off between sensitivity and reliability.

The fact that precision and F1-score improve with both prompt richness and number of references highlights the importance of combining *semantic guidance* (through detailed descriptions) with *visual diversity* (through multiple reference images). The model is no longer forced to infer the notion of defect solely from visual similarity; instead, it can align its internal representation with verbal explanations that specify which geometric deviations matter and why. Moreover, the fact that precision is still rising at six references, rather than having reached a plateau, suggests that the HIGH configuration had not yet

exhausted its potential, and that providing more than six annotated references per class could push precision, and therefore F1-score, even higher.

Overall, the HIGH prompt configuration shows that GPT-4o can approach a more controlled and predictable behaviour on this industrial task, provided that sufficient effort is invested in prompt engineering and in the construction of well-annotated reference sets.

3.2.3 Uncertainty Analysis

In addition to average performance metrics, it is crucial to assess the reliability and stability of VLM predictions. Average precision and recall describe how well the model performs *on aggregate*, but they say little about how *consistent* the model is when the in-context references change, or about how much a single reference image can distort the outcome. To this end, two complementary analyses were performed:

1. prediction instability per test image;
2. error rate per reference image.

Both evaluations were carried out using the HIGH-quality prompt, which previously showed the best overall performance. All values were normalized by usage frequency to ensure fair comparison across images that appear in a different number of runs.

Prediction Instability Prediction instability measures how often the model changes its classification for the *same* test image across different runs, where each run uses a different set of reference images. Formally, let a test image x be evaluated over $R(x)$ runs, and let $n_{\texttt{OK}}(x)$ and $n_{\texttt{DEFECT}}(x)$ be the number of runs in which x is predicted as `OK` and `DEFECT`, respectively (so that $n_{\texttt{OK}}(x) + n_{\texttt{DEFECT}}(x) = R(x)$). The instability of x is defined as the fraction of runs disagreeing with the majority prediction,

$$\text{instability}(x) \;=\; \frac{\min\big(n_{\texttt{OK}}(x),\, n_{\texttt{DEFECT}}(x)\big)}{R(x)} \;\in\; [0,\, 0.5].$$

A value of 0 denotes a perfectly stable image (always classified the same way), whereas a value approaching 0.5 denotes an image that is essentially classified at random depending on the chosen references.

Observations.

- Instability affects only `OK` samples.
- `DEFECT` samples show zero instability: the model is consistently able to recognize defective cases.

- For some `OK` samples, instability reaches up to 43%, meaning that the same image is classified inconsistently depending on the chosen reference set.
- One test image remains perfectly stable (0.00 instability), whereas others exhibit moderate to significant variability.

Interpretation. The model displays a systematic tendency towards defect detection. When the set of reference images changes, the model occasionally shifts its internal threshold for identifying a defect, causing reclassification of borderline cases. The asymmetry is telling: since instability is confined to `OK` images and defective images are always recognized, the uncertainty of the model is entirely concentrated on the "over-detection" side. This confirms that false positives are the dominant error mode and that the decision boundary of the model is sensitive to the contextual examples it is shown.

Error Rate per Reference Image The reference image error rate quantifies how frequently a given reference image leads to misclassifications across all test instances in which it was used. Formally, let $U(g)$ be the set of all evaluations (run, target-image pairs) in which reference image g was included in the context, and let $m(g)$ be the number of such evaluations that resulted in a misclassification. The error rate of g is

$$\mathrm{err}(g) \;=\; \frac{m(g)}{|U(g)|} \;\in\; [0,\,1],$$

where the normalization by $|U(g)|$ accounts for the fact that different reference images are used a different number of times.

Observations.

- Error rates vary considerably across reference images.
- The best reference images yield error rates around 0.20.
- The worst reference images yield error rates up to 0.44.
- This corresponds to a relative difference of approximately 56% in performance solely due to reference selection.
- Both `OK` and `DEFECT` reference images may be highly informative or misleading, depending on their clarity and representativeness.

Interpretation. These results highlight the sensitivity of VLM-based inference to in-context examples. Reference images effectively act as soft labels and substantially influence the model's reasoning. Poorly chosen references can bias the model towards over-detection or under-detection of defects. Even with a HIGH-quality prompt, visual context remains a major source of variance. From a practical standpoint, this means that the reference set is itself a design parameter that must be curated with care: selecting a few clear, representative examples per class can matter as much as the wording of the prompt.

3.2.4 Additional Considerations

Two further considerations help to explain the behaviour observed above and to delimit the practical operating range of the VLM-based approach. The first concerns a structural difference between the two solutions in how the final verdict is produced; the second concerns an intrinsic limitation of large language models when the amount of in-context information grows too large.

Absence of an explicit decision threshold. A fundamental difference between the computer vision pipeline and the VLM lies in *how* the binary `OK/DEFECT` verdict is ultimately reached. In the computer vision pipeline, the model does not decide the class directly: it produces a continuous, interpretable quantity (the defect ratio introduced in Section 2) which is only *afterwards* compared against an explicit decision threshold (20%, as calibrated in Section 3.1) to yield the final verdict. This decoupling of measurement from decision is what allows the pipeline to *tolerate* a small amount of detected deviation: a sample in which, say, 1% of the gasket points fall outside the acceptance envelope is still classified as `OK`, because 1% lies well below the threshold. The threshold effectively encodes how much local imperfection is considered acceptable before a gasket is rejected, and it can be tuned to trade off precision against recall in a controlled, transparent way.

The VLM offers no equivalent mechanism. When the model is asked to return an `OK/DEFECT` label, it emits the verdict directly, without exposing an intermediate, calibrated score on which a tunable threshold could be applied. There is therefore no explicit "tolerance knob": the model itself decides, internally and opaquely, how much apparent deviation is enough to warrant a `DEFECT` label. We hypothezise that this is one of the main reasons behind the strong tendency of the VLM to over-predict `DEFECT` that emerged throughout the experiments (the systematically high recall and the comparatively low precision, together with the fact that prediction instability is concentrated entirely on `OK` samples): lacking an explicit tolerance, the model appears to flag a sample as defective as soon as it perceives even a faint hint of irregularity, rather than weighing

its severity against a calibrated acceptance criterion. In principle one could try to recover a threshold-like behaviour, for example by asking the model for a numerical defect score or a confidence value and thresholding *that*, but such a score would not be a direct, physically grounded measurement of the gasket geometry, and its reliability and calibration would themselves have to be validated. As used here, the VLM conflates measurement and decision into a single opaque step, which is precisely what the engineered pipeline keeps separate.

Context degradation with larger prompts and more references. The results in Section 3.2.2 show that richer prompts and more reference images generally improve performance, with the HIGH-quality configuration still gaining precision at six references. This upward trend, however, cannot be extrapolated indefinitely. Enlarging the prompt and adding more reference images both increase the length of the input that the model must process, and it is a well-documented property of large language models that their ability to exploit information does not scale gracefully with context length.

This phenomenon is commonly referred to as *context degradation* (also popularized under the name "context rot"), and one of its best-known and empirically characterized manifestations is the *"lost in the middle"* effect: models tend to make the best use of information placed at the very beginning or the very end of the input, while information located in the middle of a long context is used far less effectively. As the context grows, the relevant evidence, for instance a particular discriminative reference image or a key sentence of the prompt, is increasingly likely to be diluted among many other tokens and to receive less effective attention, so that the model's accuracy, consistency, and instruction-following can *degrade* rather than improve.

For the present task this implies a practical ceiling. Beyond a certain point, adding further reference images or lengthening the prompt is expected to yield diminishing returns and may eventually become counterproductive: the extra examples not only raise the monetary and latency cost of each query (every additional image and every additional word consumes part of the finite context window, as discussed in Appendix C), but can also *dilute* the most informative content and destabilize the model's decisions. The favourable trend observed up to six references should therefore be read as valid within the explored range, not as evidence that "more is always better". In a production deployment, the number of references and the length of the prompt would themselves need to be treated as parameters to be optimized: rich enough to guide the model, yet compact enough to avoid triggering context degradation.

3.2.5 Summary of Experimental Findings

Across all experiments:

- Recall is almost always equal to 1, indicating that GPT-4o rarely misses defective samples.
- Precision is the primary discriminator between prompt qualities: LOW and MEDIUM prompts saturate around 0.67, while HIGH prompts reach 0.75, with a trend that had not yet plateaued at six references and could improve further with more annotated examples per class.
- Increasing the number of reference images meaningfully improves performance only when combined with a HIGH-quality prompt; under LOW and MEDIUM prompts additional references bring little benefit.
- False positives are the dominant source of error, especially under LOW and MEDIUM prompts, and prediction stability is lower on `OK` images than on `DEFECT` ones.

Overall, GPT-4o is capable of performing defect detection reasonably well, but only when provided with detailed task descriptions and semantically rich reference annotations. Without such guidance, the model systematically overestimates defects.

3.3 Comparison of the Two Approaches

Table 3 summarizes the best result achieved by each approach, using the shared precision/recall/F1 metrics.

Approach	**Precision**	**Recall**	**F1-score**	**Accuracy**
Computer vision pipeline (threshold 20%)	0.97	1.00	0.98	0.98
GPT-4o VLM (HIGH prompt, 6 references/class)	0.75	1.00	0.85	0.83

Table 3: Best-configuration comparison between the two approaches on the same 64-image test set, with `DEFECT` as the positive class. The VLM accuracy is derived from its precision and recall on the common test set.

Both approaches achieve perfect recall, meaning that neither lets a defective gasket pass undetected on the evaluated data: exactly the behaviour required by the application. The decisive difference lies in *precision*: the computer vision pipeline reaches 0.97, producing only a single false positive, whereas the best VLM configuration stops at 0.75. On

the shared 64-image test set, a precision of 0.75 at perfect recall corresponds to roughly 11 false positives (against the single one of the computer vision pipeline), i.e. about a third of the 31 `OK` gaskets would be needlessly flagged for re-inspection. In terms of F1-score, the engineered pipeline (0.98) clearly outperforms the VLM (0.85), and the gap widens further on accuracy (0.98 versus 0.83).

Beyond the raw numbers, the two solutions differ in nature. The computer vision pipeline provides a precise, interpretable, and stable measurement, at the cost of a dedicated engineering effort (segmentation, envelope calibration, and automatic point selection). The VLM, on the other hand, requires no task-specific training and can be deployed almost entirely through prompt engineering, but it is more sensitive to the choice of references and to the richness of the prompt, and it exhibits a residual tendency to over-detect defects. These technical differences set the stage for the economic comparison developed in Section 4, where the development and operational costs of each solution are weighed against the performance reported here.

4 Economic Analysis: Economic Sustainability of Automated Defect Detection in Industrial Contexts

In this chapter we analyze the economic impact of the gasket defect detection task and compare three alternative ways of carrying out the inspection:

1. **manual inspection** performed by a human operator;
2. the **computer vision-based** pipeline;
3. the **VLM-based** solution built on GPT-4o.

The goal is not only to estimate the raw operational cost of each option, but also to understand *where* the cost is concentrated (upfront versus recurring), how it *scales* with the production volume, and how the classification errors reported in Section 3 translate into economic terms. As will become clear, comparing the three options on a single figure such as "cost per year" is misleading: the three alternatives have fundamentally different cost structures, and the most convenient choice depends on the production volume, the planning horizon, and a set of non-monetary factors such as data sovereignty and vendor dependency.

4.1 Cost Model and Assumptions

To compare the three options on a common ground, each solution is decomposed into two types of cost:

- **CAPEX** (capital expenditure / one-off cost): a fixed cost paid once, independent of how many gaskets are inspected. It includes development effort, hardware, and initial setup.
- **OPEX** (operating expenditure / recurring cost): a cost paid continuously, which typically scales with the production volume. It includes labour, cloud/API fees, energy, maintenance, and the cost of handling classification errors.

The two automated solutions share part of the physical infrastructure (the 4K camera and the privacy-masking setup described in Section 2) which is therefore treated as a *sunk cost*, common to both, and excluded from the differential comparison.

Throughout the chapter, the following production regime is assumed, consistent with the figures used for the VLM cost estimate: five inspections per hour, eight hours per day, five days per week, and four weeks per month, i.e.

$$V = 5 \times 8 \times 5 \times 4 \times 12 \approx 9{,}600 \text{ inspections per year.}$$

Some quantities (development effort, hardware price, error-handling time) cannot be measured exactly and are therefore introduced as *illustrative assumptions*, clearly stated in Table 4. The purpose is to expose the *structure* of the trade-off rather than to claim precise absolute figures; the sensitivity of the conclusions to these assumptions is discussed in Section 4.5.

Parameter	**Assumed value**
Annual inspection volume V	9,600 / year
Loaded operator cost (fully burdened)	€ 25 / hour
Manual re-check time per flagged image	1 minute
CV development (one-off engineering)	€ 20,000
CV inference hardware (edge GPU / industrial PC)	€ 3,000
CV annual maintenance and energy	€ 1,000 / year
VLM integration and prompt/reference engineering	€ 5,000

Table 4: Illustrative assumptions used in the economic comparison. Hard figures taken from external sources (operator salary, OpenAI pricing) are reported in the respective sections.

4.2 Manual Inspection by a Human Operator

The reference cost of manual inspection carried out by a dedicated human operator is estimated at approximately

€ 27,000 / year.

This figure is essentially pure OPEX: it recurs every year and is largely independent of the number of gaskets inspected, since the operator is paid for their time rather than per unit. Two important qualifications apply. First, a human employee with an annual cost of around € 27k would rarely be dedicated *exclusively* to gasket inspection; they would also perform other tasks within the production workflow. The € 27k figure should therefore be read as an *upper bound* on the potential savings of automation, rather than as a strict one-to-one replacement cost. Second, automation does not necessarily eliminate this cost but may *reallocate* it: the operator freed from the repetitive inspection task can be assigned to higher-value activities, which is itself an economic benefit (an opportunity gain) even when the headcount is unchanged.

Beyond cost, manual inspection has intrinsic limitations that are relevant to the comparison: it is subject to fatigue, attention drift, and inter-operator variability, and it does not naturally produce a digital, auditable record of each decision.

4.3 Cost of the VLM-Based Approach (OpenAI GPT-4o)

The operational cost of GPT-4o was estimated using the official pricing information provided by OpenAI[1]. According to the pricing calculator, the cost of processing a 4K image (3840×2160 px) is

$$P = €\,0.002763 \text{ per image (worst case).}$$

A single inference in the best-performing configuration (HIGH prompt, six references per class) requires a total of 13 images: six `OK` reference images, six `DEFECT` reference images, and one target image. The resulting cost per inference is therefore

$$13 \times 0.002763 = €\,0.035919 \text{ per inference.}$$

Propagating this figure through the production regime of Section 4.1 gives

$$0.035919 \times 5 = 0.179595\ €/h, \qquad 0.179595 \times 8 = 1.43676\ €/\text{day},$$

$$1.43676 \times 5 = 7.1838\ €/\text{week}, \qquad 7.1838 \times 4 = 28.7352\ €/\text{month},$$

$$28.7352 \times 12 = \mathbf{€\,344.82\ /\ year}.$$

This value reflects the worst-case inference cost and excludes expenses related to infrastructure, cloud services, portals, deployment, or maintenance. The distinctive feature of the VLM approach is that its cost is almost entirely *recurring and volume-proportional*: every additional inspection adds a small but non-zero API fee, whereas the upfront investment is limited to integration and prompt/reference engineering (assumed at €5,000). There is no task-specific training and no data annotation in the machine-learning sense, which keeps the initial deployment cost low.

The main recurring liabilities of this approach are the inference latency (approximately 25 seconds per image in our experiments, dependent on network and API conditions), the dependency on an external cloud provider and its pricing policy, and, crucially, the residual false positives, whose economic impact is quantified in Section 4.6.

4.4 Cost of the Computer Vision-Based Approach

The computer vision pipeline has the opposite cost profile to the VLM. Its cost is dominated by a *one-off* investment: the engineering effort required to design and tune the segmentation, the tolerance envelope, and the automatic point-selection algorithm (assumed at €20,000), plus the local inference hardware (an edge GPU or industrial PC,

[1] `https://openai.com/api/pricing/`

assumed at € 3,000). Once deployed, the pipeline runs *on-premises* and incurs an almost negligible marginal cost per inspection: no per-image API fee, only local energy consumption and periodic maintenance (assumed at € 1,000 per year combined).

In other words, the computer vision approach front-loads the cost. It is expensive to build but very cheap to run, and its recurring cost is essentially *independent of the production volume*. This makes it structurally attractive whenever the number of inspections is high or the system is expected to operate for many years, because the large upfront investment is amortized over an ever-growing number of essentially free inferences.

Additional economic advantages, harder to quantify but real, follow from the on-premises nature of the solution: it does not send images to an external cloud (a benefit for data sovereignty and for the sustainability footprint of each inference), it is not exposed to third-party pricing changes or service outages, and its low, predictable latency does not depend on network conditions.

4.5 Total Cost of Ownership and Break-Even Analysis

Because the three options concentrate their costs so differently, the fairest way to compare them is through the *Total Cost of Ownership* (TCO) over a fixed planning horizon. Table 5 reports the TCO over a three-year horizon, using the assumptions of Table 4 and the error-handling costs derived in Section 4.6.

Cost component	**Human**	**Computer vision**	**VLM**
One-off (CAPEX / development)	≈ 0	€ 23,000	€ 5,000
Recurring labour / year	€ 27,000	–	–
API or compute / year	–	$\approx$ € 50	€ 345
Maintenance / year	–	€ 1,000	$\approx$ € 0
Error verification / year	–	€ 63	€ 693
3-year TCO	€ 81,000	€ 26,339	€ 8,114

Table 5: Illustrative three-year Total Cost of Ownership for the three inspection strategies. The human figure is an upper bound, as the operator is typically not dedicated exclusively to this task.

Two conclusions emerge clearly. First, **both automated options dominate manual inspection** on cost: even the more expensive computer vision solution costs, over three years, roughly a third of the human option, and the VLM about a tenth. Against a dedicated operator, the VLM investment pays for itself in a matter of weeks and the computer vision investment in well under a year.

Second, the comparison *between the two automated options* is more subtle and depends

on volume and horizon. Their cost can be modelled as a fixed part plus a per-inspection marginal part:

$$C_{\mathrm{VLM}}(N) = 5{,}000 + m_{\mathrm{VLM}}\, N, \qquad C_{\mathrm{CV}}(N) = 23{,}000 + m_{\mathrm{CV}}\, N,$$

where N is the cumulative number of inspections and m the marginal cost per inspection. Using the API fee plus the expected error-verification cost of Section 4.6, the marginal costs are approximately $m_{\mathrm{VLM}} \approx €\,0.108$ and $m_{\mathrm{CV}} \approx €\,0.008$ per inspection. The break-even volume, at which the two solutions cost the same, is

$$N^{\star} = \frac{23{,}000 - 5{,}000}{m_{\mathrm{VLM}} - m_{\mathrm{CV}}} \approx \frac{18{,}000}{0.100} \approx 180{,}000 \text{ inspections},$$

i.e. on the order of *eighteen years* at the plant's current throughput of 9,600 inspections per year.

The interpretation is that, *at the volume and horizon of this specific application*, the VLM is not only cheaper to build but also cheaper in total, and the computer vision pipeline's near-zero marginal cost is not enough to overcome its higher upfront investment within a realistic planning window. The balance tips in favour of the computer vision solution only at substantially higher volumes, for instance if the same system were replicated across many stations or lines, or if its development cost were lower than assumed. This is the classical CAPEX-versus-OPEX trade-off: the VLM behaves like a "pay-as-you-go" service, the computer vision pipeline like a "buy-once" asset.

4.6 Incorporating the Cost of Errors

A purely operational cost comparison would ignore a decisive factor: the two automated solutions do not make the same number of mistakes, and mistakes have a price. This section connects the classification results of Section 3 to their economic consequences.

Recall that both automated approaches achieve perfect recall on the test set, so in expectation neither lets a defective gasket through: the dominant, and cheaper, error mode is the *false positive*, in which a correct gasket is flagged and must be re-checked. Assuming that each false positive triggers a manual verification of one minute by an operator costing €25 per fully burdened hour, the cost of a single false positive is

$$c_{\mathrm{FP}} = \frac{25}{60} \approx €\,0.42.$$

The two approaches differ sharply in how often this happens. On the shared 64-image test set, the computer vision pipeline produced 1 false positive (1.6% of all images), whereas the best VLM configuration produced about 11 (17.2%). Scaled to the annual

volume $V = 9{,}600$, this yields

$$\text{CV:}\quad 0.016 \times 9{,}600 \times 0.42 \approx €\,63/\text{year}, \qquad \text{VLM:}\quad 0.172 \times 9{,}600 \times 0.42 \approx €\,693/\text{year}.$$

This result is striking: for the VLM, the annual cost of *verifying its false alarms* (€ 693) is roughly *twice* its annual API cost (€ 345). In other words, the "true" recurring cost of the VLM is dominated not by the inference fee but by the downstream human effort its lower precision generates. The higher precision of the computer vision pipeline is therefore not merely a technical nicety; it is an economic asset that becomes more valuable as the inspection volume grows.

A second asymmetric component is the cost of a *false negative*: a defective gasket that passes inspection and propagates into the finished product, potentially causing sealing failures, warranty claims, or a recall. Although neither automated approach produced false negatives on the test set, the potential cost of a single such event is very high and highly non-linear. The perfect recall observed for both methods is thus economically reassuring, but the residual instability of the VLM on borderline cases (Section 3.2.3) represents a latent risk that a purely on-premises, deterministic pipeline does not carry to the same degree.

4.7 Three-Way Comparison Summary

Table 6 summarizes the three options along the economic and operational dimensions discussed above, combining monetary figures with the qualitative attributes that a purely numeric comparison would miss.

Dimension	**Human**	**Computer vision**	**VLM**
Upfront cost (CAPEX)	very low	high	low
Recurring cost (OPEX)	very high	very low	low-moderate
Marginal cost per inspection	n/a	negligible	€ 0.036 + errors
Scales with volume	yes (time)	no	yes
Latency	seconds	low (local)	~ 25 s (cloud)
Detection quality (F1)	variable	0.98	0.85
False positives (test set)	–	1/64	11/64
Interpretability	implicit	high	moderate
Data stays on-premises	yes	yes	no
Vendor / pricing dependency	no	no	yes
Consistency / auditability	low	high	moderate

Table 6: Three-way comparison of the inspection strategies across economic and operational dimensions. Detection metrics are taken from Section 3.

4.8 Discussion of Economic Sustainability

The analysis paints a coherent picture. Against manual inspection, **automation of any kind is economically compelling**: both the computer vision pipeline and the VLM reduce the direct inspection cost by roughly one to two orders of magnitude and pay back their investment quickly, while additionally offering consistency, auditability, and the possibility of reallocating the operator to higher-value tasks.

The choice *between* the two automated approaches is a genuine trade-off rather than a clear win for either.

When the VLM is preferable. The VLM is the more economical option at the volume and horizon of this application. It requires almost no upfront investment, no task-specific training, and can be deployed largely through prompt and reference engineering. It is therefore particularly attractive for pilots, low-to-moderate volumes, rapid prototyping, or contexts where the ability to change the task by simply editing a prompt outweighs raw precision. Its weaknesses are the recurring API cost, the dependence on an external provider (with the associated pricing, availability, and data-sovereignty risks), the higher latency, and the cost generated by its lower precision.

When the computer vision pipeline is preferable. The computer vision pipeline is the more sustainable option at scale and over long horizons. Its near-zero marginal cost, superior precision (and therefore lower error-handling cost), on-premises data handling, low latency, and independence from external services make it the natural choice for a mature, high-volume deployment where predictability and control are valued. Its cost is that it must be engineered and maintained, and that this upfront effort only pays off once the cumulative volume is large enough.

A hybrid perspective. Finally, the two approaches are not mutually exclusive. Because the computer vision pipeline yields an explicit, continuous defect ratio, most images fall comfortably inside the "clearly OK" or "clearly DEFECT" regions and can be decided locally at negligible cost; only the small fraction of borderline cases near the threshold need to be escalated to a VLM query or to a human operator for a second opinion. Such a design confines the more expensive, less precise judgement to the genuinely ambiguous cases, combining the low marginal cost and high precision of the computer vision pipeline with the flexibility of the VLM. It is a promising direction for maximizing the economic sustainability of automated defect detection in industrial contexts.

5 Conclusions

This thesis addressed the automatic detection of defects in the positioning of a sealing gasket on a tractor cabin frame, an inspection carried out immediately before the roof is mounted, when a misaligned gasket can still be corrected cheaply. The problem was tackled with two fundamentally different technological strategies: an engineered computer vision pipeline and a general-purpose Vision-Language Model (GPT-4o); designed, implemented, and then compared on a common test set from both a technical and an economic standpoint.

5.1 Summary of the Work

The **computer vision pipeline** combines a deep-learning segmentation model (SAM) with classical image-processing techniques. The gasket and the metal frame are segmented into binary masks, their upper boundaries are extracted as polylines, and the local deviation between the two curves is compared against an empirically learned "OK" tolerance envelope to produce the final verdict. A dedicated automatic point-selection algorithm, based on texture (Laplacian) and color (CIELAB) cues, makes the segmentation robust to the moderate acquisition shifts that occur on the production line.

The **VLM-based approach** reframes the same task as an in-context, few-shot classification problem, in which GPT-4o is guided only by a natural-language prompt and a small set of labelled reference images, without any task-specific training. A systematic protocol varied the number of reference images per class (1, 2, 4, 6) and the richness of the prompt (LOW, MEDIUM, HIGH), averaging each configuration over 70 different reference selections.

5.2 Main Findings

From a **technical** perspective, the engineered pipeline proved highly effective: on the 64-image test set it reached an F1-score of 0.98 (precision 0.97, recall 1.00), with a single false positive and no false negatives, and a wide, stable decision margin that makes the result robust to the exact choice of threshold. The VLM also achieved perfect recall, but its precision was consistently lower and improved only when the prompt was enriched with detailed per-image annotations and several reference images, reaching a best F1-score of 0.85 (HIGH prompt, six references per class). Two further observations characterize the VLM: recall saturates almost immediately, so precision is the discriminating quantity; and the model exhibits a systematic tendency to over-detect defects, with a residual prediction instability confined to the `OK` class and a marked sensitivity to which reference images are provided.

From an **economic** perspective, both automated solutions reduce the direct inspection cost by one to two orders of magnitude with respect to a dedicated human operator, and pay back their investment quickly. The comparison between the two automated options is, however, a genuine trade-off rather than a clear win for either: the computer vision pipeline is an upfront-heavy asset with a near-zero marginal cost, whereas the VLM is a pay-as-you-go service with a low fixed cost but a marginal cost that scales with volume. At the volume and horizon of this specific application the VLM is also the cheaper option overall, and the balance tips towards the computer vision pipeline only at much higher volumes or longer horizons. A notable result of the analysis is that the cost of *verifying* the VLM's false positives can exceed its raw inference cost, so that the superior precision of the computer vision pipeline is not merely a technical merit but an economic asset.

5.3 Limitations

Several limitations should be acknowledged. The evaluation was performed on a relatively small test set (64 images), which, although balanced and independent for the computer vision envelope, is limited in absolute size. Because the number of available defective samples is small, the reference images used by the VLM are drawn from the same pool as the test targets (with the target itself excluded from its own references), which is common practice for few-shot evaluation but is less strictly separated than the disjoint train/test split used for the computer vision envelope. Finally, the economic analysis relies on a number of illustrative assumptions (development effort, hardware price, error-handling time); these were chosen to expose the *structure* of the cost trade-off rather than to provide exact absolute figures, and the corresponding conclusions should be read accordingly.

5.4 Future Work

Several directions naturally extend this work. First, the two approaches could be combined into a **hybrid system**: since the computer vision pipeline produces an explicit, continuous defect ratio, the vast majority of images can be decided locally at negligible cost, and only the few borderline cases near the decision threshold need to be escalated, to a VLM query or to a human operator, for a second opinion, confining the more expensive and less precise judgement to genuinely ambiguous cases. Second, the experimental campaign could be broadened by testing **more than six reference images** per class, since the VLM's precision had not yet plateaued, and by evaluating on a larger, fully disjoint dataset to remove the reference/target overlap. Third, replacing the cloud-based GPT-4o with an **on-premises or open-weight VLM** would address the recurring cost, latency,

vendor-dependency, and data-sovereignty concerns identified in the economic analysis. Finally, a longer **in-production trial** would validate the robustness of both approaches under the full variability of real operating conditions and refine the economic estimates with measured, rather than assumed, values.

5.5 Closing Remarks

Overall, this work shows that a carefully engineered computer vision pipeline still offers the best combination of accuracy, interpretability, and long-term economic sustainability for this industrial inspection task, while a general-purpose Vision-Language Model provides a remarkably fast and low-effort route to a working solution. Rather than being competitors, the two paradigms are complementary, and their thoughtful combination appears to be the most promising way to build defect-detection systems that are at once accurate, flexible, and economically sustainable.

A The Segment Anything Model (SAM)

This appendix provides a general overview of the Segment Anything Model (SAM), the deep-learning segmentation model used in the computer vision pipeline described in Section 2. The goal is not to reproduce the full technical detail of the original work, but to give the reader enough background to understand *why* SAM was chosen and *how* it is prompted in this thesis.

A.1 Motivation and Key Idea

Traditional segmentation networks are trained to recognize a fixed, predefined set of classes: a model trained to segment "cars" and "pedestrians" cannot segment a "gasket" without being retrained on a dedicated, pixel-wise annotated dataset. Building such a dataset for an industrial task is expensive and slow, since every training image must be manually labelled at the pixel level.

SAM takes a different approach. It is a *promptable* segmentation model: instead of predicting a fixed set of classes, it receives an image together with a *prompt* that indicates *which* object should be segmented, and it returns the corresponding mask. Having been trained on an extremely large and diverse dataset (billions of masks over millions of images), SAM generalizes in a *zero-shot* manner to objects, materials, and domains never seen during training. This is precisely the property exploited in this work: the gasket and the metal frame can be segmented without collecting and annotating a task-specific training set.

A.2 Architecture at a Glance

SAM is composed of three main components:

Image encoder. A heavy, high-capacity vision transformer that processes the input image *once* and produces a dense image embedding. This is the most computationally expensive step, but it is performed a single time per image, independently of the prompts.

Prompt encoder. A lightweight module that converts the user prompts (points, boxes, or a coarse mask) into embeddings.

Mask decoder. A fast, lightweight decoder that combines the image embedding with the prompt embeddings and outputs the segmentation mask(s).

Because the expensive image embedding is computed only once, SAM can respond to *several different prompts on the same image* very efficiently: only the cheap prompt

encoder and mask decoder need to be re-run. This design is convenient in the present pipeline, where multiple prompt points are used on the same acquired frame.

A.3 Types of Prompts

SAM accepts several kinds of prompts, which can also be combined:

- **Point prompts.** One or more points, each labelled as *foreground* (the point lies on the object to be segmented) or *background* (the point lies on something that must be *excluded* from the mask).
- **Box prompts.** A bounding box that roughly encloses the target object.
- **Mask prompts.** A coarse, low-resolution mask that provides an approximate initial segmentation to be refined.

In this thesis, only **point prompts** are used, since they offer a simple and controllable way to indicate the gasket and the frame.

A.4 Effect of the Number of Points

A central practical aspect of SAM is that the *number* and *placement* of the point prompts directly influence the resulting mask.

One point. A single foreground point is the minimal prompt. It is sufficient to indicate "segment the object located here", but it is inherently *ambiguous*: a single click on a gasket could reasonably refer to the whole gasket, to a local portion of it, or to a larger structure that contains it. To cope with this ambiguity, SAM can return *multiple candidate masks* for a single point (typically corresponding to different plausible interpretations, e.g. part / whole / super-structure), each accompanied by a confidence score. With only one point, however, there is no guarantee that the highest-scoring mask covers the *full extent* of an elongated object.

Two points. Adding a second foreground point on the same object provides additional spatial evidence and substantially reduces ambiguity. For an elongated object such as the gasket, placing one point towards the left and one towards the right encourages SAM to segment the object *along its whole length*, rather than only a local patch around a single click. This is exactly the strategy adopted in this work: two points on the gasket and two on the frame.

More points and background points. Additional foreground points can be added to further stabilize the mask or to force the inclusion of regions that a smaller prompt would miss. Conversely, *background* points can be used to *exclude* regions that SAM would otherwise incorporate: for example, a background point placed on the metal frame can prevent it from being merged into the gasket mask. In general, more (well-placed) points yield a more constrained and more predictable segmentation, at the cost of a slightly more complex prompting logic.

A.5 Prompting Strategy Adopted in This Thesis

Based on the considerations above, the pipeline uses **four foreground point prompts in total**: two on the gasket (left and right extremes) and two on the frame (left and right extremes). This choice was validated empirically: two points per object are enough for SAM to segment the full horizontal extent of each elongated structure while preserving the boundaries, which are the geometric feature of interest for defect detection. The four points are not placed at fixed image coordinates; they are relocated automatically for each acquisition by the point-selection algorithm described in Section 2, which keeps the prompts on the correct objects even under the moderate acquisition shifts that occur on the production line.

B Classical Computer Vision Techniques for Automatic Point Selection

This appendix provides a detailed treatment of the two classical computer vision techniques used to automatically localize the gasket and place the SAM prompt points described in Section 2: *texture analysis via the Laplacian operator* and *color analysis in the CIELAB color space*. The goal is to give the reader enough theoretical and practical background to understand *why* these two cues were chosen, *how* they are computed, and *how* their combination yields a robust localization of the gasket even under the moderate illumination and viewpoint variations that occur on the production line.

Both techniques are examples of *classical* (i.e. non-learned, hand-crafted) computer vision: unlike the deep-learning segmentation performed by SAM, they rely on explicit, interpretable image-processing operations whose behaviour can be reasoned about analytically. This interpretability is a desirable property in an industrial setting, where the behaviour of the point-selection stage must be predictable and easy to debug.

B.1 The Role of Point Selection in the Pipeline

Before describing the techniques themselves, it is worth recalling the problem they solve. SAM requires point prompts that lie *on* the object to be segmented. If a prompt point were to fall on the background or on the wrong structure (for example on the metal frame instead of the gasket), the resulting mask would be incorrect, and every downstream geometric measurement would be compromised.

Because the tractor is not always stopped at exactly the same position, the gasket does not project onto fixed pixel coordinates from one acquisition to the next. Prompt points placed at hard-coded image locations would therefore be unreliable. The purpose of the point-selection algorithm is to *find the gasket in each new image* and place the prompt points at a safe interior location, so that SAM always receives valid prompts.

The algorithm restricts the search to two predefined regions of interest (ROIs) (a left ROI and a right ROI along the frame) within which the gasket is expected to appear. Inside each ROI, the gasket is distinguished from its surroundings by exploiting two independent physical cues:

- its **texture**: the gasket is a porous, sponge-like rubber profile whose surface exhibits rapid local intensity variations, in contrast to the comparatively smooth metallic frame;
- its **color**: the gasket material has a characteristic color that differs from the surrounding metal.

Each cue is turned into a binary mask; the two masks are then intersected, and the centroid of the largest resulting connected component is used as the "safe" prompt point. The remainder of this appendix explains each cue in detail.

B.2 Texture Analysis via the Laplacian Operator

B.2.1 Texture as a Discriminative Cue

Texture refers to the spatial pattern of intensity variations on a surface. Intuitively, a smooth, uniform surface (such as polished metal) varies slowly across neighbouring pixels, whereas a rough, granular surface (such as porous rubber) varies rapidly, producing many small local "bumps" in intensity. In the frequency domain, a smooth surface is dominated by *low spatial frequencies*, while a rough surface contains substantial *high spatial frequency* energy.

The gasket's sponge-like surface is rich in high-frequency content, whereas the metal frame is comparatively low-frequency. A filter that responds strongly to high spatial frequencies and weakly to smooth regions is therefore an effective detector of the gasket. The *Laplacian operator* is exactly such a filter.

B.2.2 The Laplacian as a Second-Order Derivative

For a continuous two-dimensional image intensity function $I(x, y)$, the Laplacian is defined as the sum of the unmixed second partial derivatives:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}.$$

The first derivative of the intensity measures the *rate of change* (the local slope) of the image; it is large across edges and small in flat regions. The second derivative measures the *rate of change of the rate of change*, in other words, how quickly the slope itself is changing. This makes the Laplacian an *isotropic* (rotation-invariant) measure of local curvature in the intensity surface: it responds strongly wherever the intensity bends sharply, regardless of the orientation of that variation.

The key property for our purposes is that the Laplacian:

- produces a response close to *zero* in smooth regions, where the intensity varies slowly and almost linearly (the second derivative of a linear ramp is zero);
- produces a *large-magnitude* response (positive or negative) in regions of rapid, non-linear intensity change, such as fine texture and edges.

Being isotropic, a single Laplacian filter captures variation in all directions simultaneously, unlike first-order gradient operators (such as Sobel) that must be applied separately along x and y. This is convenient for texture detection, where the intensity fluctuations of a granular surface have no preferred orientation.

B.2.3 Discrete Approximation

Digital images are sampled on a discrete pixel grid, so the continuous derivatives are approximated by finite differences implemented as a convolution with a small kernel. A common 3×3 discrete Laplacian kernel is

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

which encodes the finite-difference approximation

$$\nabla^2 I(x, y) \approx I(x{+}1, y) + I(x{-}1, y) + I(x, y{+}1) + I(x, y{-}1) - 4\, I(x, y).$$

An eight-connected variant, which also includes the diagonal neighbours, is

$$K_8 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

In both cases the kernel weights sum to zero, which guarantees that a perfectly uniform region (constant intensity) produces a response of exactly zero: the operator is insensitive to the absolute brightness level and reacts only to *variations*.

In the implementation, this convolution is performed by the OpenCV routine

$$L = \text{cv2.Laplacian}(I, \text{CV_64F}),$$

where I is a greyscale, contrast-enhanced version of the ROI. The output type `CV_64F` (a 64-bit floating-point image) is important: the Laplacian response is signed, negative on one side of a change and positive on the other, and can exceed the range of the original 8-bit image. Storing the result in a floating-point buffer prevents the clipping and truncation of negative values that would occur if an 8-bit integer type were used, thereby preserving the full response and avoiding a systematic loss of information.

B.2.4 From Laplacian Response to a Texture Measure

The raw Laplacian response L is a signed image. Because we are interested in the *amount* of local variation rather than its sign, the response is squared:

$$\text{variance} = L^2.$$

Squaring makes the measure non-negative and, crucially, *amplifies* strong responses relative to weak ones: a pixel with a large Laplacian magnitude (typical of textured rubber) is boosted much more than a pixel with a small magnitude (typical of the near-uniform metal frame). This increases the contrast between the textured gasket and its smooth surroundings, making the subsequent thresholding more reliable.

The name "variance" reflects the close relationship between this quantity and the classical *variance of the Laplacian*, a well-known scalar measure of image sharpness/focus: an image (or region) with a high Laplacian variance is rich in high-frequency detail, while a blurred or smooth one has low Laplacian variance. Here the same idea is applied *locally*, per pixel, to build a spatial map of "texturedness" rather than a single global number.

B.2.5 Thresholding into a Binary Texture Mask

Finally, the per-pixel texture measure is converted into a binary mask by comparing it against a texture threshold τ_{texture}:

$$M_{\text{texture}}(x, y) = \begin{cases} 1 & \text{if variance}(x, y) > \tau_{\text{texture}}, \\ 0 & \text{otherwise.} \end{cases}$$

Pixels whose local variation exceeds the threshold are retained (labelled as candidate gasket), while smoother pixels are discarded. The result is a binary texture mask that highlights the regions where the intensity fluctuates rapidly (predominantly the porous rubber gasket) while largely suppressing the smoother, metallic areas of the frame.

The value of τ_{texture} controls the sensitivity of the detector: a low threshold retains more pixels (higher recall but more false positives from noise or minor surface irregularities on the metal), whereas a high threshold is more selective (fewer false positives but a risk of missing weakly textured parts of the gasket). It is chosen empirically so that the mask reliably covers the gasket without being overwhelmed by spurious responses.

B.2.6 Practical Considerations

Two practical points deserve emphasis. First, because the Laplacian is a second-order derivative, it is inherently *sensitive to noise*: sensor noise also produces rapid local inten-

sity changes and can therefore generate spurious high responses. Contrast enhancement and, where appropriate, mild smoothing prior to the Laplacian help ensure that the operator responds to genuine surface texture rather than to acquisition noise. (The combination of Gaussian smoothing followed by a Laplacian is the classical *Laplacian-of-Gaussian*, which trades a little spatial resolution for substantially improved noise robustness.)

Second, texture alone is *not* a sufficient cue. Other rough or highly detailed regions in the ROI (dirt, scratches, machined patterns, or reflections on the metal) can also produce a strong Laplacian response and would be wrongly retained by the texture mask. This is precisely why a second, independent cue based on color is introduced, as described next.

B.3 Color Analysis in the CIELAB color Space

B.3.1 Why Not Work in RGB?

Digital images are usually stored in the RGB color space, in which each pixel is described by the intensities of its red, green, and blue components. RGB is convenient for display but poorly suited to *measuring* color similarity, for two reasons:

- **It is not perceptually uniform.** A fixed numerical distance between two RGB triples does not correspond to a fixed perceived difference in color: the same Euclidean step can be visually negligible in one part of the space and very noticeable in another.
- **It entangles color and brightness.** All three channels vary together with illumination, so a change in lighting shifts all coordinates at once, making it hard to separate "what color is this?" from "how brightly is it lit?".

For a task that asks "is this pixel the same color as the gasket material?", a color space in which numerical distance matches perceived difference is far more appropriate. The CIELAB space was designed for exactly this purpose.

B.3.2 The CIELAB Color Space

CIELAB (also written $L^*a^*b^*$), standardized by the Commission Internationale de l'Éclairage (CIE), represents each color with three coordinates:

L^* — **Lightness.** Ranges from 0 (black) to 100 (diffuse white). It encodes how light or dark the color is, independently of its hue.

a^* — **Green-Red axis.** Negative values tend towards green, positive values towards red/magenta.

b^* — **Blue-Yellow axis.** Negative values tend towards blue, positive values towards yellow.

Two properties make CIELAB especially useful here. First, it is *approximately perceptually uniform*: Euclidean distances in $L^*a^*b^*$ correspond closely to differences perceived by the human eye, so a single distance threshold behaves consistently across the whole space. Second, it *separates lightness from chromaticity*: the L^* channel carries the brightness information, while the a^* and b^* channels carry the color (chromatic) information. This separation is valuable for robustness to illumination: if the lighting on the line changes slightly, most of the effect is absorbed by the L^* channel, while the chromatic coordinates a^* and b^*, which describe the intrinsic color of the gasket material, remain comparatively stable.

B.3.3 Measuring Color Similarity: the ΔE Distance

The perceptual difference between two colors in CIELAB is quantified by the *color-difference* metric ΔE. In its simplest (CIE76) form it is the Euclidean distance between the two $L^*a^*b^*$ triples. Given a pixel color $p = (L_p^*, a_p^*, b_p^*)$ and a reference color $r = (L_r^*, a_r^*, b_r^*)$,

$$\Delta E(p, r) = \sqrt{(L_p^* - L_r^*)^2 + (a_p^* - a_r^*)^2 + (b_p^* - b_r^*)^2}.$$

A small ΔE means the two colors are perceptually similar; a large ΔE means they are perceptually distinct. Because CIELAB is approximately perceptually uniform, this single scalar behaves like a meaningful "perceived color distance", which is exactly what is needed to decide whether a pixel matches the gasket material.

If robustness to lightness variation is desired, the same distance can be computed using only the chromatic channels,

$$\Delta E_{ab}(p, r) = \sqrt{(a_p^* - a_r^*)^2 + (b_p^* - b_r^*)^2},$$

which discards the L^* term and therefore ignores differences that are purely due to brightness. (More elaborate, perceptually refined formulas such as CIE94 and CIEDE2000 exist and further improve uniformity, but the Euclidean CIE76 distance is sufficient and computationally cheap for this application.)

B.3.4 Building the Color Mask

The color cue is computed as follows. One or more reference color patches, representative of the gasket material, are selected in advance (a single average reference color, or several,

to cover natural variation in the material and its appearance under the station's lighting). Each pixel in the ROI is converted to CIELAB, and its color distance ΔE to the reference (or to the nearest of several references) is evaluated. The pixel is then classified as a gasket candidate if this distance falls within a predefined tolerance τ_{color}:

$$M_{\text{color}}(x, y) = \begin{cases} 1 & \text{if } \min_r \Delta E\big(p(x, y),\, r\big) \leq \tau_{\text{color}}, \\ 0 & \text{otherwise.} \end{cases}$$

The tolerance τ_{color} defines the radius of an acceptance region around the reference color(s) in CIELAB space: a larger tolerance accepts a broader range of shades (higher recall, but a risk of admitting similarly colored background), while a smaller tolerance is stricter (fewer false positives, but a risk of rejecting genuine gasket pixels whose color has drifted, e.g. due to shadows). As with the texture threshold, its value is chosen empirically.

B.3.5 Strengths and Limitations of the Color Cue

The color cue is complementary to the texture cue. Its main strength is that it captures a property, the specific material color of the gasket, that is largely independent of surface roughness, and, when computed in CIELAB (especially on the chromatic channels), it is relatively robust to moderate lighting changes. Its main limitation is the mirror image of the texture cue's limitation: color alone can be fooled by any background object that happens to share the gasket's color, and, despite the partial invariance offered by CIELAB, strong shadows, highlights, or color casts can still shift a pixel's coordinates enough to cross the tolerance boundary. Neither cue is individually reliable; their combination is what makes the detection robust.

B.4 Fusion of the Two Cues and Safe Point Selection

The two binary masks encode independent pieces of evidence: the texture mask marks "rough" regions, and the color mask marks "gasket-colored" regions. The final candidate region is obtained by taking their intersection (logical AND):

$$M_{\text{gasket}}(x, y) = M_{\text{texture}}(x, y) \;\wedge\; M_{\text{color}}(x, y).$$

A pixel is retained only if it is *both* textured *and* of the right color. This intersection is the crux of the method's robustness: the false positives of the two cues are largely uncorrelated. A smooth metal region that happens to match the gasket color is rejected by the texture mask; a rough metallic scratch or reflection is rejected by the color mask. Only regions that satisfy both characteristic properties of the gasket (its sponge-like micro-

texture *and* its material color) survive the intersection.

This fusion is performed independently within the left ROI and the right ROI. Within each ROI, the intersected mask may still contain small spurious fragments, so the algorithm isolates the *largest connected component*, which corresponds to the main body of the gasket in that region. The prompt point is then taken as the *centroid* of this largest component: the most central interior point of the detected gasket region.

Choosing the centroid, rather than an arbitrary or boundary pixel, is deliberate: it yields a "safe" point that lies well inside the gasket, as far as possible from its edges, from the metal frame, and from noisy border regions. This minimizes the risk that the SAM prompt accidentally lands on the wrong structure. The procedure produces two such safe points, one in the left ROI and one in the right ROI, which serve as stable, reliable anchors for the SAM segmentation step, even under moderate variations in illumination, viewpoint, or vertical shift of the acquired image.

B.5 Summary

The automatic point-selection stage combines two classical, interpretable computer vision cues. The Laplacian operator provides a texture measure that responds to the gasket's rough, high-frequency surface while ignoring the smooth metal; the CIELAB color analysis provides a perceptually meaningful, illumination-robust measure of color similarity to the gasket material. Each cue is individually fallible, but their intersection is highly specific to the gasket, and the centroid of the resulting region yields a robust interior prompt point. This engineered, transparent pre-processing step is what allows the downstream deep-learning segmentation (SAM) to be prompted reliably on every acquisition, despite the geometric variability inherent to a manual, production-line setting.

C The GPT-4o Vision-Language Model

This appendix provides a general overview of `gpt-4o`, the Vision-Language Model (VLM) used in the second solution described in Section 2. Because the field of large multimodal models evolves very rapidly, many readers may be unfamiliar with this specific model, or may know only the more recent systems that have since superseded it. The goal here is therefore twofold: to place `gpt-4o` in its historical and technical context, and to explain the capabilities that are actually exploited in this thesis, in particular, its ability to jointly reason over images and text and to perform *in-context, few-shot* classification without any task-specific training.

C.1 Historical Context

`gpt-4o` (the "o" stands for *omni*) is a multimodal model released by OpenAI in May 2024. It belongs to the GPT-4 family, which began with the original GPT-4 in early 2023, but it introduced a significant architectural change: whereas earlier systems handled different modalities through separate, loosely coupled models (for example, a text model paired with a distinct image-understanding component and a separate speech-to-text front end), `gpt-4o` was designed as a *single* neural network trained end-to-end across text, vision, and audio.

At the time of the experiments described in this thesis, `gpt-4o` was OpenAI's flagship general-purpose multimodal model, offering a strong combination of reasoning ability, image understanding, response speed, and cost. It is worth stressing, for the reader more familiar with later systems, that the model landscape has since moved on: subsequent releases (including reasoning-oriented models and further "omni" iterations) have improved on `gpt-4o` along several axes. The choice of `gpt-4o` in this work reflects the state of the art *at the time of the study* and, importantly, a model that was widely available, well documented, and representative of the class of general-purpose VLMs a practitioner would realistically reach for.

C.2 What Is a Vision-Language Model?

A *Vision-Language Model* is a neural network that jointly processes visual input (images) and textual input (natural language) within a single model, so that it can answer questions about images, describe them, compare them, or reason about their content in response to written instructions. This is in contrast to a traditional computer vision classifier, which is trained to output one of a fixed set of labels and cannot be re-tasked through language.

The defining capability exploited in this thesis is that a VLM can be given, in the same request, *both* a set of images *and* a textual prompt describing what to do with them, and it

will produce a textual answer that takes both into account. This is precisely what makes it possible to reframe the gasket inspection as a language-guided visual classification task: the model is shown reference images and a target image, told what "OK" and "DEFECT" mean, and asked to return a verdict, all without any change to the model's parameters.

C.3 Architecture at a Glance

Like the rest of the GPT family, `gpt-4o` is built on the *Transformer* architecture and is fundamentally an autoregressive next-token predictor: given a sequence of input *tokens*, it predicts the most likely continuation, one token at a time. The essential ideas relevant to this work are the following.

Tokenization of text and images. The model does not operate on raw characters or raw pixels but on *tokens*. Text is split into sub-word tokens; images are divided into patches and encoded into a sequence of visual tokens that live in the same representation space as text tokens. As a consequence, an image and its accompanying instructions are processed as a single, unified sequence, allowing the model's attention mechanism to relate words to image regions and vice versa.

Unified multimodal training. Because `gpt-4o` was trained end-to-end over interleaved text and images (and audio), it learns cross-modal associations directly, rather than bolting a vision module onto a pre-existing text model. This tends to produce more coherent joint reasoning over mixed image-and-text inputs.

Autoregressive generation. The output, for instance, the string `OK` or `DEFECT`, possibly accompanied by a short justification, is generated token by token, each token conditioned on the input and on the tokens produced so far.

For the purposes of this thesis, the internal details of the architecture are less important than one practical consequence: everything the model "sees", every reference image, the target image, and every word of the prompt, is converted into tokens and consumed from a finite *context window*. This directly ties the number of reference images and the length of the prompt to the cost and feasibility of each query, a point that is central to the experimental design.

C.4 In-Context Learning and Few-Shot Prompting

The single most important capability exploited in this work is *in-context learning*: the ability of a large model to adapt its behaviour to a task purely from examples and instructions supplied *in the prompt*, without any gradient update or fine-tuning of its weights.

The model's parameters remain frozen; the "learning" happens only within the current request, as the model conditions its answer on the examples it has just been shown.

Two related terms recur throughout the thesis:

Zero-shot. The model is asked to perform the task from instructions alone, with no worked examples.

Few-shot. The model is additionally shown a small number of labelled examples (here, reference images labelled `OK` or `DEFECT`) and asked to classify a new, unlabelled target by analogy.

The gasket-inspection experiments are a few-shot setup: for each query, a handful of `OK` and `DEFECT` reference images are placed in the prompt alongside the target image, and the model is asked to decide the target's class. Increasing the number of references gives the model more evidence about the natural variability of each class, but also lengthens the input and therefore raises the per-query cost, which is exactly why the number of references is treated as an experimental variable.

C.5 The Role of the Prompt

Because the task is specified in natural language, the wording of the prompt is not a cosmetic detail but a genuine lever on performance: an activity commonly called *prompt engineering*. The same model, shown the same images, can behave quite differently depending on how clearly the task, the domain context, and the meaning of the labels are described. This is the motivation for the three prompt tiers (LOW, MEDIUM, HIGH) studied in this thesis: they progressively add domain context and explicit, per-example semantic annotations, making it possible to isolate how much of the model's accuracy comes from the images themselves versus from the guidance provided in the prompt.

It is worth noting that VLMs are also susceptible to well-known failure modes that make careful prompting necessary: they can *hallucinate* (produce confident but incorrect statements), they can be sensitive to superficial aspects of the input such as image ordering or phrasing, and their verdicts are not perfectly deterministic. Framing the task precisely, stating explicitly what should be ignored (for example, the black privacy-masked regions), and constraining the expected output format all help mitigate these issues.

C.6 Relevance to This Thesis

In summary, `gpt-4o` is used here as a representative, general-purpose Vision-Language Model that can be applied to the gasket-inspection task *out of the box*, with no task-specific

training. The properties that make this possible are its unified multimodal processing of images and text, its autoregressive generation of a textual verdict, and, above all, its capacity for in-context, few-shot learning driven entirely by the prompt. These same properties also define the practical trade-offs analyzed in the following chapters: the model requires essentially no development effort and no annotated training set, but each inference consumes a token budget that scales with the number of reference images and the richness of the prompt, and its behaviour must be steered and validated through careful prompt design rather than through explicit engineering of a measurement.

References

[1] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 4015-4026. Also available as arXiv:2304.02643.

[2] OpenAI, "Hello GPT-4o," 2024. [Online]. Available: `https://openai.com/index/hello-gpt-4o/`

[3] OpenAI, "GPT-4o System Card," 2024. arXiv:2410.21276.

[4] OpenAI, "GPT-4 Technical Report," 2023. arXiv:2303.08774.

[5] OpenAI, "API Pricing," 2024. [Online]. Available: `https://openai.com/api/pricing/`

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, *et al.*, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 1877-1901.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, *et al.*, "Learning Transferable Visual Models From Natural Language Supervision," in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021, pp. 8748-8763.

[9] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 12, pp. 157-173, 2024. Also available as arXiv:2307.03172.

[10] K. Hong, A. Troynikov, and J. Huber (Chroma Research), "Context Rot: How Increasing Input Tokens Impacts LLM Performance," Technical Report, 2025. [Online]. Available: `https://research.trychroma.com/context-rot`

[11] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY, USA: Pearson, 2018.

[12] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, vol. 25, no. 11, pp. 120-125, 2000.

[13] D. Marr and E. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 207, no. 1167, pp. 187-217, 1980.

[14] I. Sobel and G. Feldman, "A 3×3 Isotropic Gradient Operator for Image Processing," presented at the Stanford Artificial Intelligence Project (SAIL), 1968.

[15] International Commission on Illumination (CIE), *Colorimetry - Part 4: CIE 1976 L*a*b* Colour Space*, CIE Standard S 014-4/E:2013 (ISO/CIE 11664-4:2019), Vienna, Austria, 2013.

[16] G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations," *Color Research & Application*, vol. 30, no. 1, pp. 21-30, 2005.

[17] M. Sokolova and G. Lapalme, "A Systematic Analysis of Performance Measures for Classification Tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427-437, 2009.

[18] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 (General Data Protection Regulation)," *Official Journal of the European Union*, L 119, pp. 1-88, 2016.