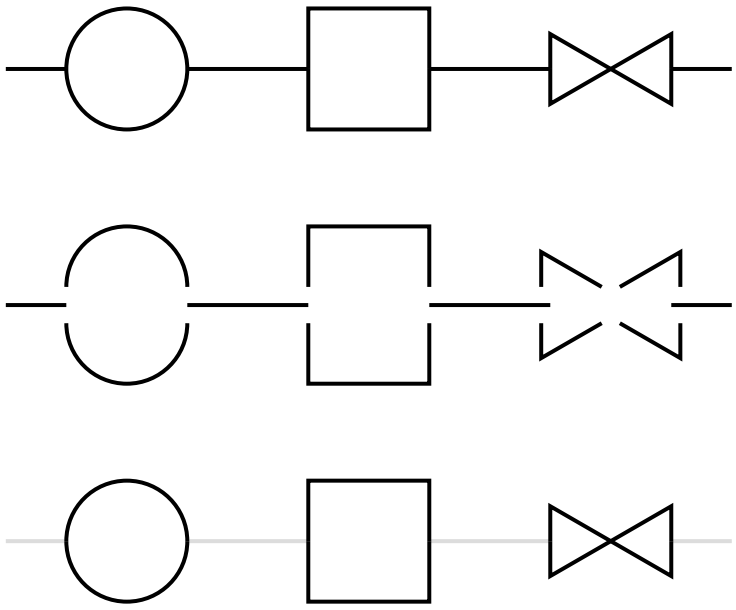
CHALMERS
UNIVERSITY OF TECHNOLOGY



Zero-Shot Detection and Classification of Symbols in P&IDs

Leveraging Vision-Language Models for Reference-Free Symbol Detection and Classification

Master's thesis in Data Science and AI

CARL ODQVIST

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Zero-Shot Detection and Classification of Symbols in P&IDs

Leveraging Vision-Language Models for Reference-Free Symbol Detection and Classification

CARL ODQVIST

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Zero-Shot Detection and Classification of Symbols in P&IDs
Leveraging Vision-Language Models for Reference-Free Symbol Detection and Classification
CARL ODQVIST

Supervisor and Examiner: Ida Häggström, Department of Electrical Engineering
Supervisor: Ola Löseth, Actemium Energy AI

Master's Thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An illustration of how symbols can be localized by investigating segments derived from a schematic.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Zero-Shot Detection and Classification of Symbols in P&IDs
Leveraging Vision-Language Models for Reference-Free Symbol Detection and Classification
Carl Odqvist
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Piping and instrumentation diagrams (P&IDs) are a type of industrial schematic used to represent complex industrial processes. Digitization involves extracting information from a P&ID, such as locations and symbol interpretations. This thesis demonstrates how a vision-language model (VLM) can be leveraged for domain-specific zero-shot detection and classification of symbols.

A problem with training models is that symbols vary between projects, combined with a scarcity of annotated training data from real-world projects. Training a model on insufficient data would limit its ability to generalize. For this reason, VLMs are leveraged to provide general knowledge, enabling zero-shot detection and classification without requiring any training data.

The thesis presents a novel approach for transforming the detection of symbols into a classification task performed by a VLM. For symbol classification, the pipeline matches a detected symbol with one of the possible symbol categories. The symbol classifier must be provided with a list of possible categories.

The digitization pipeline showed strong performance despite no training and no examples given. The proposed digitization pipeline provides an adaptable solution where symbol variability is high and annotated data is scarce.

Keywords: zero-shot, vision-language model, piping and instrumentation diagram, industrial schematics, object detection, visual prompt engineering, digitization.

Acknowledgements

I would like to express my gratitude to all who have supported me throughout the project. A special thanks to Tomas Grahn, Erik Sahlin, and Ola Löseth at Actemium Energy AI. This thesis would not have been possible without your guidance, valuable feedback, and insightful discussions. I would also like to thank my examiner and academic supervisor, Ida Häggström, for her helpful advice.

Carl Odqvist, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CoT	Chain-of-Thought
IoU	Intersection over Union
LLM	Large Language Model
LSTM	Long Short-Term Memory
MoE	Mixture of Experts
NLP	Natural Language Processing
OCR	Optical Character Recognition
P&ID	Piping and Instrumentation Diagram
VLM	Vision-Language Model

Contents

List of Figures

List of Tables

1
Introduction

Piping and Instrumentation Diagrams (P&IDs) is a type of industrial schematic for representing instrumentation, piping and control devices in industrial processes. Although there are standards such as ISO 14617 and ISO 10628, in practice, the symbols used in industry vary between engineering projects [2]. The variety of symbols, text and nonstandard formatting makes digitizing industrial schematics a complex task [3].

Because the exact appearance of a symbol is not always known in advance, there is a need for reference-free symbol detection. With reference-free referring to symbol detectors that have no knowledge of which symbols they are tasked to identify or their appearance. Humans can still recognize these symbols by relying on general structural patterns, geometry, and context rather than matching them to a specific reference symbol. But the human process is time-consuming and prone to errors [3, 4].

Recent advances in Vision-Language Models (VLMs) make it possible for models to answer complex questions about visual content [5]. VLMs possesses general knowledge and general problem solving ability, reducing the need for task specific training. Some recent VLMs enables understanding of visual inputs such as geometric shapes and diagrams, allowing them to extract relevant information from documents [6].

One previously proposed digitization pipeline employed multiple for detection, each detection different features, and multiple methods for comprehension of the detected features [7]. Another digitization pipeline interprets schematics in a single step with a deep learning model [8]. There are also digitization pipelines combining rule-based heuristics and Optical character recognition (OCR) techniques in combination with deep learning models [9, 10].

1.1 Motivation

One challenge is the lack of annotated real-world datasets and the class imbalances in the existing ones [3], leading some researchers to create synthetic datasets [7]. But modern VLMs are built for downstream applications [11] and have been used for analysis of industrial schematics [2]. By using VLMs instead of training a model, higher adaptability is achieved. Adapting to new symbols without retraining the model drastically simplifies project specific adaptations [2]. This thesis is motivated

by the limited amount of available training data and the promising potential of VLMs within the domain of P&IDs.

1.2 Objective

The objective of this thesis is to develop an approach for identifying symbols in P&IDs without relying on a predefined symbol list with images of the symbols. The purpose of the thesis is to investigate techniques that can extract and interpret symbols based on visual and contextual cues, supporting use cases where symbol variability is high. Specifically, the aims of the thesis are:

1. Development of a zero-shot detector for P&ID symbols.
2. Development of a zero-shot classifier for P&ID symbols.
3. Insights into the strengths and weaknesses of the detector and classifier.

1.3 Scope

P&IDs consist of symbols, text, and connections. The scope of this thesis is limited to the detection and classification of symbols. Connections which includes pipes, pipe junctions and arrows indicating flow direction, are excluded from the study. Text can belong to a symbol and are sometimes located inside a symbol, but the thesis will not match texts to symbols. The scope of the thesis does not include deriving information about how the symbols relate to each other.

In this thesis, zero-shot refers to the ability of the model to provide an answer without any visual examples of schematics or symbols. There will be no training or fine-tuning of VLMs. Throughout the thesis, the model relies solely on its pretrained knowledge of the model and information from the prompt.

2
Background

2.1 P&ID Digitization

Piping and instrumentation diagrams are technical drawings for a graphical representations of complex industrial processes. They are widely used in the design and planning of manufacturing processes [12]. Digitization of P&IDs refers to the process of extracting information from a P&ID in image format to a structured format that is interpretable for computers [4]. The types of data extracted can be divided into symbols, text and connectors with some research only targeting one of them while others extracts all three [3].

2.1.1 Data Scarcity

Due to the lack of annotated data, previous research has investigated techniques that do not require training such as circle and line detection algorithms [12]. However, these techniques often require parameter tuning and limit the types of shapes that can be detected. Much research on P&ID digitization has been evaluated on proprietary datasets that are not open to the public, which limits research and complicates comparisons [3]. Other research has focused on the generation of synthetic P&IDs that mimic real-world schematics to train deep learning models, some of which have been turned into public datasets [7, 13].

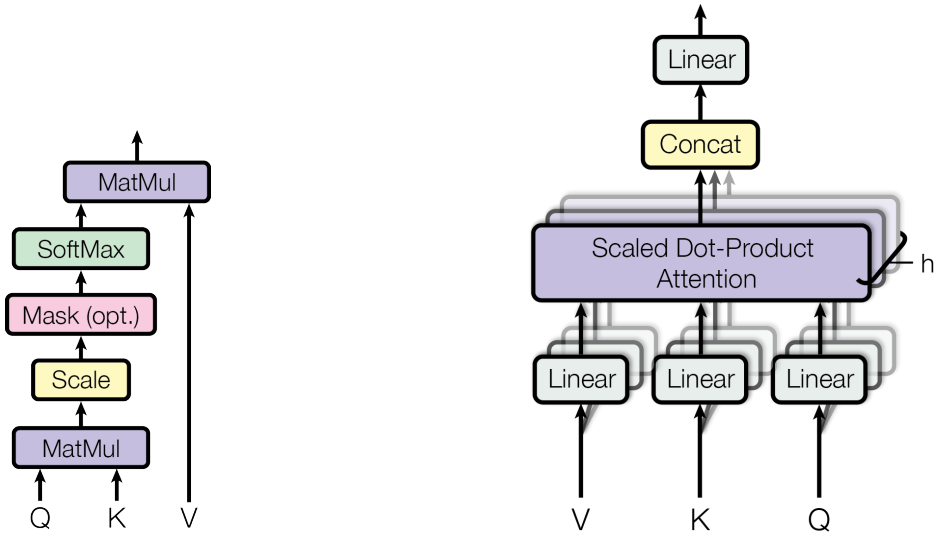
2.1.2 Model Architectures

An early approach for deep learning for P&ID digitization was based on image segmentation models used to detect symbols [10]. Essentially, determining for each pixel in an image whether it belongs to a symbol or not and if so, what type of symbol. In [10], one type of symbol characterized by a specific shape was detected algorithmically while other symbols were detected with the image segmentation model. Image segmentation models have also been used to find symbols of any category, then using a separate model to distinguish between the different types of symbols [7]. Another architecture extracts bounding boxes, their symbol type, and symbol connections simultaneously [13]. A bounding box is a rectangular boundary that encloses an object of interest. This is less detailed information than image segmentation, where the area of a symbol is described pixel-wise.

2.2 Vision-Language Models

Vision-language models are at the intersection of natural language processing (NLP) and computer vision [14]. NLP is a branch of artificial intelligence that concerns how human language can be understood, interpreted and generated by computers. Computer vision is the field of artificial intelligence regarding visual information. VLMs are models that process textual and visual inputs together. This section will cover how vision-language models have evolved for the foundational transformer architecture, which is based on the attention mechanism.

2.2.1 Attention Mechanism



(a) Attention **(b)** Multi-head attention.

Figure 2.1: Attention on the left and multi-head attention on the right. Reproduced from [1]

In the case of language translation, text is transformed into a sequence of tokens, where the tokens are vector representations of the meaning [15]. Similar words will have similar vector representations. At the core of the transformer architecture is the attention mechanism. One of the advantages of the attention mechanism is its ability to learn long range dependencies between tokens in a sequence [1]. Thus, the attention mechanism can better understand how words affect the meaning of other words in long sentences. The attention mechanism is visualized on the left in Figure 2.1 and expressed mathematically using the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

For each token, a query, key and value vector is derived using learned linear projections. The matrices Q, K, and V are sets of all the query, key and value vectors for

the entire sequence. The attention mechanism first computes the matrix of similarity scores QK^T, where each value in the matrix represents how each token learns from every other token [16]. The matrix of similarity contains the dot products for all pairs of key and query vectors. It is scaled with the square root of the dimension of the key vector $\sqrt{d_k}$ and normalized with a softmax function [1].

The resulting attention weights are then multiplied by the value matrix. The attention mechanism produces weighted sums of the value vectors, where the weights depend on the degree to which the keys and queries match [1]. This can be interpreted as the query vector representing what type of information a token is looking for, while the key vector represents what kind of information a token can provide, and the value vector contains the information it will share. The attention mechanism enables tokens to collect relevant information from other tokens [16]. To improve performance, multiple attention heads can be used in parallel, where the attention heads attend to different features [1]. Multi-head attention is visualized on the right in Figure 2.1 and scaled dot-product attention is just the standard attention expressed by the formula and on the left in Figure 2.1. Self-attention is an attention mechanism that relates different tokens within a sequence to produce a refined representation of that sequence [1].

2.2.2 Transformer Architecture

The transformer architecture is a deep learning architecture originally developed for improved sequence modeling which includes analysis, interpretation, and prediction of ordered data [1]. Compared to previous architectures, this allows for greater parallelization, which significantly reduces training time [1]. It established a new state-of-the-art for language translation where the input sequence is text in one language and the output sequence is the translated text in another language. Previous language models were based on models such as recurrent neural networks. One such model, called Long Short-Term Memory (LSTM), had previously been able to translate long sentences [15].

The encoder-decoder transformer architecture can be divided into two parts, the decoder and the encoder. In Figure 2.2, the encoder is on the left side and the decoder is on the right side. The input text is divided into tokens that are entered into the embedding layer, which turns every token into a corresponding vector. Positional encodings are added to the vector embeddings of the tokens so that each token contains information about its position in the sequence of tokens [1]. The encoder takes in the embedded input text, whereas the decoder takes in the embedded output text and the encoding from the encoder. The output text is tokens tokens that has already been generated. The output of the entire transformer architecture is the probabilities of how likely each token is to be the next token in a sequence. In the case of language translation, the model encodes a text in one language and predicts one new token at a time in a second language [1]. With each token corresponding to one or multiple text characters.

The encoder consists of multiple identical layers, with each layer having two sub-layers. The first sub layer is multi-head self-attention and the second is a fully connected feed-forward network that operates on each single token separately [1]. The encoder layer also include normalization and residual connections, as can be seen in Figure 2.2.

The decoder is similar to the encoder with the main difference being that each decoder layer has an extra multi-head attention sub-layer. It is so called encoder-decoder attention, where the queries come from the decoder sub-layer and the keys and values come from the encoder [1]. This lets tokens in the decoder to attend to the tokens in the input sequence. The first attention block in the decoder layers is self-attention over the current output tokens. For language translation, this enables the model to analyze both the text in the first and the text generated so far in the second language, when predicting the next token.

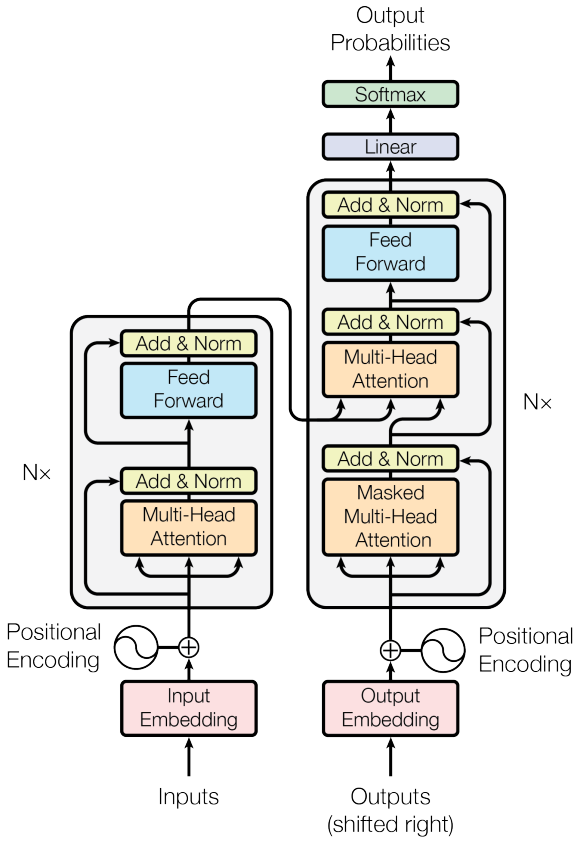


Figure 2.2: Encoder-decoder transformer architecture. Reproduced from [1].

There are three main variants of the transformer architecture with use cases far beyond language translation. An early model capable of task such as answering questions and comparing the meaning of sentences was a decoder-only transformer model [17]. This was achieved by first pre-training the model to predict the next word in books, leading to the model obtaining general information about the world and is able to handle long dependencies. The pre-trained model could then be fine-

tuned for specific tasks [17]. Encoder-only transformers can also be used for a wide array of downstream tasks [18]. The third variant is the encoder-decoder transformer described earlier, visualized in Figure 2.2.

2.2.3 Combining Computer Vision with Language Models

Early approaches to vision-language modeling focused on aligning corresponding images and texts within the same vector space. In an influential study, a separate image encoder and text encoder were trained so that semantically related image and text pairs are mapped to similar embeddings [19]. This model can be used for multiple tasks, including image classification by comparing different text descriptions with an image and using the best matching description as the prediction. However, this model could not be used to generate text. Instead, its input consists of both images and texts, with similarity scores as output.

As discussed in the previous section, the transformer architecture has contributed to the field of NLP with its ability to interpret long texts, leading to capable language models. But the transformer can be expanded beyond text inputs to handle images, audio, video and spatial information [20]. One approach was to add an attention layer that fuses together the vision tokens from a vision encoder with the text tokens in the language model [21]. This technique allows the language model to answer questions about images. A subsequent approach was to train a transformer model to extract the relevant information from a vision encoder, which is then used by an LLM [22]. Hence, neither the vision encoder nor the LLM needs to be trained, and only the bridging transformer model is trained. A more recent approach merges visual tokens from multiple layers in the vision encoder to multiple layers in the LLM. Providing high- and low-level visual information between the vision encoder and LLM [11]. VLMs can also be improved by instruction fine-tuning, which aims to teach a model to better follow instructions, leading to an improved ability to answer questions [23].

2.2.4 Text Generation and Temperature

As described earlier, the transformer model predicts only the probabilities of the next token in a sequence. Text generation is thus autoregressive, and at each step a token is chosen from the probability distribution. How the next token is sampled from the distribution differs and is controlled by variables such as top-k, top-p, and temperature [24].

Top-k means that the next token will be sampled from the k most probable tokens [24]. An alternative, top-p does not set a strict limit on the number of tokens to be considered but still samples from the most probable token. Instead, the limit is set such that the sum of their probabilities sum up to the probability p. Here, k and p are hyperparameters. Temperature affects the probability distribution itself. A low temperature skews the distribution towards the already more probable

tokens. Conversely, a higher temperature evens the probabilities [24]. Setting the temperature is a tradeoff between the quality of the generated text and its diversity [24].

2.2.5 Mixture of Experts

For neural networks, their learned information capacity is limited by the number of parameters [25]. But an increase in the number of parameters also lead to a rapid increase in computational cost [25]. The purpose of mixture of experts (MoE) is to disconnect this strong connection to allow for bigger and more efficient models. MoE layers have two main parts, one is the multiple feed-forward networks called experts. The other is the gating network that decides which experts will be used for a given input to the MoE layer. MoE layers only activates a small subset of the experts [25]. Hence, the size of the model and computational cost can be decoupled. This allows for significantly higher model capacity without a similar increase in computation during training and inference [25]. MoE can be applied to the feed-forward networks of the transformer architecture, resulting in more effective language models [26].

2.2.6 Model Distillation

Distillation is the process in which a teacher model is used to train a smaller student model, called the distilled model [27]. Typically, machine learning models performing classification are trained to achieve as high a probability as possible on the correct answer and a probability as close to zero for the wrong answers. However, if there are already models capable of a certain task, then a student model can be trained to output a probability distribution close to that of the capable model [27]. A student model can learn more useful information from the output distribution of a teacher model than training on the dataset alone [27].

Model distillation can be applied to LLMs and has brought about smaller and faster models with almost no loss in performance [28]. This is crucial for applications with hardware or time constraints. Model distillation increases the computational cost during the training phase, but makes inference more efficient, which is beneficial for large scale deployment [27]. A single general model can also be distilled into multiple task specific models [28].

2.3 Computer Vision Techniques

This section covers general techniques used for image processing and evaluation within the field of computer vision. This includes extracting text, visual features and matching model predictions with the ground truth in a dataset.

2.3.1 Optical Character Recognition (OCR)

Optical character recognition is software that identifies text in images and extracts characters into a digitized format [29]. The complexity of the task arises from the

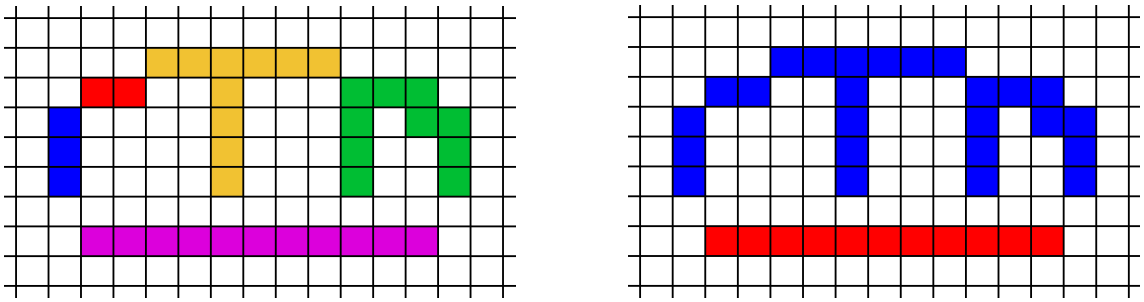
variety in fonts, styles, and languages [29]. OCR has been a research topic for decades with early models using geometric features and statistical methods [29]. Later research has been focused on deep learning techniques and in some cases natural language NLP for improved performance [30].

2.3.2 Image Skeletonization

Image skeletonization is a thinning algorithm that is applied to a binary image. The algorithms thins lines to a constant thickness while preserving connectivity between pixels [31]. The skeleton is the result of thinning and contains extracted features of the pattern in the original binary image [31].

2.3.3 Connected Component Labeling

For a binary image, a connected component is a collection of pixels that are connected to each other. There are two versions of connectivity. 4-connectivity requires pixels to be located to the right of, above, left of or below to be considered connected. 8-connectivity also accepts pixels diagonal connection. Connected component labeling is an algorithm that gives all pixels belonging to the same connected component the same label [32]. Connected component labeling is primarily used for pre- or post-processing step with use cases ranging from pre-processing training data to real-time applications [33]. Figure 2.3 illustrates the difference between the two types of connectivity.



(a) 4-connectivity. **(b)** 8-connectivity.

Figure 2.3: Examples where all pixels belonging to the same connected component has the same color, visualizing the difference between 4-connectivity and 8-connectivity.

2.3.4 Evaluation of Symbol Detection

When evaluating symbol detection, there are predictions and the ground truth. The number of predictions can be higher, lower or equal to the number of actual symbols in a P&ID. To assess the accuracy of the predictions, they need to be matched with the real symbols. In the case of symbol detection, there are two sets of bounding boxes. One set is the set of predictions, while the other is the set of ground truths. A predicted bounding box can be assigned to at most one bounding box in the

ground truth. Conversely, A bounding box from the ground truth can be assigned to at most one prediction. To measure how much two bounding boxes match, the intersection over union (IoU) can be used as a measurement [13]. The IoU is the ratio between the intersection area and the union area of the bounding boxes. These areas are visualized in Figure 2.4.

(a) Intersection area. **(b)** Union area.

Figure 2.4: Intersection area and union area of two bounding boxes.

$$\text{Intersection over Union} = \frac{\text{Intersection Area}}{\text{Union Area}} \tag{2.2}$$

A perfect overlap results in an IoU of one, whereas no overlap results in an IoU of zero. The pairing of predictions and ground truths is an assignment problem where the objective function is the sum of the IoU across all pairs. The assignment problem have been used to match the predictions in previous research about P&ID digitization [13]. In the literature, the standard criterion for a correct detection is that a pair of bounding boxes for prediction and ground truth has an intersection over union of at least 0.5 [34]. The threshold of 0.5 is also used for evaluation within the domain of P&ID digitization [13].

2.4 Capabilities and Applications of VLMs

This section covers the capabilities of VLMs and techniques related to the usage of language models. As mentioned earlier in Section 2.2.2, the transformer architecture can be trained to interpret text and images. But for some tasks, additional steps are required.

2.4.1 Prompt Engineering

In transitional supervised learning, a model is trained to predict a certain output for a given input, but for language models, there is another approach to adapt a model to a task [35]. Firstly, a language model is trained to predict the probability

of words by training on large amounts of raw text. Secondly, a prompt instructs the model so that the language model predicts the desired information. The prompt is changed depending on the task without the need to change the model. A prompt is an input to a LLM or VLM that guides the output of the models [36]. One type of prompt is textual templates that allow a language model to complete a task by predicting the most likely tokens. [35]. This makes LLMs and VLMs adaptable to new tasks with little or no labeled data [35].

Few-shot learning is to give a language model a few examples of questions with corresponding answers before answering a question [37]. Such examples demonstrate how the model should answer. Another strategy is zero-shot learning, where no examples are given. This can be done by training a model to predict the probability of a word and then letting it fill in a missing word in a sentence designed to answer the question of interest [35].

One type of answering format that few-shot learning can enforce is chain-of-thought (CoT) prompting [37]. It instructs the model to answer in multiple intermediate reasoning steps before giving the final answer, improving performance for more complex questions [37]. This approach can also be extended to zero-shot learning, including an instruction in the prompt to think step by step [38]. CoT instructs the model to lay out the models reasoning in text instead of just providing the answer directly.

Instead of forcing a model to reason in predefined steps, reasoning models can determine how to decompose a given problem [39]. An approach is to let a model solve a problem in steps, where it tries different paths and continues with the most promising ones. For this, a language model generates new reasoning paths, solves intermediate tasks, and evaluates its progress [39]. This kind of approach is suitable when deliberate reasoning is required for complex tasks where CoT struggles [39].

For VLMs, prompt engineering can be extended to visuals. Images can be edited to provide additional information related to the prompt, this technique is called visual prompt engineering [40]. In a scenario where part of an image is to be analyzed, the VLM can be guided by drawing a red circle around the area of interest instead of cropping the image. This directs the models attention while simultaneously maintaining information about the entire image [40]. This has been shown to deliver strong performance for zero-shot predictions and enables precise localizations in prompts [40].

2.4.2 VLMs for Problem Solving

Modern VLMs can handle a wide array of problems ranging from extracting textual information in images and interpreting diagrams to arithmetic. Solving tasks that require collage-level knowledge about multiple subjects [6]. The combination of visual understanding with language models performs well for classification tasks that demands outside knowledge and reasoning [41]. Tasks from various domains can be converted into a classification problem for the VLM by providing it with alternatives

[6]. This applies to problems where the answer alternatives are descriptions, numerical values or even formulas. Reasoning models can even play games and write code [5]. These capabilities are the basis for the expansion of downstream applications based on VLMs [11].

2.4.3 VLMs for P&ID Digitization

VLMs have recently been applied for P&ID digitization. One use case is to use VLMs to assess the quality of the predictions produced by a symbol detector and to refine the final result [42]. The factors used for determining quality are completeness, precision, localization and classification accuracy. In the refinement phase, a VLM is also instructed to extract the text tags of all symbols that are not already detected. The detected symbols are either marked with colored bounding boxes or masked out. This provides the regions where the detector missed symbols which can be used to improve the symbol detector, but it relies on the assumption that all symbols have text labels [42].

In another approach, a VLM was used for the detection of symbols [2]. The VLM was given two images, the first being an example image of a P&ID where the symbols of interest were marked by a red circle around them. The second image was the P&ID to be analyzed. The VLM was instructed to extract all text labels belonging to symbols in the second image if the same symbol could also be found among the marked symbols in the first image [2]. This also relies on the presence of text labels and does not provide any precise location, such as bounding boxes. unfortunately, this approach was considered inadequate for autonomous use, and thus is better used as a tool accompanied by human supervision [2].

3
Methodology

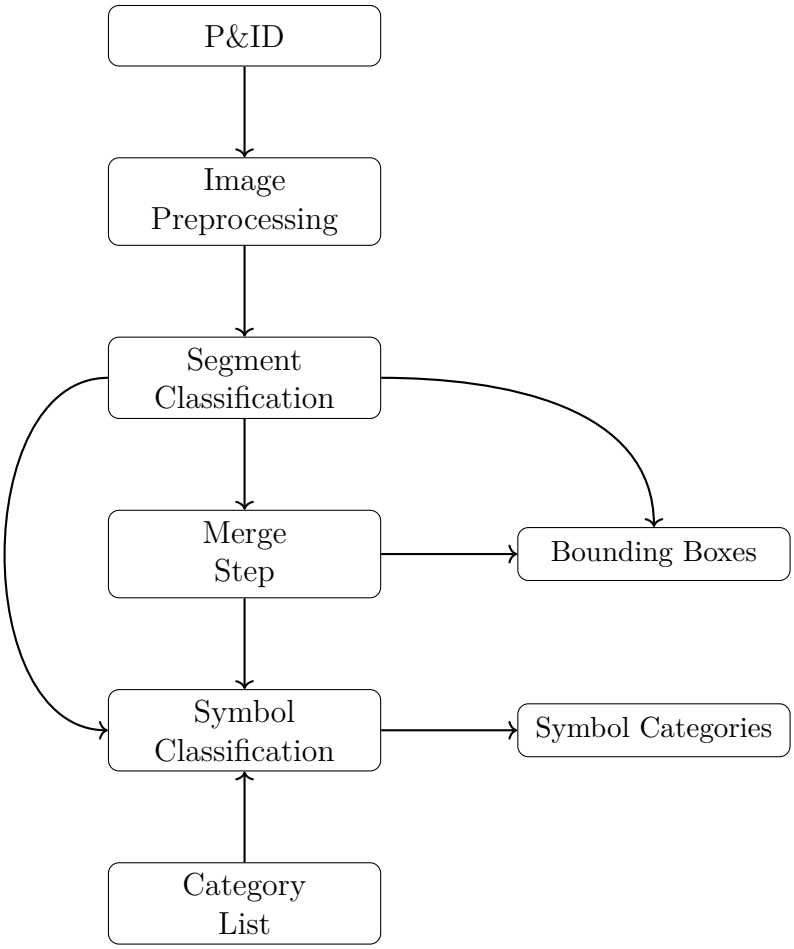


Figure 3.1: System Overview of the P&ID digitization pipeline.

The proposed digitization pipeline predicts where symbols are located and what type of symbols they are. The first step is image preprocessing of P&IDs. The preprocessed image is used in the second step for segment classification. The schematic is divided into small segments, each of which is classified as being part of a symbol or not. Adjacent symbol segments are joined into a symbol component, but the resulting symbol component might not be an entire symbol. For symbols that consist of two parts that are not adjacent, these two parts must be merged, which is done in the third step, the merge step. The third step is a refinement step of the second step.

Both the second and third step result in symbol components consisting of segments that belong to the same symbol. Bounding boxes are derived from symbol components and are the reason why both the second and third step produce bounding boxes, as seen in Figure 3.1. It is also the reason that the merging step is optional before the symbol classification step.

The classification step takes symbol components and classifies them as one of the possible categories. Thus, the classification step has to be provided a list of the symbol categories which is illustrated by the box at the bottom of Figure 3.1. Hence, the symbol detector (consisting of the segment classification step and optionally the merging step) acts without any information about which symbols are present in the P&ID but the symbol classifier relies on that information.

3.1 Dataset

The thesis relies on the three datasets PID2Graph OPEN100, PID2Graph Synthetic and Dataset-P&ID from [13]. PID2Graph OPEN100 consists of 12 P&IDs from the OPEN100 reactor project [43]. PID2Graph Synthetic as a synthetic dataset that was created by randomly placing and connecting symbols. The symbols being placed were cutouts documents containing P&ID symbols [13]. Dataset-P&ID is another synthetic dataset created due to the absence of any publicly available dataset [7]. Dataset-P&ID was later reconfigured to correspond to the symbol categories in the two PID2Graph datasets to allow for better comparisons between all three datasets [13], hereafter referred to as Dataset PID. These three datasets constitute the PID2Graph dataset, publicly available under the CC BY-SA 4.0 license [44].

Table 3.1: The evaluation subset and the entire dataset.

Dataset	Total Nr. P&IDs	Selected Nr. P&IDs
PID2Graph OPEN100	12	12
PID2Graph Synthetic	250	50
Dataset PID	500	50

The datasets are annotated with bounding boxes and labels for each bounding box, marking which symbol categories they belong to. The annotations also include the location of the legends and the scales around the schematics. For examples of the three datasets, see Figure 3.2, 3.3 and 3.4. The thesis relies on the general knowledge of pretrained VLMs without any training or fine tuning, the dataset is only used for evaluation. Thus, only a subset of the entire dataset is used, as seen in Table 3.1.

The symbol categories in PID2Graph OPEN100 are general, instrumentation, valve, tank, inlet & outlet and pump. PID2Graph Synthetic does not include instrumentation or inlet & outlet. Dataset PID does not include the tank, inlet & outlet, or pump. Table 3.2 lists the occurrence of each symbol category for the selected subset of the three datasets.

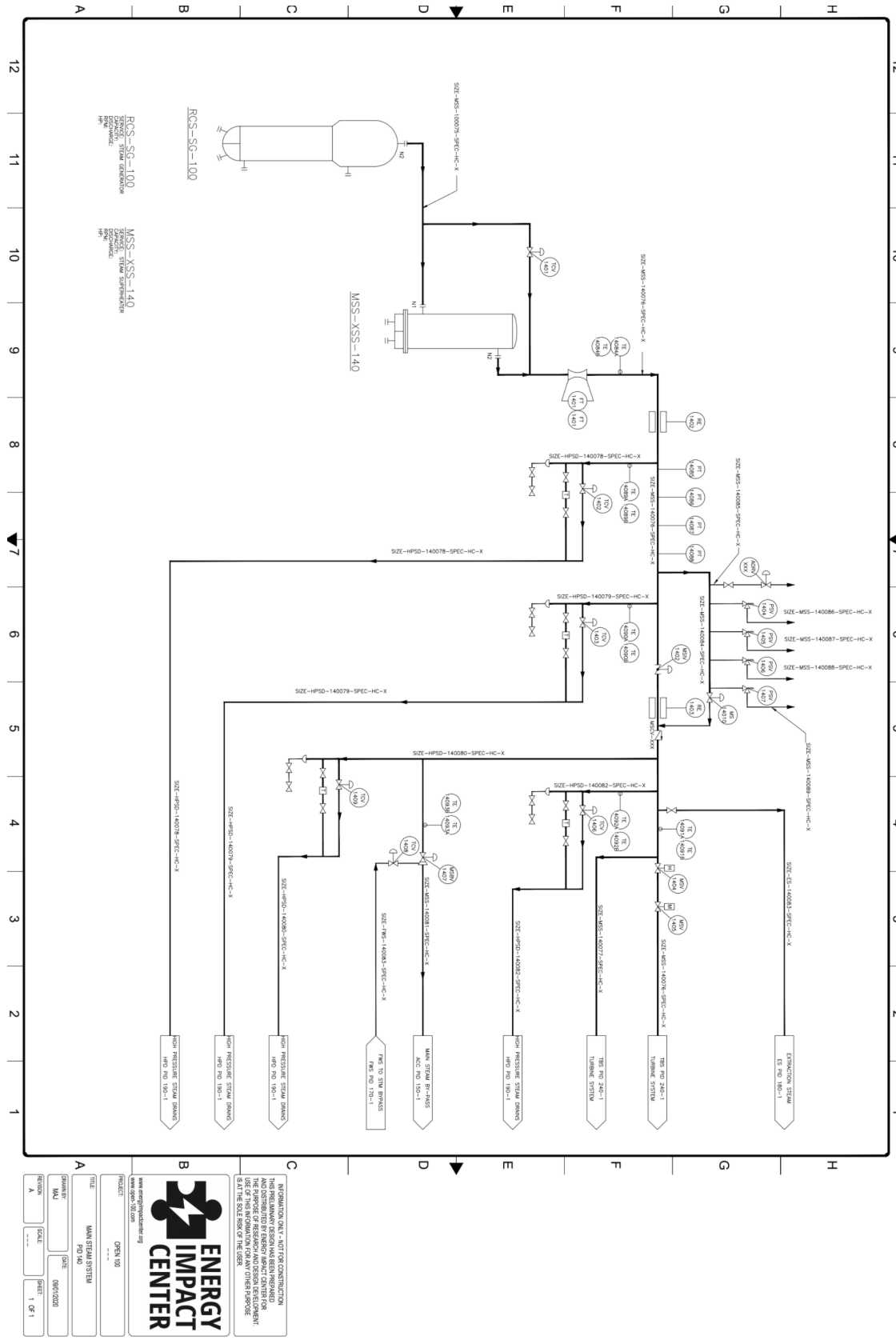


Figure 3.2: A P&ID from PID2Graph OPEN100.

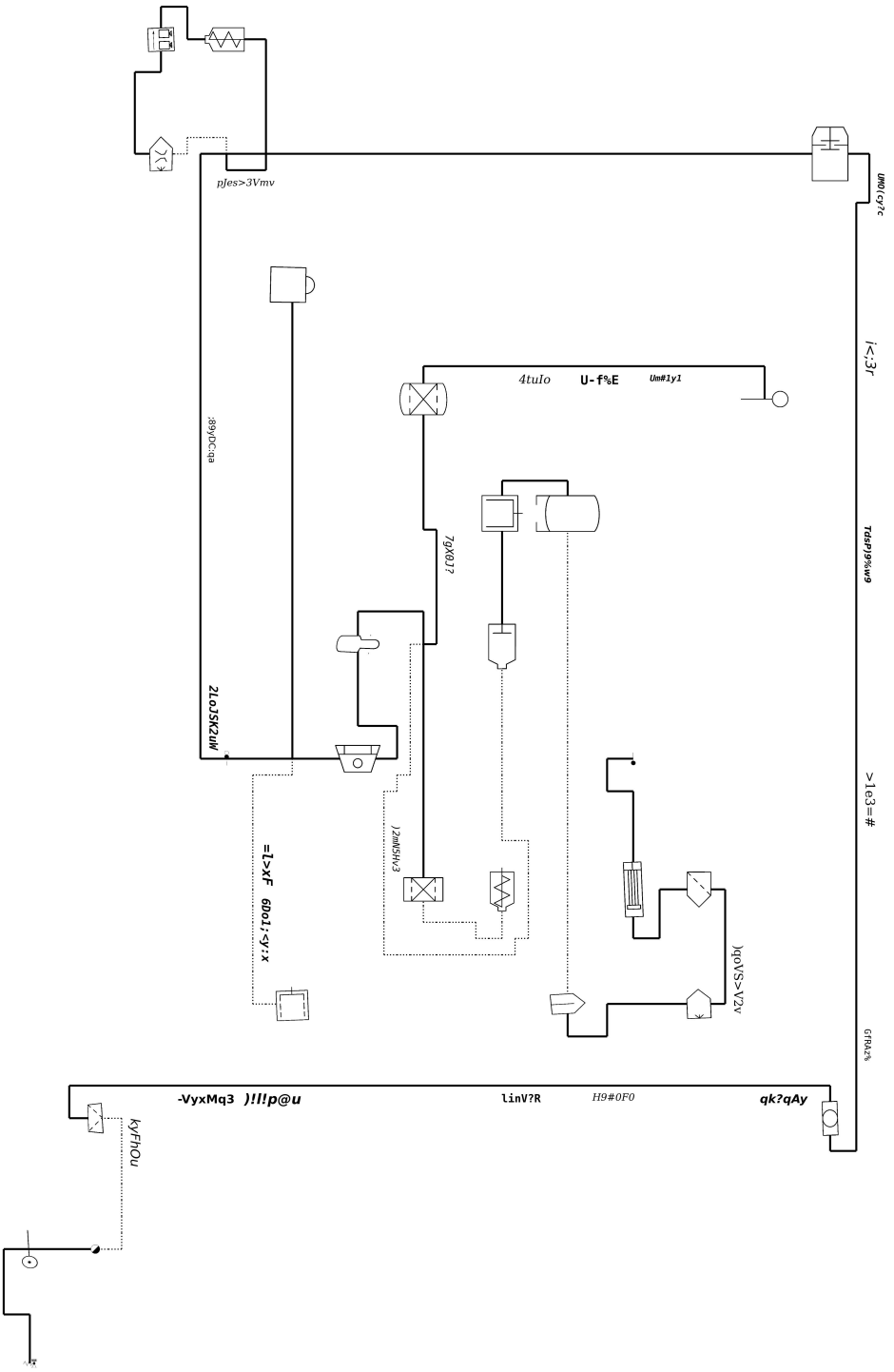


Figure 3.3: A P&ID from PID2Graph Synthetic.

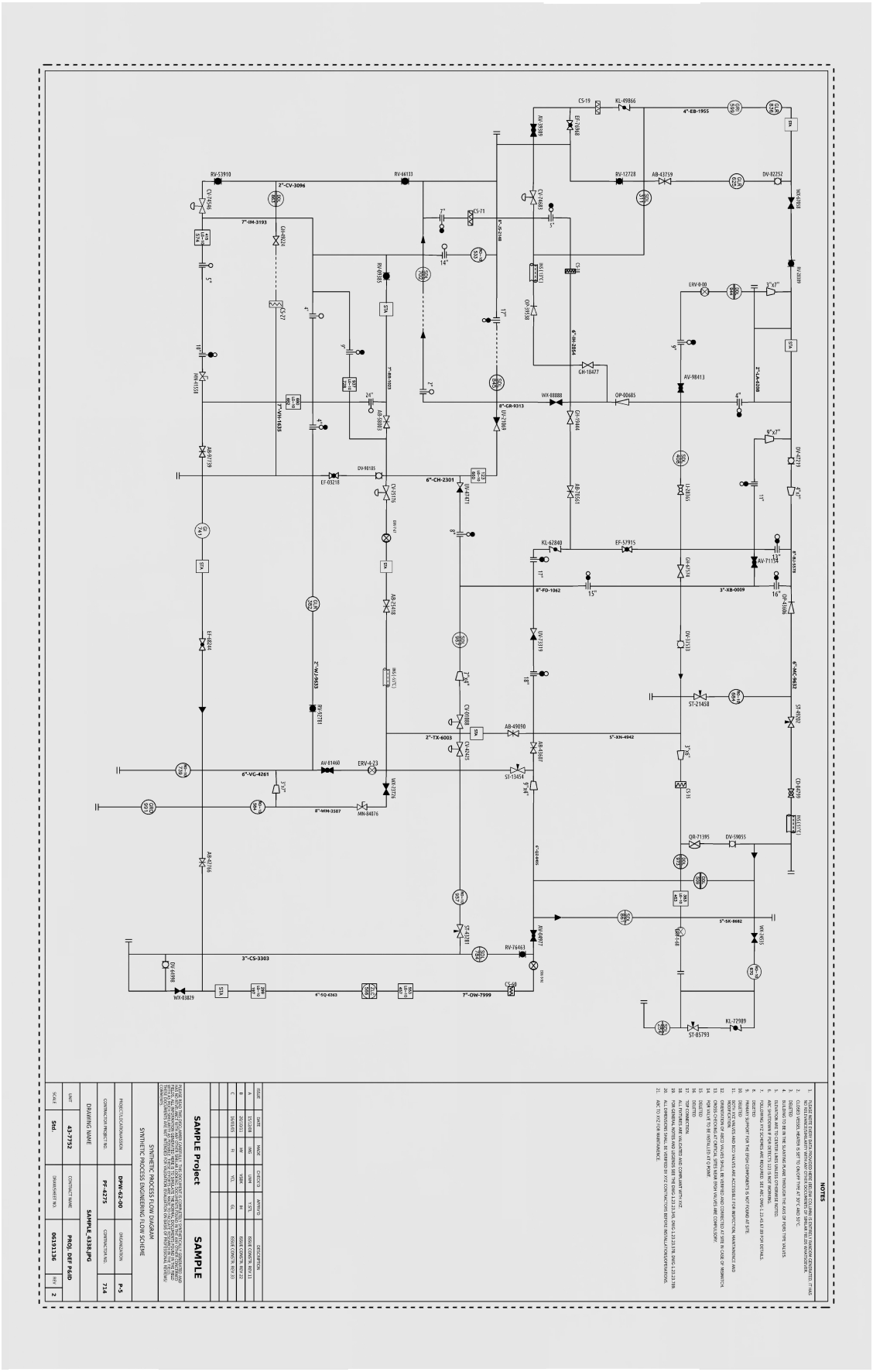


Figure 3.4: A P&ID from Dataset PID.

Table 3.2: The occurrence of each symbol category within the selected subset of the three datasets. The share of each all share of symbols belonging to each category and the number of symbols within the parenthesis.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
General	23.7% (234)	70.5% (792)	31.2% (1862)
Instrumentation	35.4% (350)	0% (0)	21.1% (1261)
Valve	26.0% (257)	10.6% (119)	47.7% (2854)
Tank	3.5% (35)	11.9% (134)	0.0% (0)
Inlet & Outlet	9.6% (96)	0.0% (0)	0.0% (0)
Pump	1.6% (16)	7.0% (79)	0.0% (0)
All	100% (988)	100% (1124)	100% (5977)

3.2 Model

The VLM used in the thesis is Gemini 2.5 Flash. It is a sparse mixture-of-expert model that has been trained with distillation [5]. It has internal reasoning capabilities, but for this thesis these capabilities have been disabled for faster performance and reduced computational load. The temperature is set to zero to make the model as deterministic as possible with the highest probability token being selected during generation [45]. Text extraction was performed using the Azure AI Document Intelligence Read OCR model, which is a proprietary model from Microsoft [46].

3.3 Data Preprocessing

The resolution of the P&IDs vary with the smallest schematic in the PID2Graph OPEN100 dataset being 1994x1330 and the largest being 3342x2222. All schematics in Dataset PID have a resolution of 7168x4562 and for PID2Graph Synthetic it is 7000x4500. For an equitable comparison, the latter two datasets are resized to half their original resolution, resulting in 3584x2281 and 3500x2250 respectively.

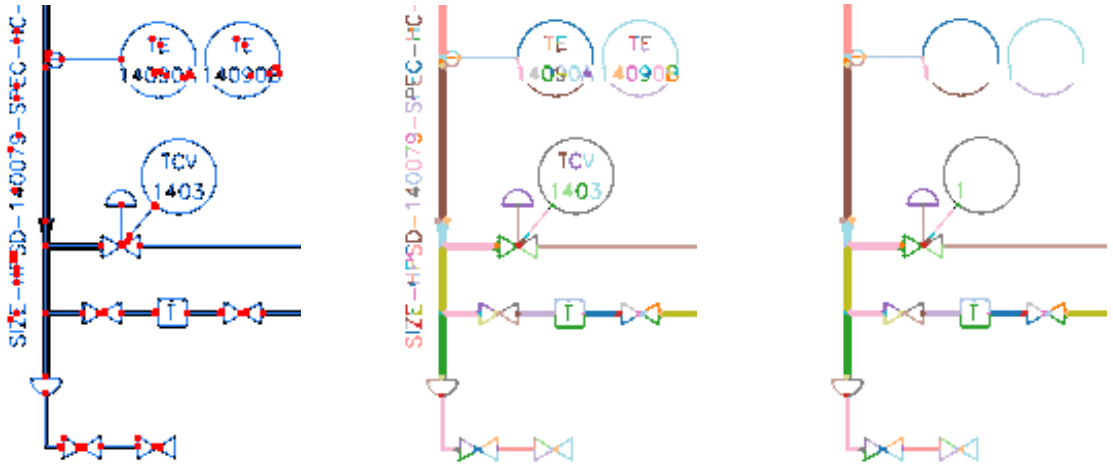
The preprocessing also includes binarization for future skeletonization and connected component labeling. Additionally, the legends and scale in the schematics are removed by covering them with the background color.

3.4 Symbol Detection

Symbols are detected by finding which parts of a schematic are part of a symbol. A schematic is partitioned into segments which are analyzed. These segments are classified as either symbols or non-symbols by the VLM. The classified segments are used to derive bounding boxes.

3.4.1 Dividing Schematics into Segments

An essential part of the P&ID digitization pipeline is to utilize VLMs for classification. This includes an approach to transform the task of symbol detection into a classification problem. This is achieved by dividing schematics into segments that are classified as either part of a symbol or non-symbol. More segments will demand more computation, but will also provide more detail. For high efficiency, the number of segments should be as low as possible while still providing high detail. Basically, producing small detailed segments in detailed areas of a schematic. The creation of segments should also adapt do different image resolutions and magnifications.



(a) The skeleton with junction points.

(b) All segments derived from the junctions.

(c) The final segments after the OCR.

Figure 3.5: A visualization of a skeletonized schematic to the left where the blue pixels mark the skeleton and the red pixels are the junctions of the skeleton. The image in the center shows the resulting segments in random colors. On the right is the segments after the OCR has removed the text.

By skeletonizing a schematic, an outline of the schematic is extracted, where all lines are one pixel thick, as seen in Figure 3.5. Thus, each pixel in the skeleton that neighbors more than two other skeleton pixels is a junction. These junctions are marked in the left image in Figure 3.5. The aim is to divide the schematic by the junctions to produce segments. This is done by removing the junctions from the skeleton and performing connected component labeling on the remaining skeleton. This results in the skeleton being separated into small disconnected segments, marked by blue pixels on the left in Figure 3.5. By dividing a schematic at the junctions, it will create more segments in complex areas.

To derive the final segments, all non-background pixels must belong to a segment. This is achieved by dilating all the disconnected skeleton segments to include the other pixels. This approach assigns each non-background pixel to its closest skeleton segment.

3.4.2 Optical Character Recognition (OCR)

At this stage, every black pixel in the binarized image belongs to a segment. However, this also includes segments that belong to the text in the schematic. The OCR is used to detect text in images and outputs both the extracted characters and the areas where the text is located. The OCR is performed on the original image, not on the preprocessed version. Segments that are fully within the text area provided by the OCR will be removed by setting the color of those segments to the background color. This produces the final segments that are used by the VLM in the next step. The resulting segments are illustrated on the right in Figure 3.5.

3.4.3 Classifying Multiple Segments at Once

Instead of querying the VLM once for each segment, multiple segments can be classified simultaneously. The eight colors of red, blue, green , yellow, orange, purple, cyan, and pink were identified as distinct from each other as well as distinct from the white background and black shapes in the schematic, see Figure 3.6. This enables eight segments to be uniquely marked when sent to the VLM. The colors are selected for high visual distinguishability. Selecting colors is a tradeoff between the number of segments that can be classified in parallel and how visually different the colors are. The colors also need to have clear names to make it unambiguous when the prompt is referencing the colors in the image. The choice of colors is inspired by the default colors of two visualization tools [47, 48]. But where the contrast has been increased for more distinct RGB values, as seen in Table 3.3.

Figure 3.6: The colors used for coloring the segments.

Table 3.3: RGB values for the eight colors.

Color	Red	Green	Blue
red	100%	0%	0%
blue	0%	0%	100%
green	0%	69%	31%
yellow	100%	100%	0%
orange	100%	50%	0%
purple	50%	100%	50%
cyan	0%	100%	100%
pink	100%	0%	100%

The segments are organized into groups of at most eight segments. This is done by applying a special version of k-means clustering where the clusters can be limited by their maximum size [49]. It has a complexity of $\mathcal{O}(n^4 \log(n))$, where n is the number

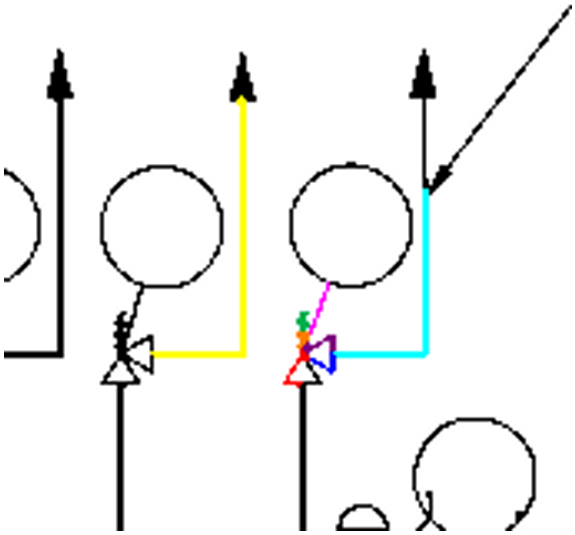


Figure 3.7: Eight segments colored in distinct colors.

of points being clustered. The 2D coordinates of the segments center points are provided to the clustering algorithm. The number of clusters is calculated as the number of segments divided by seven, rounded down to the nearest whole number, resulting in approximately seven segments per group on average. Due to the segments being clustered by their position, it results in groups of segments that are spatially close to each other. Figure 3.7 is an example of a group of eight segments clustered together.

(a) Without redundancy. **(b)** With redundancy.

Figure 3.8: Clustering with and without redundancy.

The combination of at most eight segments per group and an average close to seven offers the clustering algorithm more flexibility when clustering. If the number of segments are set so that all clusters contain at most eight segments, then a few segments located far from any other segments must be grouped with distant segments. This would create groups of segments that are not close to each other. But by adding some redundancy in the number of clusters, the algorithm can afford to create half-filled clusters to outliers, potentially leading to more compact clusters. This is illustrated by the example in Figure 3.8. When each cluster has to contain exactly four points, the blue cluster includes an outlier. But when an extra cluster

is added, the redundancy leads to more compact clusters.

3.4.4 Classifying Segments

For classification, the VLM is given three images and a text prompt. The first two images of the P&ID are the preprocessed schematics where the text is removed, and the segments being classified are colored, each with a unique color. The only difference between the first and second images is that the first is a large crop and the second is a close-up crop, as can be seen in Figure 3.7. The purpose of the big crop is to give more context and the close-up crop provides the details. The third image is a close-up crop of the same area but in the original image. Due to the VLM having a maximum token capacity for an image, not all details will be preserved [50]. For a big crop, a reduction in quality can remove important information from the area of interest. This is why the close-up crop provides a more focused image where the tokens only describe the area of interest. During the preprocessing, the OCR removes the segments constituting text. In order to still include the textual information, the model is provided with a third image of the original image.

The text prompt provides instructions on how to classify each segment and lists what is considered a symbol and what is not. This includes textual examples, explaining that valves and tanks are symbols, while pipes are not symbols. There are no image examples to keep it is zero-shot. The full segment classification prompt can be seen in the appendix. A substantial part of the prompt is information surrounding the coloring of segments. It clarifies that all segments were originally black and that the colors do not have any meaning, as they exist solely to target the prompt.

The prompt also contains information on the location of the segments. The first piece of positional information is the bounding boxes of each colored segment. The bounding boxes are expressed in relative coordinates for the close-up crop. The second piece is the information about which colored segments are adjacent to each other. Examples of the bounding box prompt and adjacency prompt are found in the appendix. The purpose of this information is to provide extra clarity if the VLM were to struggle with identifying small segments or distinguishing between the colors.

Finally, the VLM is prompted to write out its reasoning in one sentence per segment and then answer the classification task in JSON format. In the JSON output, each color is classified as either symbol or other, where other being the category for non-symbols. If the VLM cannot identify a segment, it is instructed not to include it in the output.

3.4.5 Joining Symbol Segments

The symbol segments classified as part of a symbol by the VLM are further processed into symbol components. Symbol components are a collection of segments that belong to the same symbol. This part relies on the assumption that two symbol

segments adjacent to each other belong to the same symbol. Hence, adjacent symbol segments are joined. Two segments are considered adjacent if there is a pixel in one segment that is 8-connected to a pixel in the other segment.

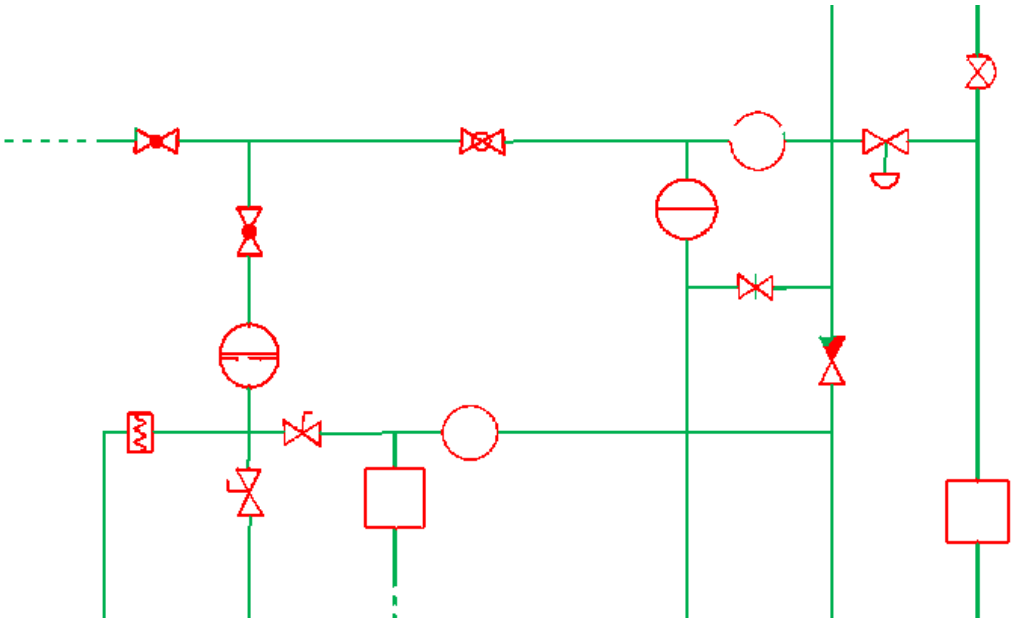


Figure 3.9: A part of a P&ID from Dataset PID where segments classified as symbols are colored red and the others are green.

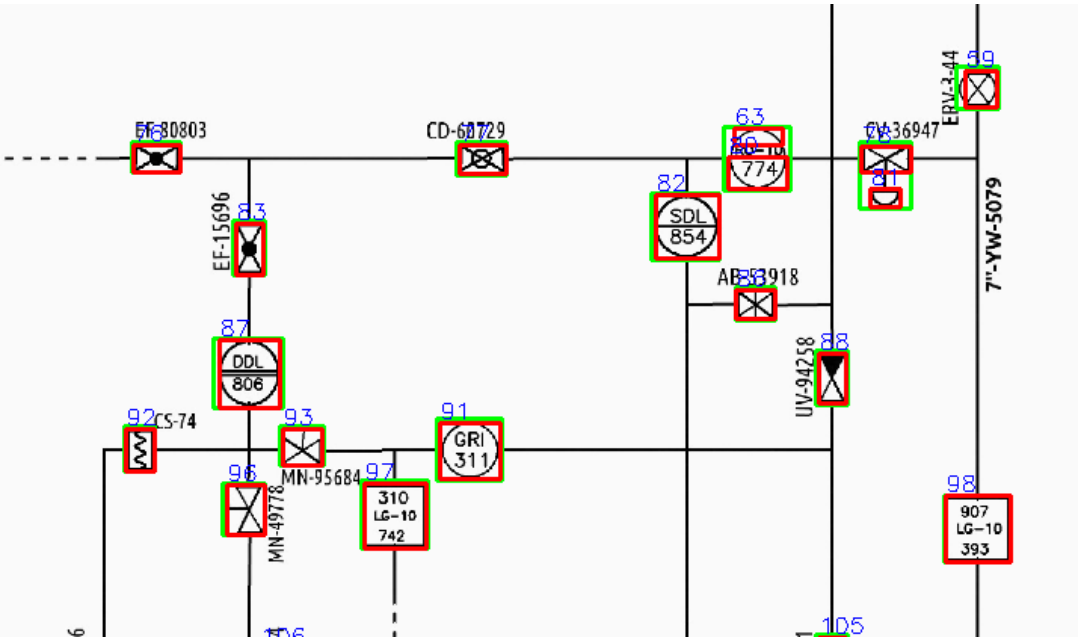


Figure 3.10: A part of a P&ID from Dataset PID with the bounding boxes derived from the classified segments. The predicted bounding boxes are red and the ground truth is green. The blue numbers are indices for each prediction.

It is from the symbol component that the bounding boxes are derived, with each

symbol component resulting in a matching bounding box. The height, width and position of a bounding box are set to precisely fit all pixels associated with a symbol component. The output of the symbol detection step consists of symbol components and their corresponding bounding boxes. The classified segments are visualized in Figure 3.9, and the corresponding bounding boxes are in Figure 3.10.

However, joining adjacent segments can only produce symbol components that are connected components. Thus, a symbol consisting of multiple parts separated by gaps will result in multiple symbol components. This is a limitation of only relying on joining adjacent segments, and is the reason for the merging of symbols in the next step of the digitization pipeline.

3.5 Symbol Merging

The merge step is an optional refinement step in the digitization pipeline. It is provided with the symbol components from the previous step and uses the VLM to determine which symbol components should be merged. As mentioned in Section 3.4.5, if there is a gap between two parts belonging to the same symbol, they will not become part of the same symbol component, thus requiring the merge step. An example of this situation is illustrated in Figure 3.11, where the green symbol component should be merged with the blue symbol component while the red component belong to another symbol. Another reason for merging symbol components is that the segment classifier can misclassify symbol segments, creating gaps between symbol segments belonging to the same symbol. In the merging step, each symbol component is investigated individually to find other symbol components it should be merged with.

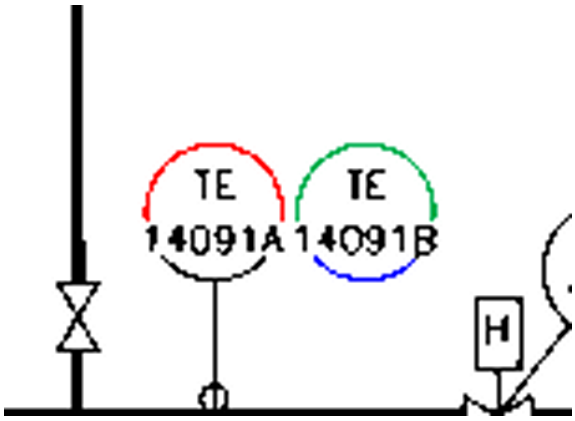


Figure 3.11: An example of the image sent to the VLM for the merge step of the green symbol component. Where the red and blue symbol components are the components within the search radius it can be merged with.

3.5.1 Identifying Potential Components to Merge

The merging step also relies on the eight distinct colors visualized in Figure 3.6. The symbol component being investigated is colored green, leaving seven colors for the surrounding symbol components that will potentially be merged with the green symbol component. The surrounding symbol components are selected within a search radius. If there are more than seven symbol components within the search area, it selects the seven closest, measured by the distance between their center points.

$$search\ radius = max(bounding\ box\ diagonal,\ 0.01 \cdot schematic\ diagonal) \quad (3.1)$$

The search radius is calculated by the maximum of the diagonal of the bounding box for the symbol component of interest and one percent of the entire schematic's diagonal. The search radius is defined in relation to the size of the symbol component and the size of the schematic to make the digitization pipeline scale invariant. This is necessary for it to handle schematics of different resolution and magnification. The combination of selecting at most seven components and the search radius is the method employed to restrict merging to near by symbol components. This limits the merging to relevant symbol components in proximity to each other.

3.5.2 Merging Components with VLMs

The VLM performs the assessment of which symbol components should be merged together. The VLM is provided with two different crops of the preprocessed P&ID, where the symbol components have been colored in unique colors. Similar to the segment classification, the prompt clarifies that the color does not have any meaning in the schematic. it is only used to guide the prompt. The VLM writes out its reasoning and positional information is provided in the form of bounding boxes for each symbol component. Contrary to the segment classification, the preprocessed images contain text information. Hence, there is no need for a third image with text.

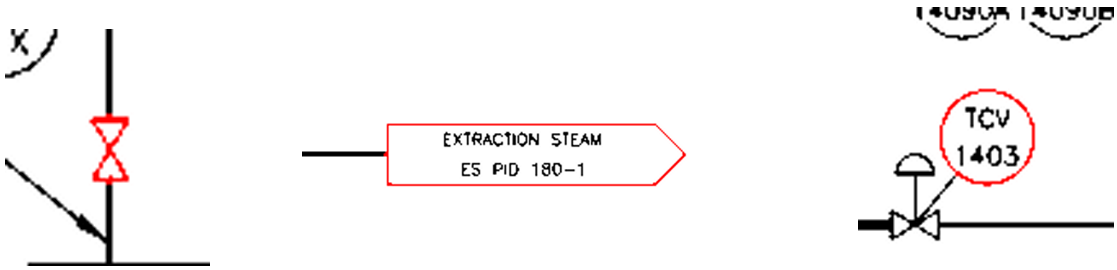
The prompt instructs the model to merge symbol components that belong to the same symbol with the full prompt available in the appendix. It is instructed to output mappings between the colored symbol components in the following format: {'red':'green', 'blue':'green'}. This is an example of how it is supposed to answer that the red and green symbol components should be merged, as well as the blue and green. The symbol component of interest is green, and the VLM is only meant to output potential merges with green. Proposed merging between two other colored symbol components will be ignored. For the example in Figure 3.11, the desired output would be {'blue':'green'}, because only the blue symbol component should be merged with the green. If the output were to include {'blue':'red'}, it will be ignored because the merge step is analyzing one symbol component at a time, and it is the green symbol component that is being analyzed.

The VLM is used once per symbol component to find which neighboring symbol components it should be merged with. In the scenario where there are two symbol components A and B. When A is analyzed, the VLM determines that it should be merged with B. But when B is analyzed, it does not propose any merge between B and A. In this scenario, the two segments will merge because there is no need for a double approval on a merge. In the scenario of three symbol segments A, B and C. If the VLM merges A with B and B with C, then all three will be merged together. This also applies when there are more than three symbol components that are merged together.

The output of the merge step is the merged symbol components as well as their corresponding bounding boxes. This is the same output format as the symbol detection step, which is the reason the merge step being an optional step that refines the result from the symbol detection.

3.6 Symbol Classification

The final step of the digitization pipeline is the symbol classifier. The symbol classification requires the symbol component for the detected symbols and a list of the categories it will use for classification.



(a) Valve. **(b)** Outlet. **(c)** Instrumentation.

Figure 3.12: Close-up crops used for symbol classification of detected symbol components of a valve, outlet and instrumentation symbol.

The VLM is provided with two images of the preprocessed P&ID, where one is a close-up crop and the other is a larger crop for more context. The images only contain one colored symbol component for the symbol being classified. The prompt, which can be found in the appendix, is also similar to the merge prompt regarding clarifications about the colored segments and a bounding box for the colored symbol component. Figure 3.12 shows examples of the close-up crops with highlighted symbol components.

The VLM is instructed to find the best suited category for the symbol component, which is the only colored part in the two image inputs. All three datasets include a miscellaneous category named general, which is the most vague category because

it could include any symbol. For this reason, the model is instructed to classify a symbol component as general only if none of the other categories match.

3.7 Evaluation

The digitization pipeline is evaluated on both symbol detection and symbol classification capabilities. The predicted bounding boxes are paired with the ground truth so that the sum of the IoU of the paired bounding boxes is maximized, solving the assignment problem. In line with previous research, a threshold of 0.5 is used for a prediction to be accepted as a true positive. Due to the digitization pipeline outputting bounding boxes before and after the merge step, both are evaluated.

$$\mathrm{F}_1 = \frac{2}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{3.2}$$

The evaluation measures for symbol detection are precision, recall and F_1 score. The evaluation metrics can also be recorded at multiple IoU thresholds. For symbol classification, accuracy is the only metric. In case the symbol classifier outputs an answer that matches none of the categories, it will be counted as a false positive. When the symbol classifier is evaluated on a dataset, it will be provided with a list of all symbol categories in that dataset.

The symbol classification is performed on all symbol components from the symbol detector. But for evaluating the classifier, only symbol detections that pass the 0.5 IoU threshold are used to evaluate the symbol classifier. The classifier cannot be assessed on classifying symbols that do not exist, which is why only detected symbols that pass the threshold are used for the evaluation.

4

Results

In this chapter, the results from the evaluation of the symbol detector and symbol classifier are presented. It includes how the symbol detector performed on the different datasets, as well as the impact of the merge step and position information in the prompts. The performance of the classifier is also presented for the different datasets and for each symbol type.

4.1 Symbol Detection

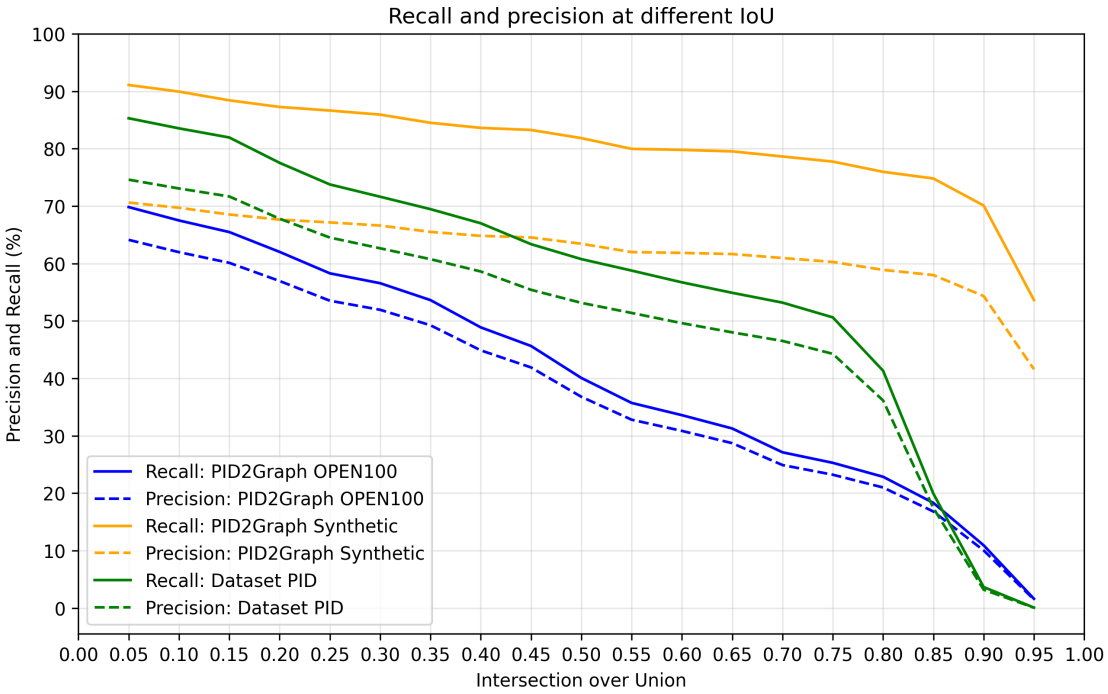


Figure 4.1: Performance for the symbol detector.

Figure 4.1 shows the average recall and precision per dataset for different IoU thresholds. The dashed lines represent precision, and the solid lines represent recall. For all three datasets, the precision is never higher than the recall for every IoU threshold. For all IoU thresholds between 0.05 and 0.85, PID2Graph Synthetic has the highest performance, while PID2Graph OPEN100 has the lowest.

Standard detection refers to bounding boxes derived from the classified segments, and detection merged refers to the bounding boxes produced by the merge step. The performance of these two variant of symbol detection is gathered in Table 4.1.

Table 4.1: Symbol detection performance.

Detection	Dataset	Recall	Precision	F_1
Standard	PID2Graph OPEN100	40.1%	36.8%	38.4%
Standard	PID2Graph Synthetic	81.9%	63.5%	71.5%
Standard	Dataset PID	60.8%	53.2%	56.7%
Merged	PID2Graph OPEN100	39.6%	51.7%	44.8%
Merged	PID2Graph Synthetic	83.8%	85.1%	84.5%
Merged	Dataset PID	67.2%	74.9%	70.9%

4.1.1 Symbol Detection after Merging

After the merge step, the recall is never higher than the precision for any dataset at any IoU threshold. As presented in Table 4.1, the merge step increased the performance of all evaluation metric for the three datasets, with the only exception being the recall for PID2Graph OPEN100, which decreased by 0.11 percentage point. Figure 4.2 shows the performance for the symbol detector for different IoU threshold after the merge step has being applied. Table 4.1 lists the evaluation metrics for the three datasets before and after the merge steps for an IoU threshold of 0.5.

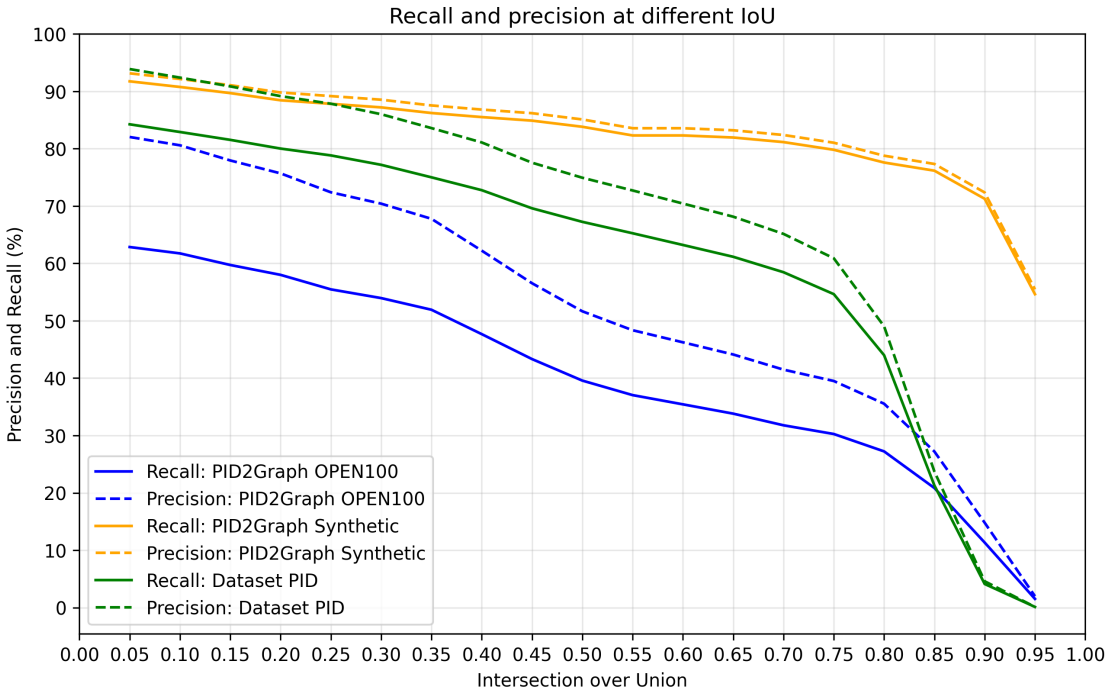


Figure 4.2: Performance for the symbol detector after the merge step.

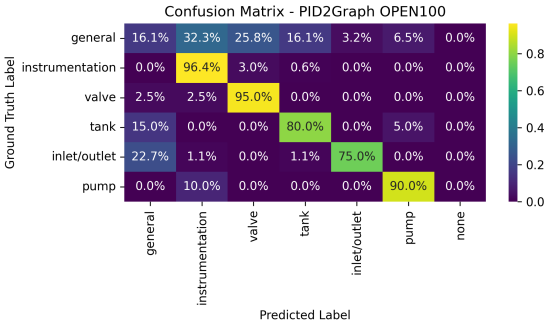
Table 4.2: Share of symbols found only before, or after the merge step, as well as those never found, and those found both before and after the merge step.

Found	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
Never	55.0%	16.1%	31.4%
Before	5.5%	0.1%	1.3%
After	5.0%	2.1%	7.8%
Both	34.6%	81.8%	59.4%

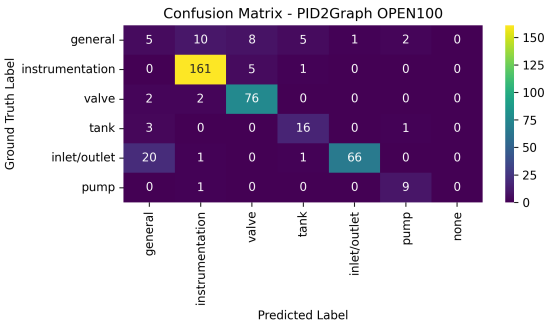
In Table 4.2, the share that is detected only after merging shows the share that was found due to the merge step. The share of symbols only found before merging is the share of symbols that were detected by standard detection but were later destroyed by the merge step.

4.2 Symbol Classification

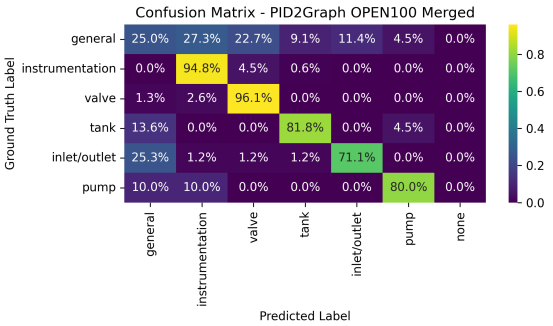
The accuracy of the symbol classifier is presented in confusion matrices for all three datasets in Figure 4.3, 4.4 and 4.5. The symbol classifier is evaluated on the detected symbol before and after the merge step. The confusion matrices on the left are in percentages and in absolute numbers on the right. The category none, in the confusion matrices, is for predictions that match none of the categories.



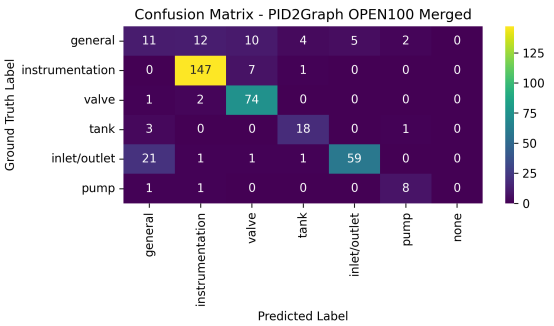
(a) Classifier on standard detections.



(b) Classifier on standard detections.

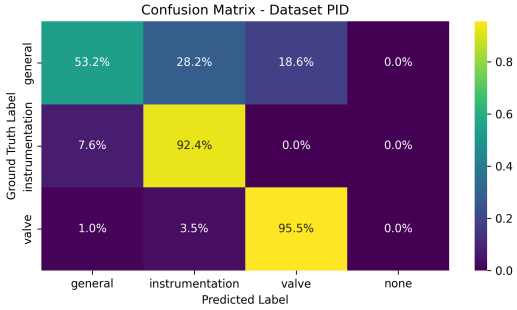


(c) Classifier on merge detections.

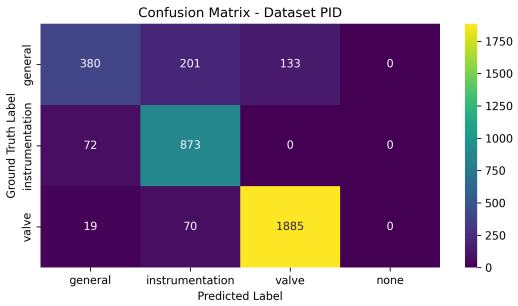


(d) Classifier on merge detections.

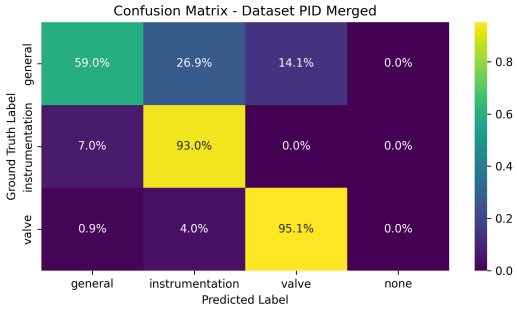
Figure 4.3: Confusion matrices for classification on PID2Graph OPEN100.



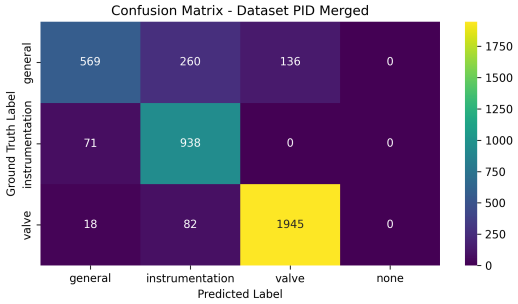
(a) Classifier on standard detections.



(b) Classifier on standard detections.

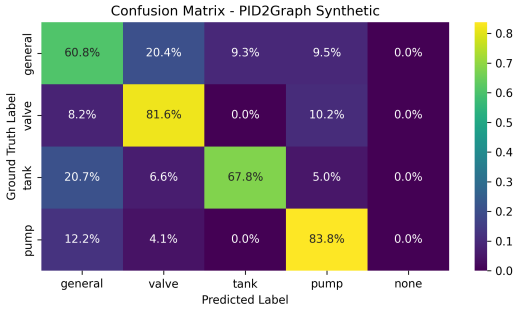


(c) Classifier on merge detections.

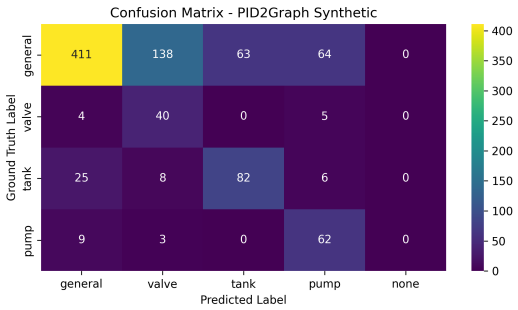


(d) Classifier on merge detections.

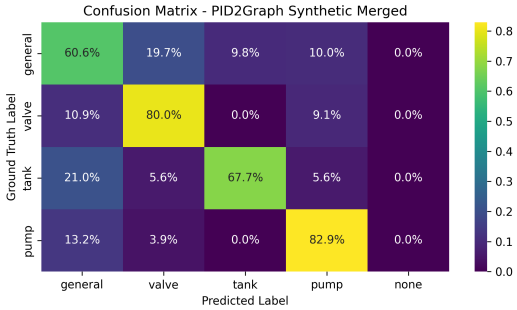
Figure 4.4: Confusion matrices for classification on Dataset PID.



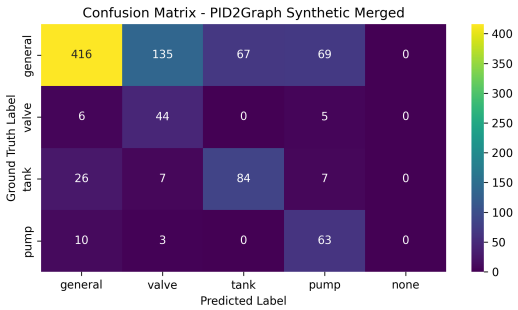
(a) Classifier on standard detections.



(b) Classifier on standard detections.



(c) Classifier on merge detections.



(d) Classifier on merge detections.

Figure 4.5: Confusion matrices for classification on PID2Graph Synthetic.

PID2Graph OPEN100 has six categories, Dataset PID has three, and PID2Graph Synthetic has four. Valve and general are the only categories that exist in all three datasets. Across all confusion matrices, the lowest performing category is general. The total accuracy for the three datasets for the classification of symbols with and without the merge step is presented in Table 4.3. Acc. is the total accuracy across the entire dataset. In Table 4.3, S and M indicate whether the classifier is performed on the standard or merge detections, respectively. Error Gen. is the share of all misclassifications that involved the general category. Involved symbols include symbols that were misclassified as general and general symbols that were misclassified. Gen. Share is the share of all symbols within a dataset that belong to the category general. Non-Gen. Acc. is the accuracy of the classification if all misclassifications involving the general category is excluded.

Table 4.3: Classification performance.

Dataset	Acc.	Error Gen.	Gen. Share	Non-Gen. Acc.
PID2Graph OPEN100 - S	84.1%	80.9%	14.1%	96.4%
PID2Graph OPEN100 - M	81.1%	79.7%	17.9%	95.3%
PID2Graph Synthetic - S	64.7%	93.2%	77.6%	89.3%
PID2Graph Synthetic - M	64.4%	93.4%	77.4%	89.7%
Dataset PID - S	86.4%	85.9%	22.2%	97.5%
Dataset PID - M	85.9%	85.5%	26.2%	97.2%

4.3 Symbol Type and Detection Performance

Table 4.4 presents the recall for each category for the standard detection of symbols. Similarly, table 4.5 lists the recall per category for the merge detection. The tables show that despite the merge step decreased the overall recall on PID2Graph OPEN100, it increased for tanks and general symbols. It also show that the merge step decreased the recall for valves on PID2Graph OPEN100 whereas it increased for valves in PID2Graph Synthetic and Dataset PID.

Table 4.4: Recall for standard detection per symbol category.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
General	13.2%	85.3%	38.3%
Instrumentation	47.7%	N/A	75.9%
Valve	31.1%	41.2%	69.2%
Tank	57.1%	90.3%	N/A
Inlet & Outlet	91.7%	N/A	N/A
Pump	62.5%	93.7%	N/A
Total	40.1%	81.9%	60.8%

Table 4.5: Recall for merge detection per symbol category.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
General	18.8%	86.7%	51.8%
Instrumentation	44.3%	N/A	80.0%
Valve	30.0%	46.2%	71.6%
Tank	62.9%	92.5%	N/A
Inlet & Outlet	86.5%	N/A	N/A
Pump	62.5%	96.2%	N/A
Total	39.6%	83.8%	67.2%

4.4 Symbol Size and Symbol Proximity Related to Detection Performance

Recall in relation to symbol size is visualized in Figure 4.6. The symbols have been divided into deciles on the area of each symbol's bounding box. The merge step increased the recall of the smallest 10% of symbols in Dataset PID by over 30 percentage points. Symbol detection across all three datasets has the lowest performance on the smallest decile which also stays true after the merge step has been applied. For PID2Graph OPEN100 and PID2Graph Synthetic, their difference in recall is smaller at the lowest and highest decile than for the middle deciles. The merge detection for Dataset PID has the smallest difference in recall between its best and worst performing decile.

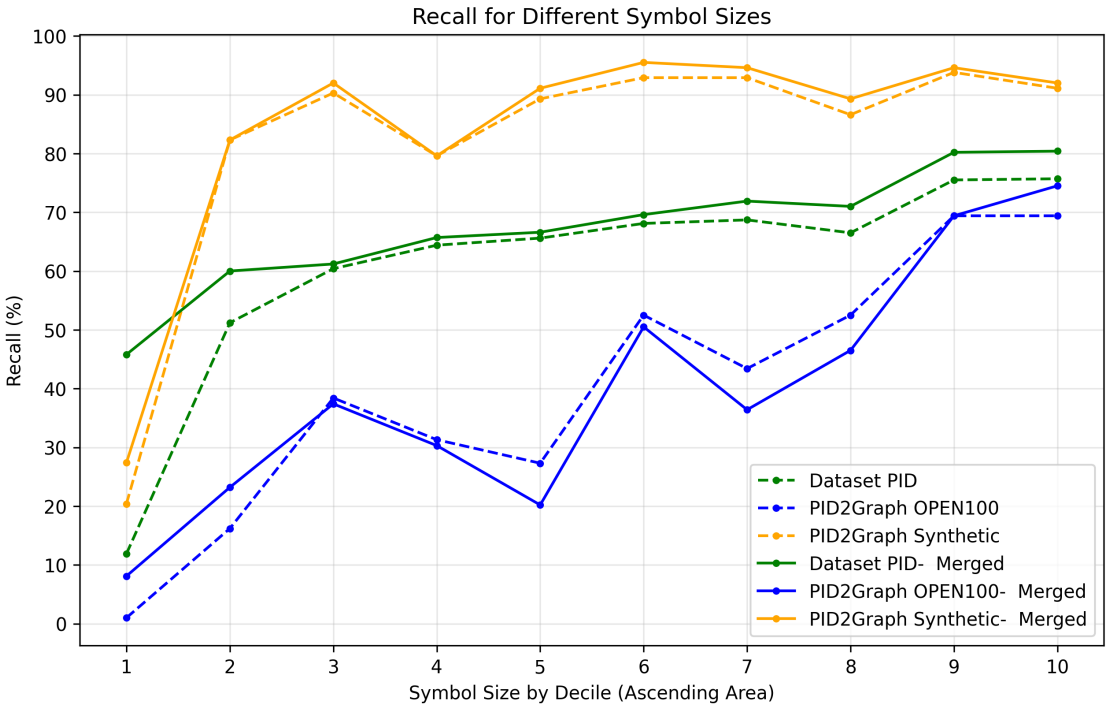


Figure 4.6: Recall per decile measured by symbol size.

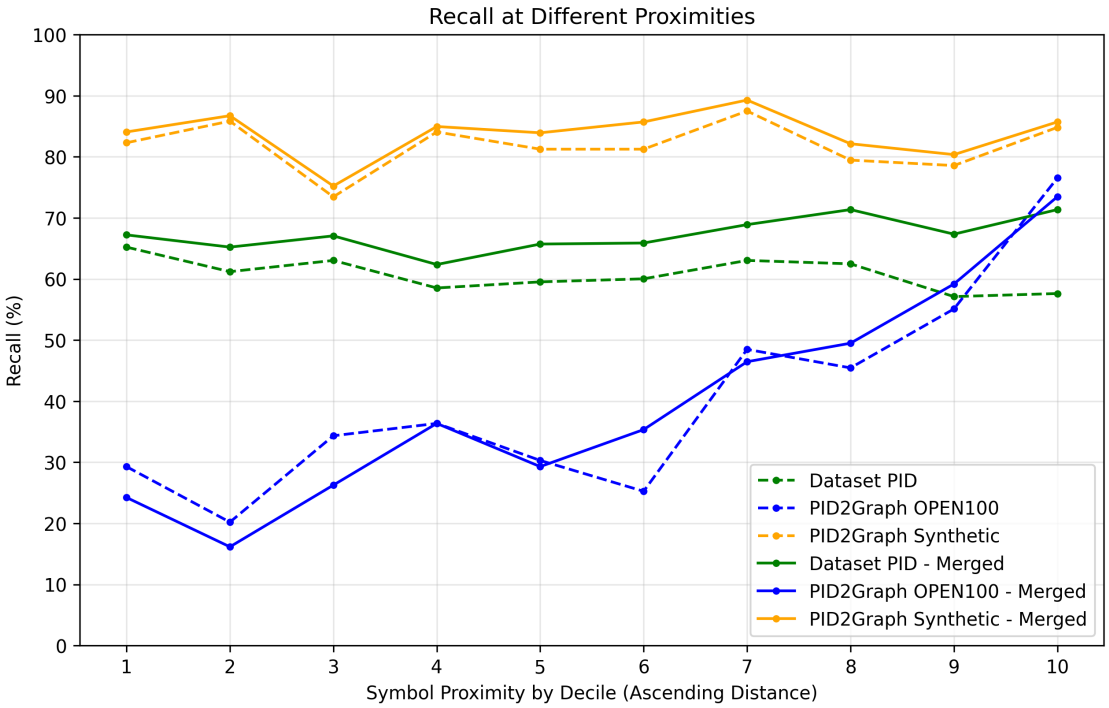


Figure 4.7: Recall per decile measured by the a symbols proximity to its closest symbol.

Figure 4.7 illustrates how proximity of a symbol to other symbols is related to detection performance. The symbols have been divided into deciles by the distance from their closest symbol. The distance is the shortest distance between the bounding box of a symbol and any other bounding box in the schematic. PID2Graph OPEN100 had the biggest difference in recall between the best and worst performing decile. For both standard detection and merge detection, PID2Graph Synthetic is the best performing dataset for every decile. For the 10th decile (the symbols with the most space around them), PID2Graph OPEN100 has higher recall than Dataset PID while the recall across the entire dataset is the opposite.

Table 4.6: Symbol diagonal measured in pixels per dataset.

Dataset	Median	Mean	Standard deviation
PID2Graph OPEN100	49.5	75.9	96.7
PID2Graph Synthetic	97.1	94.8	34.0
Dataset PID	54.6	56.1	17.7

Table 4.6 lists the mean and median symbol size as well as the standard deviation for each dataset. It measures symbols size as the diagonal of the bounding box of a symbol. PID2Graph Synthetic has on average larger symbol but PID2Graph OPEN100 has the highest variance in symbol size. Table 4.7 show that PID2Graph OPEN100 has on average the shortest distance to its nearest symbol.

Table 4.7: Symbol proximity measured in pixels per dataset.

Dataset	Median	Mean	Standard deviation
PID2Graph OPEN100	9.0	21.1	42.4
PID2Graph Synthetic	181.1	200.9	132.0
Dataset PID	47.7	58.0	43.7

4.5 Location Guided Prompting

As described in Chapter 3, the segment classification prompt and the merge prompt include location information in the form of bounding boxes associated with each colored segment. The segment classification prompt also includes adjacency information, which is another form of location information. Table 4.8 presents the performance of the symbol detector with and without the merge step, when location information is not provided in the prompt.

Table 4.8: Detection performance without bounding box prompts

Detection	Dataset	Recall	Precision	$\mathbf{F}_1$
Standard	PID2Graph OPEN100	41.8%	37.6%	39.6%
Standard	PID2Graph Synthetic	82.6%	58.7%	68.6%
Standard	Dataset PID	65.9%	55.4%	60.2%
Merged	PID2Graph OPEN100	37.7%	49.4%	42.8%
Merged	PID2Graph Synthetic	85.0%	83.5%	84.2%
Merged	Dataset PID	69.8%	76.9%	73.2%

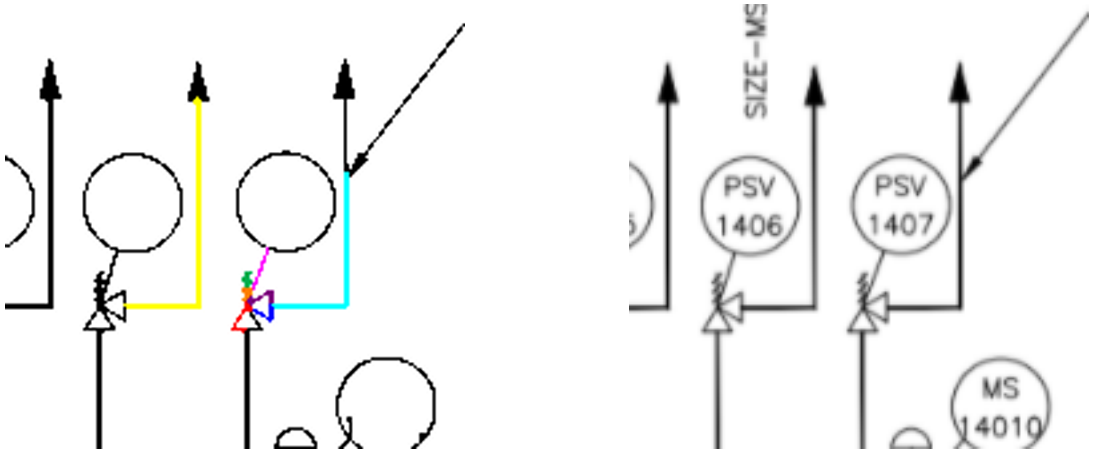
Table 4.8 has the same IoU threshold of 0.5 as Table 4.1, which lists the detection performance with location information in the prompts. The difference in performance is presented in Table 4.9. With location information removed, recall increased across all three datasets for standard detection. The precision decreases or increases with location information depending on the dataset used for evaluation.

Table 4.9: Performance difference in symbol detection after removing location information in the prompts. The difference is measured in percentage points (pp).

Detection	Dataset	Δ Recall	Δ Precision	$\Delta\mathbf{F}_1$
Standard	PID2Graph OPEN100	1.7pp	0.8pp	1.3pp
Standard	PID2Graph Synthetic	0.7pp	-4.8pp	-2.9pp
Standard	Dataset PID	5.1pp	2.3pp	3.5pp
Merged	PID2Graph OPEN100	-1.8pp	-2.2pp	-2.0pp
Merged	PID2Graph Synthetic	1.1pp	-1.6pp	-0.2pp
Merged	Dataset PID	2.6pp	2.0pp	2.3pp

4.6 Detailed Examples

4.6.1 Segment classification



(a) The colored segments. **(b)** Unprocessed image.

Figure 4.8: Examples of segments being classified for symbol detection.

The colored segments in Figure 4.8 lead to the following output of the segment classifier:

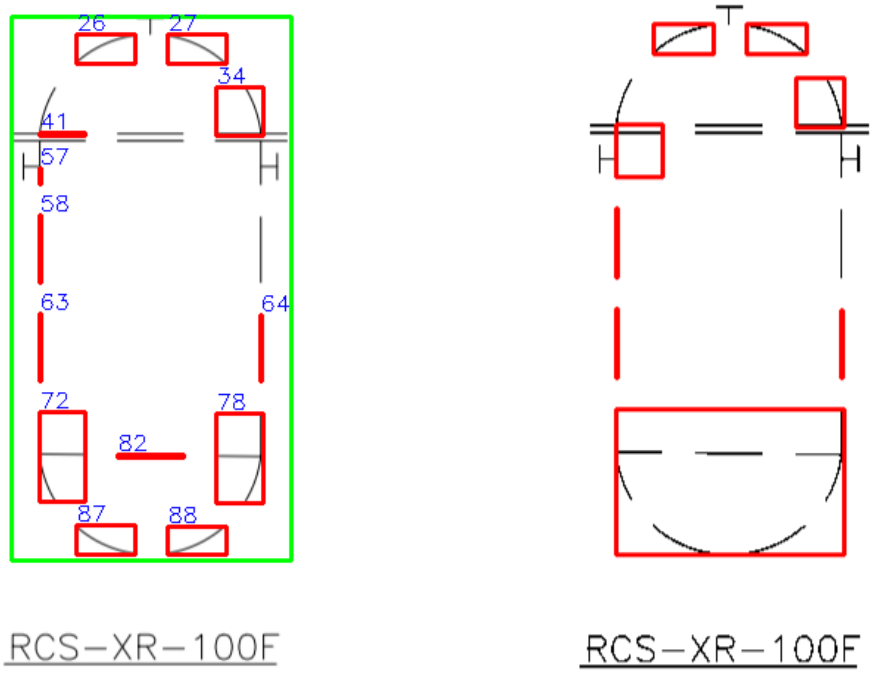
```
- The red segment is part of a valve symbol, which is a symbol.
- The blue segment is part of a valve symbol, which is a symbol.
- The green segment is part of a symbol, specifically a PSV (Pressure
  Safety Valve) symbol.
- The yellow segment is part of a pipe, which is a non-symbol.
- The orange segment is part of a symbol, specifically a PSV (Pressure
  Safety Valve) symbol.
- The purple segment is part of a valve symbol, which is a symbol.
- The cyan segment is part of a pipe, which is a non-symbol.
- The pink segment is part of a symbol, specifically a PSV (Pressure
  Safety Valve) symbol.
‘‘‘json
{
  "red": "symbol",
  "blue": "symbol",
  "green": "symbol",
  "yellow": "other",
  "orange": "symbol",
  "purple": "symbol",
  "cyan": "other",
  "pink": "symbol"
}
```

Each segment is classified as either symbol or other (non-symbol). The VLM also reasons in one sentence per segment. The VLM correctly classifies the yellow and cyan colored segments as non symbols. The circles are instrumentation symbols, and the two instances where two triangles point towards each other are valve symbols. The VLM manages to extract text from the image and use it in its reasoning.

4.6.2 Segment Unsuccessfully Not Merged

Figure 4.9 shows a tank symbol consisting of multiple unconnected parts. The symbol detector has identified some of those segments but not all. The merge step merged parts in the lower third of the tank together. But none of the resulting bounding boxes has in IoU of at least 0.5, resulting in the tank not being counted as detected.

The merge step does not produce any bounding box with the required IoU for the tank symbol to be counted as detected. However, it reduces the number of bounding boxes from 13 to 8, resulting in fewer false positives in the evaluation, improving the precision.

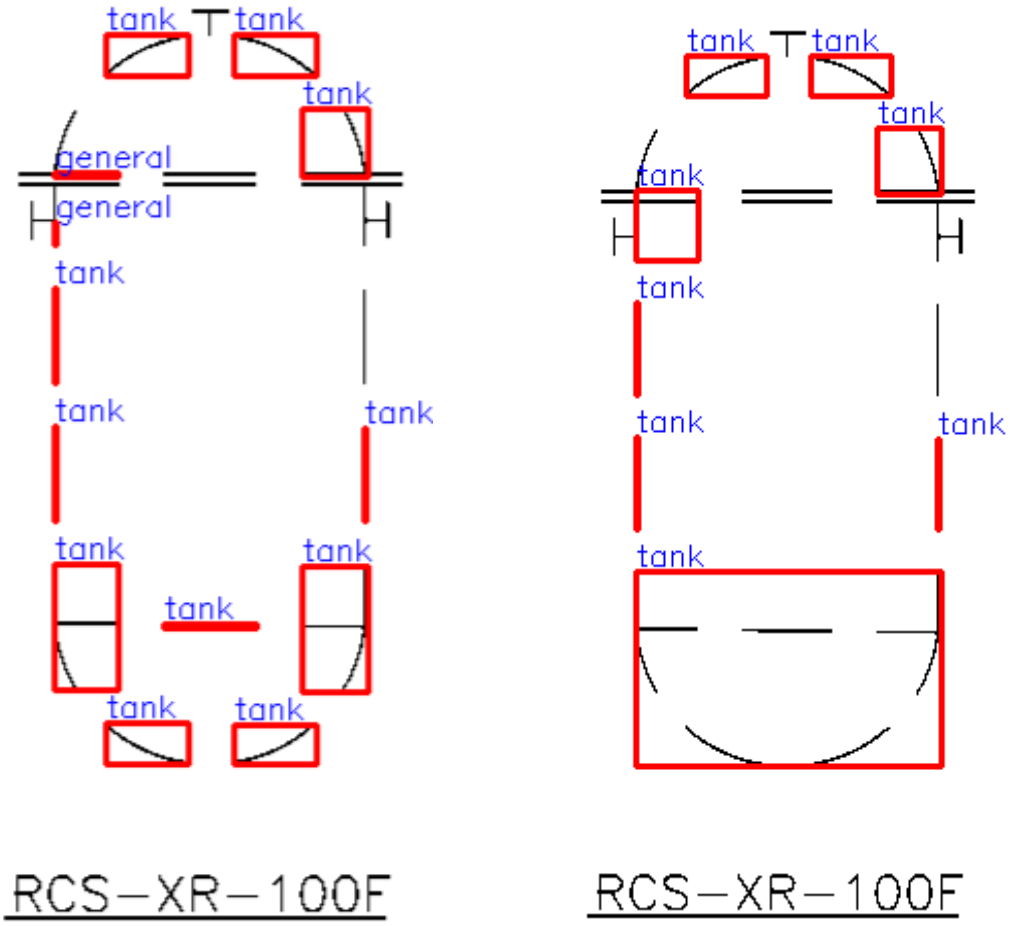


(a) Before merge step. **(b)** After merge step.

Figure 4.9: The symbol detection before and after the merge step for a tank in a P&ID from the PID2Graph OPEN100 dataset. The green box is the bounding box of the ground truth and the red boxes are the predictions.

Figure 4.10 shows the result of the symbol classifier on the tank, both before and

after the merge step. After the merge step, the symbol classifier correctly predicts all identified parts to be part of a tank. But the symbol classifier predicts two parts to be general symbols when using the symbol detections before the merge step.



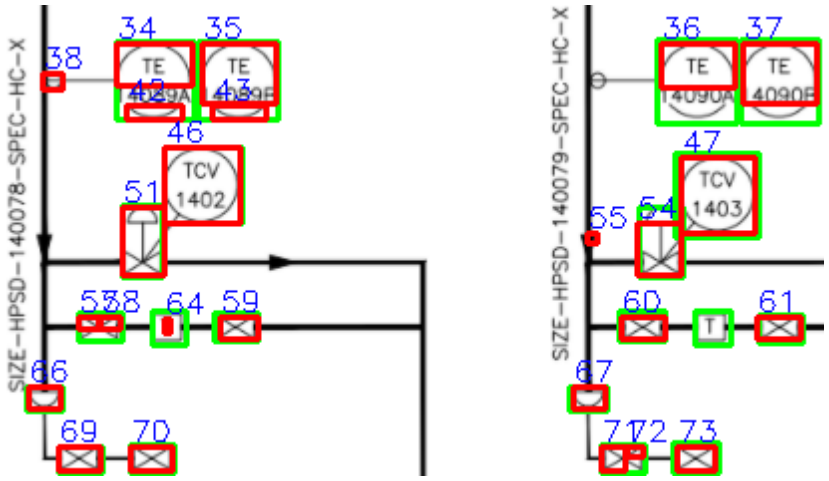
(a) Before merge step. **(b)** After merge step.

Figure 4.10: Symbol classification on the results from the symbol detector before and after the merge step. The blue text show the predicted classes for the parts of a tank in a P&ID from the PID2Graph OPEN100 dataset.

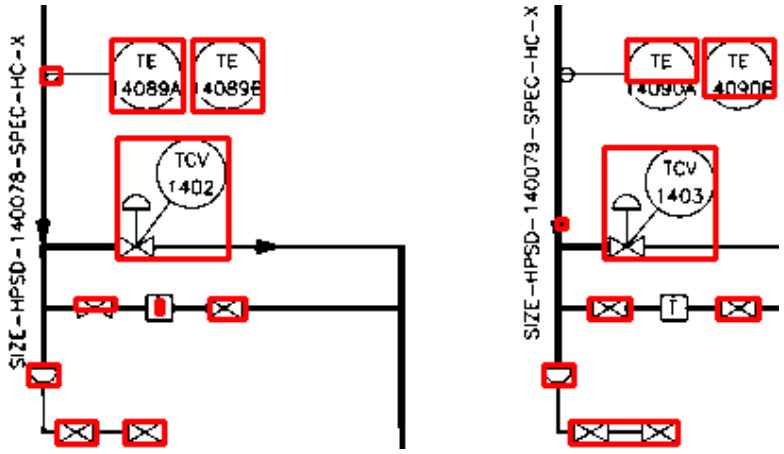
4.6.3 Entire Symbols Merged

The way in which the merge step impacts symbol detection is visualized in Figure 4.11. Bounding boxes 34 and 42 are merged, as well as 35 and 43. These are examples of how the merge step merges detections that belong to the same symbol, producing bounding boxes with a higher IoU than before the merge step.

There are also instances where the merge step takes two correctly detected symbols and merge them together, resulting in neither being detected, such as the two valves found in bounding box 69 and 70 in Figure 4.11. Another example is seen in bounding box 46 and 51 or 47 and 54 where a valve and an instrumentation are merged.



(a) Standard detection where the red bounding boxes are the prediction and the green bounding boxes are the ground truth.



(b) Symbol detection after the merge step.

Figure 4.11: Example of symbol detection before and after the merge step in a part if a P&ID from a schematic in the PID2Graph OPEN 100 dataset.

4.7 Number of VLM Queries

The digitization pipeline divides schematics into segments, which are grouped, where each group requires one query for the VLM to classify all segments. This section presents the number of VLM queries at different stages of the pipeline for the datasets. The number of segments are presented in Table 4.10.

Table 4.10: Number of segments per dataset and average number of segments per schematic per dataset.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
Segments	8224	30920	68663
Seg. Avg.	685.3	618.4	1373.3

4.7.1 Queries for Standard Detection and Classification

Standard detection involves segment classification for symbol detection, and symbol classification is performed for every detection. Some detections are false positives that the symbol detector mistakes for being symbols, but they are still classified because the model itself cannot determine which detections are faulty. Each detection requires one query to classify it as one of the symbol types.

Table 4.11: VLM queries per step in the pipeline for standard detection and classification as a share and in absolute numbers within the parenthesis.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
Detection	52.1% (1169)	75.2% (4397)	68.9% (9788)
Classification	47.9% (1076)	24.8% (1450)	41.1% (6834)
All	100% (2245)	100% (5847)	100% (16622)

Table 4.11 lists the number of queries to perform standard detection and classification per dataset. Whereas Table 4.12 lists the average number of queries per schematic for each dataset.

Table 4.12: Average number of VLM queries per schematic per step in the pipeline for standard detection and classification as a share and in absolute numbers within the parenthesis.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
Detection	52.1% (97.4)	75.2% (87.9)	68.9% (195.8)
Classification	47.9% (89.7)	24.8% (29.0)	41.1% (136.7)
All	100% (187.1)	100% (116.9)	100% (332.4)

4.7.2 Queries for Merge Detection and Classification

For merge detection, the merge step requires additional queries to the VLM, but it also reduces the number of VLM queries in the classification step. The merge step is only applied for symbol components that have other symbol components within the search radius. Hence, the number of requests associated with the merge step is not proportional to the number of symbol components but rather to the extent to which they are within the search radius of each other.

Table 4.13: VLM queries per step in the pipeline for merge detection and classification as a share and in absolute numbers within the parenthesis.

Category	PID2Graph OPEN100	PID2Graph Synthetic	Dataset PID
Detection	44.6% (1169)	73.2% (4397)	54.0% (9788)
Merge	26.5% (695)	8.3% (500)	16.4% (2983)
Classification	28.9% (757)	18.4% (1107)	29.6% (5364)
All	100% (2621)	100% (6004)	100% (18135)

Table 4.13 lists the number of queries to perform merge detection and classification per dataset. Whereas Table 4.14 lists the average number of queries per schematic for each dataset.

Table 4.14: Average number of VLM queries per schematic per step in the pipeline for merge detection and classification as a share and in absolute numbers within the parenthesis.

Category	**PID2Graph OPEN100**	**PID2Graph Synthetic**	**Dataset PID**
Detection	44.6% (97.4)	73.2% (87.9)	54.0% (195.8)
Merge	26.5% (57.9)	8.3% (10.0)	16.5% (59.7)
Classification	28.9% (63.1)	18.4% (22.1)	29.6% (107.3)
All	100% (218.4)	100% (120.1)	100% (362.7)

5
Discussion

This chapter investigates the strengths and weaknesses of the proposed zero-shot digitization pipeline for P&IDs by examining the effectiveness of the symbol detector and classifier. The thesis is an exploratory work investigating how VLMs can be utilized for domain specific zero-shot document understanding.

Zero-shot P&ID digitization enables more flexible digitization that can handle new symbols and overcome the limited amount of publicly available annotated data. The use of VLMs makes the pipeline more flexible because it can be easily adapted by changing the prompt. This is an advantage over deep learning models that have to be retrained to adapt to new symbols.

5.1 Symbol Detection

At the core of the digitization pipeline is a novel approach to transforming the task of symbol detection into a classification problem that can be answered by a VLM. By dividing schematics at the junctions in the skeleton, there will be more segments for more complex schematics. Thus, more complex schematics requires more VLM queries, increasing the computational cost. But simultaneously, dividing schematics into segments also makes it adaptable to schematics of varying complexity, size and resolution.

The symbol detector performs differently on the three datasets. The strongest performance is seen on PID2Graph Synthetic, which achieves a recall of 82% without the merge step. With the merge step, it achieves 84% recall and 85% precision. The symbol detections for PID2Graph Synthetic also perform well even for IoU thresholds above 0.5. With the merge step applied, the recall is 80% for an IoU threshold at 0.75 and more than 70% when the IoU threshold is set to 0.9, see Table 4.2. This indicates that the bounding boxes are very accurate. However, PID2Graph OPEN100 is the worst performing dataset with 40% recall, and it is the only non-synthetic dataset. Dataset PID is also synthetic and achieves a recall of 67%, but it is by far the dataset with the most symbols among the three, as was previously shown in Table 3.2.

5.1.1 The Impact of the Merge Step

The most significant impact of the merge step is the increase in precision seen in Table 4.1. Dataset PID benefits the most from merging, increasing the recall with 6.5 percentage points and the precision with 21.7 percentage points. The recall is further examined in Table 4.2, which shows that some detected symbols only reach the IoU threshold of 0.5 before the merge step while others only reach the threshold after merging. It demonstrate that the merge step can both improve and worsen the detection of individual symbols even though the merge step clearly increased the F_1 score on all three datasets.

For PID2Graph OPEN100 the merge step worsens the recall slightly while at the same time improving the precision significantly. Section 4.6.3 explains when how merging of symbols can cause correctly detected symbols to merge into faulty detections, reducing the recall. There are two main explanations for how the merge step can increase the precision without improving the recall. One explanation is visualized in the example in Figure 4.9 in section 4.6.2, where the merging of faulty detections leads to fewer faulty detections. Another explanation is that bounding boxes that already reaches the required IoU are merged with a small faulty detections nearby, resulting in a new bounding boxes that also satisfies the IoU threshold. In this scenario, the merge step only removes a small faulty detection, which improves the precision without impacting the recall.

The search radius in the merge step restricts merging to symbol components that are close to each other. The decline in performance for one of the datasets due to the merge step indicates that the merging has issues. By removing the search radius, it risks merging more symbol components that do not belong to the same symbol, which is already a problem, as seen in Figure 4.11. One possible approach would be to classify all symbol components and only allow components of the same category to be merged. However, this does require a high performing symbol classifier because the mistakes of the symbol classifier and merge step will compound.

5.1.2 Symbol Size and Density Impacting Performance

The symbol detector performs poorly on small symbols, with the only exception being Dataset PID after the merge step was applied. On PID2Graph OPEN100, the recall for the smallest 10% of all symbols is around one percent before the merge step, as seen in Figure 4.6. Hence, it is unable to detect the smallest symbols. For the largest 10% of symbols in the same dataset, the recall is 70% before the merge step. Even for PID2Graph Synthetic, where the recall for the entire dataset is 82% before the merge step, the smallest 10% of symbols only achieves a recall of 20%. This stark relationship between symbol size and performance can have many causes. One possibility is that the resolution is too low to detect the fine details that are necessary to find the smallest symbols. Other possibilities are that the smaller symbols have few characteristic traits, or that small symbols are mistaken for being part of larger nearby symbols.

Another variable impacting symbol detection is a symbols proximity to other symbols, illustrated in Figure 4.7. For PID2Graph OPEN100, the recall of the detection after the merge step has a difference of more than 55 percentage points between the second and tenth deciles. The tenth decile is the best performing and includes the 10% of symbols that are the furthest away from any symbol. This is unlike the other two datasets, where the recall is fairly constant across all deciles. Symbols with a short proximity distance are located in denser regions, whereas symbols with longer distances to their nearest neighbor are more isolated. Thus, this indicates that regions with high symbol density are making it more difficult for the symbol detector. For the 10% most isolated symbols, PID2Graph OPEN100 even outperforms Dataset PID. A possible interpretation of Figure 4.7 is that PID2Graph Synthetic and Dataset PID are sparse enough for the model, even in their densest areas. But for PID2Graph OPEN100, only the most sparse deciles are sparse enough for the model to detect a high proportion of the symbols.

5.1.3 Prompt with Location Information

The prompts include the bounding boxes, and the segment classification prompt also includes adjacency information as described in the methodology. Table 4.9 lists the change in performance when the location information is removed from the prompts compared to when it is included. The table shows instances where the recall increases while the precision decreases and vise versa, and where both metrics increase and decrease. It all depends on which dataset it was evaluated on and whether the merge step was applied or not. Hence, there is no clear advantage or disadvantage to including location information in the prompt, except the prompt being longer.

5.2 Symbol Classification

The result of the symbol classifier is mainly illustrated in the confusion matrices in Figure 4.3, 4.5 and 4.4. The main problem for the symbol classifier is the general category, which consistently underperforms the other symbol categories across all confusion matrices. The general category is a collection of all remaining symbols that do not match standard P&ID categories. This indicates that for VLMs to effectively classify symbols, it is better to have clear categories defined by what they are, instead of what they are not.

A remarkable result regarding accuracy is that the symbol classifier performs worse when classifying symbols found with merge detection than with standard detection, as seen in Table 4.3. The symbol classifier is only evaluated on the detected symbols that reach an IoU threshold of 0.5. The degradation in symbol classification indicates that the merged detections reaching the threshold are of lower quality than the non-merged detections reaching the threshold. However, the performance metrics illustrated in Figure 4.1 and 4.2 show no indications that merge detection would have worse recall for higher IoU thresholds. Hence, merge detection produces better detections in terms of overlap between the predictions and ground truth. A

possible explanation is that the symbol classifier is capable of correctly predicting the category, even when given a small fragment of a symbol, as seen in Figure 4.10. Therefore, the symbol classifier can perform well even if the symbol detector underestimates the size of a symbol. But if the symbol detector overestimates the size of a symbol, it might risk including parts of nearby symbols, which negatively impact the symbol classifier.

5.3 Visual Prompting

The thesis is based on visual prompting using colors to mark multiple areas with precision, which can also be differentiated by the VLM. The number of unique colors used limits the number of areas that the VLM can analyze per query. Utilizing other techniques for marking could enable the full potential of VLMs, allowing more work per query. The colors used in the thesis were selected for their visual contrast and the clear names used to reference them in the text prompt. But VLMs might differentiate colors differently than humans, thus having the potential for more efficient use of colors.

The visual prompting used differs from approaches that highlight large areas or make a mark around the area of interest. By coloring the segments themselves, multiple adjacent segments can be differentiated without creating a clutter of marks that obscures the segments. The prompting does not require any names of the objects being marked because they are only referenced by color. Another potential technique is to add labels with numbers in the image to reference multiple areas in a single prompt [51]. But the labels in this approach risk obscuring the segments.

5.4 Limitations

There are multiple limitations to the thesis. One of them being how the evaluation of the thesis is limited to the publicly available annotated datasets, which consists mostly of synthetic data. The 12 P&IDs in PID2Graph OPEN100 are the only non-synthetic schematics, and these P&IDs are all from a single project. It would be beneficial to examine the digitization pipeline on annotated schematics from a range of projects and industries. The lack of annotated real-world data remains a limitation for research about P&ID digitization.

During the segment classification, segments are clustered into groups to be classified together in a single VLM query. The computational complexity of $\mathcal{O}(n^4 \log(n))$ for the clustering algorithm being used is not sustainable for larger schematics with more segments. The goal of the clustering algorithm is to group segments that are best suited to be classified together, which is itself a complex task that deserves further investigation. The constrained k-means algorithm achieves groups with a maximum number of segments, where segments within a group are closely located to each other. But there are numerous ways to perform this clustering, some of which might outperform the current algorithm.

After the segments have been classified as either part of a symbol or not, segments directly adjacent to each other are joined into symbol components. This is the most questionable assumption in the thesis because it disregards the scenario where two symbols are touching each other and will thus be joined automatically. This could also explain some of the impact of proximity distance on performance seen in Figure 4.7. The alternative would be to replace the automatic joining with a segment wise merge step. But this would require one VLM query per segment, which would sharply increase the computational cost of the digitization pipeline.

The thesis is focused on zero-shot detection, which has the advantage of not requiring any examples. Examples also risk introducing bias. The prompts do include textual examples but do not introduce any preconceptions about how the symbols look. However, few-shot learning could serve as a way to clarify instructions and improve performance. Just like zero-shot, few-shot learning is also more flexible and adaptable than training or fine tuning deep learning models. The zero-shot approach limits itself to the pretrained knowledge of the VLM. A few examples could thus be highly beneficial for domain specific tasks such as analyzing P&IDs.

The proposed digitization pipeline is computationally demanding, querying the VLM more than 360 times on average for a single P&ID in Dataset PID according to Table 4.14. The evaluation of the digitization pipeline includes segment classification, the merge step, and symbol classification of the detections derived with and without the merge step. The investigation of the effect of location information in the prompt, displayed in Table 4.9, required segment classification and the merge step to be evaluated without location information. The result presented in this thesis demanded over 55 000 VLM queries, each containing two or three images. However, within each step of the pipeline, the VLM requests can be performed in parallel for fast digitization. The VLM used is a distilled model with internal reasoning disabled. The proposed pipeline would be run with more demanding models to improve performance but at a higher computational cost. As VLMs continue to improve, approaches similar to the digitization pipeline proposed in this thesis will become more viable.

5.5 Future Work

There are many areas within the digitization of industrial schematics that can be improved in future work. If more extensive datasets become available, fine-tuning VLMs for the domains of P&IDs could become a feasible approach. This would give the VLMs more domain knowledge.

Another area to be improved is prompt engineering. One technique is automated prompt learning to derive the best format for prompts for a given task. Prompt engineering also includes few-shot learning, as described earlier, which has the potential to improve performance. Instead of providing a few input-output examples, as is done in few-shot learning, another approach would be to provide visual examples of

symbols to clarify what kind of symbols it should look for. For symbol detection, reference-based approaches, where the model knows exactly which symbols it should identify, are also an area for future research.

The proposed pipeline divides the digitization task into small tasks that are solved with a VLM. Future work could investigate how to leverage VLMs and other computer vision techniques for deciding how to break down and solve the problem. This could be approached with agentic AI to create a less rigid solution compared to the proposed pipeline.

The thesis is based on visual prompt engineering with multiple colors that enable VLM prompts to reference multiple distinct areas in an image. Future work could investigate which colors are the most distinct for VLMs, how many colors can be referenced per query, and how accurate the referencing is. Another avenue is to investigate other techniques for referencing multiple objects in an image.

Finally, despite the thesis focusing on P&IDs, similar approaches could be used within other domains. The most closely aligned problems are the digitization of other types of industrial schematics, such as electrical schematics and electrical drawings.

6
Conclusion

In this thesis, a method for zero-shot symbol detection and classification in P&IDs is presented. The proposed digitization pipeline is capable of localizing and classifying symbols without any training or example images of symbols. This is advantageous for use cases where training data is insufficient to train models. The pipeline also includes the merge step for the symbol detector, which is an optional refinement step for the symbol detector. Because the pipeline is based on a VLM, it can be adapted to project specific requirements by changing the prompt instead of retraining the model, leading to greater flexibility. By relying on the general knowledge of a VLM, the pipeline will not be biased towards any specific dataset.

The symbol detector achieves strong performance on the two synthetic datasets. But when examining symbols that have more space to their nearest symbol, the detector performs well on the non-synthetic dataset as well. For this reason, symbol separation is an important factor determining the success of the symbol detector. The symbol detector struggles to detect small symbols, and there are big variations in performance between symbol types.

The merge step significantly improves the precision of the symbol detector. However, the merge step can also destroy correct detections by merging symbols that should not be merged. But the net effect of the merge step is still positive. In the symbol detection and merge step, the VLM is given additional positional information in the text prompt for the highlighted segments in an image. There is no evidence that this additional information improves performance.

The symbol classifier provides solid performance for positively defined categories and performs worse for the negatively defined category. This reveals that categories should be clear and defined in terms of what they are rather than what they are not. This is evident across both non-synthetic and synthetic datasets.

The proposed digitization pipeline is computationally heavy, with a few hundred VLM queries per P&ID. Parallelization of VLM queries can reduce the time to process a schematic, but it does not change the amount of computation required.

Finally, the thesis contributes with advances in visual prompt engineering for P&IDs digitization. By using multiple colors to highlight different areas in an image, a VLM prompt can reference multiple distinct areas with high precision in a single query. This technique could be applied to other types of industrial schematics.

Bibliography

[1] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] V. Shteriyanov, R. Dzhusupova, J. Bosch, and H. H. Olsson, "Blueprintsymvl: A discriminative benchmark for vlm symbol recognition in engineering blueprints," *Results in Engineering*, vol. 28, p. 108171, 2025. [Online]. Available: https://doi.org/10.1016/j.rineng.2025.108171

[3] L. Jamieson, C. F. Moreno-García, and E. Elyan, "A review of deep learning methods for digitisation of complex documents and engineering diagrams," *Artificial intelligence review*, vol. 57, 05 2024.

[4] B. C. Kim, H. Kim, Y. Moon, G. Lee, and D. Mun, "End-to-end digitization of image format piping and instrumentation diagrams at an industrially applicable level," *Journal of Computational Design and Engineering*, vol. 9, no. 4, pp. 1298–1326, 08 2022. [Online]. Available: https://doi.org/10.1093/jcde/qwac056

[5] G. Comanici *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," arXiv.org, 2025. [Online]. Available: https://arxiv.org/abs/2507.06261

[6] X. Yue *et al.*, "Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 9556–9567.

[7] S. Paliwal, A. Jain, M. Sharma, and L. Vig, "Digitize-pid: Automatic digitization of piping and instrumentation diagrams," in *Trends and Applications in Knowledge Discovery and Data Mining*, M. Gupta and G. Ramakrishnan, Eds. Cham: Springer International Publishing, 2021, pp. 168–180.

[8] S. Shit *et al.*, "Relationformer: A unified framework for image-to-graph generation," in *Computer Vision – ECCV 2022*, ser. Lecture Notes in Computer Science, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., vol. 13697. Cham: Springer Nature Switzerland, 2022, pp. 422–439. [Online]. Available: https://doi.org/10.1007/978-3-031-19836-6_24

[9] V. Shteriyanov, R. Dzhusupova, J. Bosch, and H. H. Olsson, "Enhancing ocr-based engineering diagram analysis by integrating diverse external legends with vlms," *Journal of Software: Evolution and Process*, vol. 37, 11 2025.

[10] R. Rahul, S. Paliwal, M. Sharma, and L. Vig, "Automatic information extraction from piping and instrumentation diagrams," *arXiv (Cornell University)*, 01 2019. [Online]. Available: https://doi.org/10.48550/arxiv.1901.11383

[11] S. Bai *et al.*, "Qwen3-vl technical report," arXiv.org, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2511.21631

[12] R. Hantach, G. Lechuga, and P. Calvez, "Key information recognition from piping and instrumentation diagrams: Where we are?" in *Document Analysis and Recognition – ICDAR 2021 Workshops*, E. H. Barney Smith and U. Pal, Eds. Cham: Springer International Publishing, 2021, pp. 504–508.

[13] M. Stürmer, M. Graumann, and T. Koch, "From engineering diagrams to graphs: Digitizing p&ids with transformers," in *2025 IEEE 12th International Conference on Data Science and Advanced Analytics (DSAA)*. Birmingham, UK: IEEE, 2025. [Online]. Available: https://doi.org/10.1109/DSAA65442.2025.11248012

[14] S. Danish, A. Sadeghi-Niaraki, S. Khan, L. Dang, L. Tightiz, and H. Moon, "A comprehensive survey of vision–language models: Pretrained models, fine-tuning, prompt engineering, adapters, and benchmark datasets," *Information Fusion*, vol. 126, Feb. 2026, publisher Copyright: © 2025 Elsevier B.V.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3104–3112. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5a18e133cbf9f257297f410bb7eca942-Paper.pdf

[16] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, 04 2022.

[17] A. Radford, "Improving language understanding with unsupervised learning," *OpenAI Res*, 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423/

[19] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8748–8763. [Online]. Available: https://proceedings.mlr.press/v139/radford21a.html

[20] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, "Perceiver: General perception with iterative attention," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 18–24 Jul 2021, pp. 4651–4664. [Online]. Available: https://proceedings.mlr.press/v139/jaegle21a.html

[21] J.-B. Alayrac *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[22] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proceedings of the 40th International Conference on Machine*

Learning, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 19 730–19 742. [Online]. Available: https://proceedings.mlr.press/v202/li23q.html

[23] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 34 892–34 916. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf

[24] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019. [Online]. Available: https://arxiv.org/pdf/1904.09751

[25] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1701.06538

[26] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2101.03961

[27] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1503.02531

[28] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1910.01108

[29] N. Islam, Z. Islam, and N. Noor, "A survey on optical character recognition system," *arXiv preprint arXiv:1710.05703*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1710.05703

[30] N. Subramani, A. Matton, M. Greaves, and A. Lam, "A survey of deep learning approaches for ocr and document understanding," *arXiv preprint arXiv:2011.13534*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2011.13534

[31] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, p. 236–239, Mar. 1984. [Online]. Available: https://doi.org/10.1145/357994.358023

[32] F. Bolelli, S. Allegretti, L. Baraldi, and C. Grana, "Spaghetti labeling: Directed acyclic graphs for block-based connected components labeling," *IEEE Transactions on Image Processing*, vol. 29, pp. 1999–2012, 2020. [Online]. Available: https://doi.org/10.1109/TIP.2019.2946979

[33] F. Bolelli, S. Allegretti, and C. Grana, "One dag to rule them all," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3647–3658, 2022. [Online]. Available: https://doi.org/10.1109/TPAMI.2021.3055337

[34] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 740–755.

[35] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM computing surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[36] S. Schulhoff *et al.*, "The prompt report: A systematic survey of prompt engineering techniques," *arXiv preprint arXiv:2406.06608*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.06608

[37] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 24 824–24 837. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[38] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 22 199–22 213. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf

[39] S. Yao *et al.*, "Tree of thoughts: Deliberate problem solving with large language models," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 11 809–11 822. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf

[40] A. Shtedritski, C. Rupprecht, and A. Vedaldi, "What does clip know about a red circle? visual prompt engineering for vlms," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 11 987–11 997. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Shtedritski_What_does_CLIP_know_about_a_red_circle_Visual_prompt_ICCV_2023_paper.html

[41] A. Cooper *et al.*, "Rethinking vlms and llms for image classification," *Scientific Reports*, vol. 15, no. 1, p. 19692, 2025. [Online]. Available: https://doi.org/10.1038/s41598-025-04384-8

[42] S. Ghosh, "Visual language model as a judge for object detection in industrial diagrams," *arXiv preprint arXiv:2510.03376*, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2510.03376

[43] Open100. Energy Impact Center. [Online]. Available: https://www.open-100.com/

[44] J. M. Stürmer, M. Graumann, and T. Koch, "Pid2graph," Feb. 2025. [Online]. Available: https://zenodo.org/records/14803338

[45] Experiment with parameter values. Google. [Online]. Available: https://docs.cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/adjust-parameter-values

[46] Microsoft, "Azure AI Document Intelligence Read OCR Model," 2025, version 4.0. [Online]. Available: https://learn.microsoft.com/azure/ai-services/document-intelligence/prebuilt/read
[47] Specifying colors. Matplotlib. [Online]. Available: https://en.matplotlib.net/stable/tutorials/colors/colors.html
[48] M. Waskom. Choosing color palettes. Seaborn. [Online]. Available: https://seaborn.pydata.org/tutorial/color_palettes.html
[49] J. Levy-Kramer, "k-means-constrained," Apr. 2018. [Online]. Available: https://github.com/joshlk/k-means-constrained
[50] Media resolution. Google. [Online]. Available: https://ai.google.dev/gemini-api/docs/media-resolution
[51] J. Wu *et al.*, "Visual prompting in multimodal large language models: A survey," *arXiv preprint arXiv:2409.15310*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2409.15310

A

Appendix 1

A.1 Prompts

This section includes all prompts used in the project. Be aware that when curly braces surround text in capital letters, that is a placeholder in the prompt where some situation specific information is inserted.

A.1.1 Bounding Box Prompt

The bounding box prompt is supporting information about the positions of the bounding boxes. The following prompt is an example with eight segments and their relative positions. Supporting information on this format is inserted into multiple prompts.

```
The color segments have the following bounding boxes on the format
[ymin, xmin, ymax, xmax] with values in range 0-1000:
 - The red segment is located within the following bounding box
   [530, 507, 590, 537]
 - The blue segment is located within the following bounding box
   [393, 447, 484, 507]
 - The yellow segment is located within the following bounding box
   [606, 462, 621, 537]
 - The green segment is located within the following bounding box
   [393, 492, 469, 611]
 - The orange segment is located within the following bounding box
   [590, 537, 621, 567]
 - The purple segment is located within the following bounding box
   [484, 388, 621, 477]
 - The cyan segment is located within the following bounding box
   [484, 477, 530, 537]
 - The pink segment is located within the following bounding box
   [469, 522, 575, 626]
```

A.1.2 Adjacency Prompt

The adjacency prompt is also supporting information, but is only used in the segment classification prompt. It contains information on which segments are adjacent

to other segments. The following prompt is an example.

```
The colored segments are adjacent to each other in the following way:
 - red is adjacent to cyan
 - blue is adjacent to purple
 - blue is adjacent to cyan
 - blue is adjacent to green
 - yellow is adjacent to purple
 - yellow is adjacent to orange
 - green is adjacent to blue
 - green is adjacent to pink
 - orange is adjacent to yellow
 - purple is adjacent to cyan
 - purple is adjacent to blue
 - purple is adjacent to yellow
 - cyan is adjacent to blue
 - cyan is adjacent to purple
 - cyan is adjacent to pink
 - cyan is adjacent to red
 - pink is adjacent to cyan
 - pink is adjacent to green
```

A.1.3 Segment Classification Prompt

This prompt is used to classify segments, which forms the basis for the symbol detector. In NR OF COLOR SEGMENTS, the numerical value of the number of color segments is inserted. COLOR NAMES is a list of color names of all colors used for a specific prompt. ADJACENCIES and BOUNDING BOXES are replaced with the previously mentioned adjacency prompt and bounding box prompt, respectively.

```
You are given a P&ID where some segments have been colored (they were
originally black). The first image is a big crop of the schematic second
image is a close up-crop of the colored segments. If you want more detail
look at the second image. If you want more context for the colored segments
and how they relate to other parts of the schematic, them look at the first
image. The third image is the same close up crop as the the second image
but the third image is not colored and no text is removed. Use the third
image to see if the colored segments are part of any symbol. The bounding
boxes are for the second and third image (the close-up crops).

The colors them self does not mean anything, they are only used for guiding
this prompt. All black pixels in the images has been divided into small
segments. Currently {NR OF COLOR SEGMENTS} segments have been
randomly picked and colored in a unique color. The colored segments does not
have to belong together. You should analyze segments with the following
```

```
colors: {COLOR NAMES}. All these colors exists in the provided image and if
a color cannot be found, then do not mention it in the JSON answer.

{ADJACENCIES}

{BOUNDING BOXES}

Your task is to determine for each segment whether it is a part of a symbol
or non symbol (other). Remember that the colored segments does not have to
belong to the same symnol or non symbol. One segment can be a symbol and
another one can be a non symbol (other). Also Remember to not separate
between the colors. For example blue is deep blue, cyan is bright blue and
purple is darker than pink.

symbol:
 - e.g. valves, tanks, inlets or outlets.
 - outlets to the outside of the page/schematic.
 - inlets from the outside of the page/schematic.

other:
 - Connections between symbols like pipes.
 - Mid-arrows (that indicate the flow of a pipe).
 - Arrow symbols.
 - Boundary lines (e.g. unit/package/functional boundary).
 - Anything that is not a symbol.
 - Any segment that is a part of a text character.

For a segment to be classified as a symbol, the entire segment has to be a
part of the symbol. For example, a yellow segment is a part of a symbol if
the  yellow pixels belongs to a symbol.

["symbol", "other"] are the only alternatives when answering. Make a new
row for each segment segment you are analyzing. You write out your reasoning
in one sentence per colored segment about what function it has in to P&ID
and then answer on the following JSON style format:
{{
    "orange":"symbol",
    "blue":"other",
}}
```

A.1.4 Merge Prompt

The merge prompt is used in the merge step of the digitization pipeline. Similarly to the bounding box prompt, the merge prompt also includes a bounding box prompt for additional positional information. OTHER COLORS is replaced with a list of the colors of the other segments that are being investigated.

You are given a P&ID where some segments have been colored (they were originally black). The first image is a big crop of the schematic second image is a close-up crop of the colored segments. If you want more detail look at the second image. If you want more context for the colored segments and how they relate to other parts of the schematic, them look at the first image. The bounding boxes are for the second image (the close-up crop).

The colors them self does not mean anything, they are only used for guiding this prompt. All black pixels in the images have been divided into small segments. Currently some segments have been colored.

{BOUNDING BOXES}

The green segment is the segment of interest. Your task is to predict which of the other segments colored {OTHER COLORS} belong to the same symbol as the green segment.

A color belong to a symbol if parts of the symbol is covered with that color. The green segment should maybe belongs with some different colored segment(s).

- E.g. if a the of valve symbol is colored green and another part if the same valve symbol is colored blue. Then answer 'blue':'green' because they belong to the same symbol.
- E.g. if the segments colored green belongs to a tank and the red colored segment belongs to an instrumentation. Then do not answer anything with 'red' since the green and red segment belongs to different symbols.
- If the green segments belong to one valve and the blue segments belong to another valve, then do not answer anything with 'blue' since these two segment belongs to two different symbols, even thought they are the same type of symbol.

Write color name in {OTHER COLORS} together with 'green' if that color belongs to the same symbol as the green segment. View the answer as a list of segments that should be merged.

You write out your reasoning in two sentences and then answer on a new line in the following JSON style format:

```
{{
'red':'green'
'blue':'green'
}}
```

A.1.5 Classification Prompt

The classification prompt is used to classify segments. Similarly to the two previous prompts, it includes bounding box information. CATEGORIES is the placeholder for a list of all categories to which a symbol can be assigned. CATEGORIES EXCEPT GENERAL is the same list but without the category general.

```
You are given a P&ID where some segments have been colored red (they were
originally black). The first image is a big crop of the schematic second
image is a close-up crop of the colored segments. If you want more detail
look at the second image. If you want more context for the colored segments
and how they relate to other parts of the schematic, them look at the first
image. The bounding boxes are for the second image (the close-up crop).

The colors them self does not mean anything, they are only used for guiding
this prompt. All black pixels in the images have been divided into small
segments. Currently some segments have been colored red.

{BOUNDING BOXES}

Your task is to determine what kind of symbol the red segment belongs to.
The alternatives are {CATEGORIES}. Answer 'general' only if none of
{CATEGORIES EXCEPT GENERAL} match the red segments. {CATEGORIES} are the
only alternatives when answering. The red segments belongs to one of them.

You write out your reasoning in two sentences and then answer on a new line
in the following JSON style format (these are two separate examples):
{{'red':'valve'}}
{{'red':'general'}}
```

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY